UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

BAYESIAN DEEP NEURAL NETWORKS FOR PREDICTIVE MAINTENANCE UNDER UNCERTAINTY

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MECÁNICO

IGNACIO EDUARDO RAPOSO VOGEL

PROFESOR GUÍA:
ENRIQUE LOPEZ DROGUETT

MIEMBROS DE LA COMISIÓN:
RAFAEL OMAR RUIZ GARCÍA
VIVIANA MERUANE NARANJO

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: IGNACIO EDUARDO RAPOSO VOGEL
FECHA: 2020
PROF. GUÍA: ENRIQUE LOPEZ DROGUETT

BAYESIAN DEEP NEURAL NETWORKS FOR PREDICTIVE MAINTENANCE UNDER UNCERTAINTY

En un mercado global crecientemente más competitivo, bajar los costos operacionales y asegurar la estabilidad operacional ha saltado directamente como prioridad en las decisiones estratégicas de administración relacionadas a producción y mantenimiento. Una gran cantidad del costo operacional en escenarios industriales está ligado a mantenimiento, generando un cambio en las políticas de mantenimiento para maximizar la vida útil y confiabilidad de los activos físicos cambiando de correctivo a preventivo, y predictivo cuando sea posible [Mobley., 2002]. Esto, en adición a el incremento exponencial de equipos de monitoreo a bajo costo, ha generado mucha investigación ligada a Ingeniería de Confiabilidad, donde los objetivos principales son incrementar la disponibilidad de planta, vida útil del equipo y manejo optimizado de inventario de repuestos, incrementando la eficiencia del proceso.

Machine/ Deep Learning es un área de las ciencias de la computación que usa modelos matemáticos para extraer y aprender patrones a partir de datos, siendo extremadamente útil para tareas repetitivas con comportamientos complejos. Para la Ingeniería de Confiabilidad, este campo ha presentado maneras innovadoras para construir herramientas para tareas de diagnostico y pronostico de activos mediante el monitoreo de su degradación.

Independientemente, estos modelos tienen algunas desventajas dada su naturaleza determinística que tienen que ser solucionadas antes de su aplicación en el mundo real. Los puntos pueden caracterizarse como: La falta de interpretabilidad[1] de los modelos a medida que crecen en complejidad, y la susceptibilidad de emitir predicciones sobre-confiadas a eventos fuera del conjunto de entrenamiento. Mientras, las aplicaciones que son expuestas a riesgo en situaciones reales, desean por modelos escalables y robustos que sean capaces de manejar estos eventos, o, al menos, proveer explicaciones o metricas de seguridad sobre sus decisiones.

Una de las soluciones propuestas, es la de utilizar modelos que sean capaces de emitir una aproximación de la incertidumbre sobre sus propias predicciones, indicando la falta de conocimiento. El campo de Deep Learning ha introducido una solución escalable en la forma de Redes Neuronales Bayesianas [Blundell et al., 2015]. Esto permite atacar las reservaciones sobre la aplicación de estos modelos en el mundo real, donde una medida de la incertidumbre de predicción es un requerimiento para la aplicación de cualquier modelo.

Este trabajo propone las Redes Neuronales Bayesianas Profundas como una alternativa factible y ventajosa desde el punto de ingenieria de confiabilidad aplicada, estudiando un caso con un alto nivel de incertidumbre correspondiente a el de diagnóstico de fallas mediante muestras de aceite de varias familias de equipos de diferente naturaleza. Luego, es comparado con otros algoritmos determinísticos de punta para exponer las ventajas y desventajas de realizar mantenimiento predictivo sobre activos físicos con modelos probabilísticos.

[1] *Interpretability is the degree to which a human can understand the cause of a decision.[Miller, 2017]*

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: IGNACIO EDUARDO RAPOSO VOGEL
FECHA: 2020
PROF. GUÍA: ENRIQUE LOPEZ DROGUETT

BAYESIAN DEEP NEURAL NETWORKS FOR PREDICTIVE MAINTENANCE UNDER UNCERTAINTY

In an increasingly more competitive global market that causes overall lower utility margins, lowering operational costs and ensuring operational stability of processes has jumped directly into the top priorities on almost all strategic management decisions related to production and manufacturing. A large part of the total operating cost on industrial scenarios is used in maintenance. This has generated a change on the maintenance policies to maximize physical assets reliability, shifting from corrective actions to preventive, and even predictive where possible [Mobley., 2002]. This, added to the exponential increase of low cost monitoring technology for diverse process variables, has generated lots of research related to reliability engineering, where the main objectives are to increase plant availability, equipment lifetime and optimized spare parts handling, which ultimately, increases production efficiency.

Machine/Deep Learning is an area of computer science that aims to use mathematical models to extract and learn patterns from data, being extremely useful to automate repetitive tasks and monitor certain patterns associated with complex behaviors. For Reliability Engineering, this specific field has presented innovative ways to build useful tools for tasks related to the diagnostics and prognostics of assets and process degradation monitoring.

Nevertheless, this models have some downsides due to their deterministic nature that have to be addressed before deployment in the real world. These main points to be addressed can be characterized as the lack of interpretability[2] of the models as they grow in complexity, and, the susceptibility to make over-confident predictions to out of training distribution events. Applications that involve high risk decision making in real situations, crave for scalable robust models that are able to handle this events and provide some sort of explanation over the confidence of their decision when deployed under uncertain conditions.

One of the solution proposed is of algorithms that are able to output a measure of uncertainty over their predictions that indicates the lack of knowledge of the model over the prediction. The Deep Learning research field, has introduced a scalable solution in the form of Bayesian Neural Networks [Blundell et al., 2015]. This allows to address key reservations on the application of these models in the real world, where a measure of prediction uncertainty is a requirement for any robust model to be deployed.

This work proposes a Bayesian Deep Neural Network as a feasible and advantageous alternative from an applied reliability engineering standpoint, studying the case of a high uncertainty fault diagnostics task related to off-line laboratory oil analysis of several equipment families of very different nature. Then, it will be compared with other available state-of-the art deterministic algorithms to expose the advantages of performing predictive maintenance over assets under uncertain conditions.

[2] *Interpretability is the degree to which a human can understand the cause of a decision.[Miller, 2017]*

They say
'Luck Favors the Bold',
I say,
'I don't need Luck when Patience Rewards the Brave'

Acknowledgements

To my dear family, that has supported with me on every idea and crazy adventure I have thought of. Thank you for showing me the world, waking my critical thinking, also, that there are more important things to ambition than money and success in this life.

Thanks to my dad for showing me that life is about going on adventures and enjoying every moment, that you can always learn of every little thing, and, that the secret of life is nothing more than facing everything you do with eagerness, humility, curiosity and the best of attitudes.

Thanks yo my mom for waking on me the scientific side of things, the researcher, critic thinker. Also, for teaching me to be generous, brave, bold and patient, to never back down when you have arguments to be right, to share and protect anyone who needs my help. Overall, to be a good, just, leader.

Thank you to my sister for being my nemesis and my 'bro', we interactively fight, but there will always be love and care on that bond. She has been there whenever I have needed her and vice versa. It fills my spirit to know that I got her unconditional support on my endeavors, and I would like her to know that she has mine.

I thank specially Daniela Caicedo, which has been my partner in crime for so long, enduring the good and the bad times. You always manage wake a smile on me. Thank you for being not only my girlfriend, but my confident and best friend.

Finally, I want to give special thanks to my grandmother and great-aunt, may them rest in peace, for raising me and give me inspiration to follow my dreams. You wisdom shall be passed on to newer generations, and you will always be in my heart.

Contents

List of Tables

List of Figures

Objectives

General Objective

Developing and Evaluating Bayesian Deep Neural Network for Machine Health State Diagnostics Under Uncertainty.

Specific Objectives

1. Setting an adequate pre-processing pipeline for the supervised task based on a real world database comprised of laboratory analysis performed over off-line lubricant samples from different mining equipment to determine their fault state. The adequate pre-processing of this data is crucial for the posterior model training and evaluation.
2. Developing, training and calibrating the Bayesian Deep Neural Network on the classification task.
3. Determine the feasibility of these networks to be used in real world operational context and compare with other algorithms commonly used from machine learning, such as Decision-Tree based models and regular neural networks.
4. Show the advantages of having a Out Of training Distribution (OOD) samples filtering methods contributing to an increase in the model's prediction reliability during the decision process.
5. Exploring the advantages of having a metric that quantifies the lack of knowledge of the model about a respective prediction on real world uncertain settings.

Methodology

1 Resources Used

Pre-processing and training phases are coded in Python 3.6.8 programming language on a dedicated python environment running in a Linux-based server.

The following publicly available high level open-source libraries were used for this work:

- SciPy: Statistical & Preprocessing High Level Functions
- NumPy: Array Manipulation
- Pandas: DataFrame Manipulation
- GPy & GPyOpt: Bayesian Optimization
- MatPlotLib: Data Visualization & Plotting
- Seaborn: Data Visualization & Plotting
- Tensorflow 1.14: Neural Networks Building, Training & Inference
- Tensorflow Probability: Probabilistic Neural Networks Building, Training & Inference
- Sci-kit Learn: Preprocessing & Machine Learning Models Building, Training & Inference
- XGBoost: Extra Gradient Boosted Building, Training & Inference
- LightGBM: Light Gradient Boosted Machine Building, Training & Inference

The utilized server computational power characteristics are presented in Table 1, where all the running times stated in this work are based on these computational attributes.

Component	**Resources**
CPU	Intel Quad Core i7 7700K 4.5 MHz
GPU	Dual Nvidia 1070Ti SLI 8Gb
RAM	64Gb RAM 3200MHz
Disk	WD 1Tb SSD PCIe NVMe M.2

Table 1: Computational Characteristics of Server

All preprocessing steps, models training and testing are performed on this server. There was no need for any experimental setup or physical tools.

2 Work Methodology

For this work, the methodology followed can be decomposed in 5 main phases.

2.1 Task Study and Planning

In the first place, as in any engineering sciences problem, familiarizing with the information, process and task in order to grasp the key insights is crucial. Studies were undertaken with the objective of understanding the oil analysis diagnostics process in general, the features present in the data along with their significance, and, finally, the specific lubricant analysis and diagnostics process performed over the data provided. Some of the sources used as base study are [Ebersbach and Peng, 2013] and [Newell, 1999].

Any emerging doubts about the data, procedures and denominations were cleared with the data provider, MCM Ingenieria, before any programming was done in order to determine the best alternatives available to efficiently approach the task.

In this stage the assumptions that govern the task objective and data preprocessing are generated, which will ultimately influence directly the results obtained.

Given the task had a label associated, the problem was deemed to be appropriate for a supervised learning classification task. The root cause of each of the 6 major faults was studied in detail with help of the data provider.

2.2 Data Preprocessing

Dataset Overview & Cleaning

Usually, given the iterative nature of changing assumptions and data exploration, this stage takes approximately from 70% to 90% of the projects time.

The diagnostics dataset was kept on different file than the oil sample laboratory features; therefore, each of the sample's diagnostic was correlated to its respective laboratory obtained features using the ID lab correlation code that paired the sample with its label.

As the the dataset provided kindly by MCM Ingenieria, corresponding to 14,302 laboratory tests performed over a specific oil sample, was not intended for machine learning purposes when it was conceived, the information contained in the original '.csv' files needed a previous stage of cleaning given several redundant/unnecessary columns which were discarded as they held no importance under a machine learning scope.

In this step, a target column was added with the machine condition diagnostic to the feature data, while seven columns with no-structured data type (text) referring to each specific equipment description, equipment location, fault description, recommendations, etc. was discarded given they added no information directly usable for the physical asset condition diagnostics.

Given that a core principle in the machine learning community is to try to keep as much data volume as possible while cleaning the dataset's, a second step of cleaning was performed

to rule out any columns that had a significant lack of data, as in some case not all the laboratory tests were performed on a sample, leaving many empty fields.

Therefore, any feature column with over 10,000 empty values was discarded, as most machine learning models are not able to handle missing data. The result is a data format where each sample has 29 laboratory test derived features, a feature that states which equipment family does the sample comes from, a feature column for the ID tag of the specific sample, and a target label column.

The pre-processing steps up to this point is summarized in Figure 1.

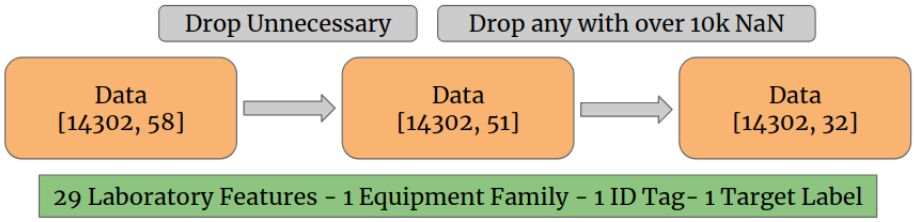

Figure 1: First Preprocessing Steps & Final Feature Columns

Family Grouping

In the Family feature column, equipment's have over 60 different denominations in the raw data. After a careful analysis with MCM engineers, it was assumed that they could be grouped into 4 main families corresponding to Pumps, Hydraulic Units, Shafts and Gears. To deal with the unstructured data, an algorithm was made to change the specific denominations into one of the 4 major families by pivoting on hierarchical key words present in the feature column.

This is done with the objective of further studying the implications of performing diagnostics specific to each equipment family, as there should be less variance introduced into the data. Under this pretense, performing diagnostics on a specific equipment family should improve the results, but there is an important loss of data volume if each case is analyzed separately.

Data Augmentation

The only data augmentation practice that was possible to do was in the case of the Water Contaminated labels samples, where they had the particular situation in which the particle counting laboratory tests were not filled/performed on the sample once water contamination was detected.

In order to keep a large part of this samples in the dataset, an average particle counting test value was extracted from the healthy samples to fill the empty fields. This can be done without loss of generality or biasing the samples in a considerable amount, because the base for this fault state relies on water concentration, in parts per million, present in the lubricant; ergo, as there is no important influence of the particle counting in the diagnostic, it can be

assumed that a healthy amount of particles will neutralize the effect of this specific missing features, while allowing to keep the samples for more robust insights.

Some of the equipment did not belong into any of the major families, and the Family field was left empty as a NaN.

Not-a-Number Filtering

In this step, all the rows that have one or more NaN are filtered from the dataset. This generates a mild loss of data, but it is imperative in order to have data clean for the feasibility analysis and posterior training phases. After discarding all rows with one or more NaN in any feature column, the dataset is reduced from 14,302 samples to 12,838 clean samples.

Extreme Outlier Filtering

Given than this dataset has a very large variance, due to the different operational conditions and equipment families present in the samples, and there is no certainty that all the features have been transcribed correctly into the database, it is necessary to disregard any outliers that might excessively escape the distribution of samples adding extra undesired variance. This step is performed to to ensure a smoother feature scaling, aiding in the training phase to obtain better results with each model. The method used for this step relies in calculating the Z score for each fault state (label) and discarding any samples that excessively escape the distribution of these. The assumption that they all are generated from a Normal distribution, as shown in Figure 2, is undertaken and any sample that escapes from 4 σ standard deviations (99.9% of respective samples distribution) from the mean is discarded.

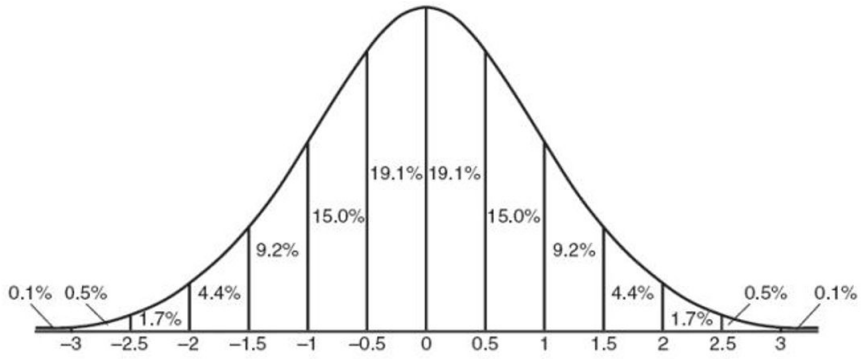

Figure 2: Normal Distribution - Standard Deviation Area Under Curve

Finally, this step only discards a few samples, concluding the preprocessing stage obtaining the final clean dataset of 12,504 samples which will be used in the following phases.

Train & Test Separation

A holdout set of 25% of the dataset is removed from the dataset to be used as a Test set on the trained models. The rest of the 75% of data remaining, is determined to be the Training set for this task.

Scaling

Feature Scaling is performed to each feature independently in order to decrease the effect of different magnitudes present on different interacting features to obtain proportional variations. This improves substantially the pattern identification and learning process, generating better model performance.

In this case three different scaling methods were analyzed: MinMax, Standard and Robust scaling. Given that the first two are affected greatly by the variance of the dataset, Robust scaling is chosen, as it dramatically decreases the effect of outliers by giving greater weight to the feature samples between the 1st and 3rd quantile, and less influence to the ones in the 4^th quantile.

2.3 Feasibility Analysis

On this phase, the feasibility of the resulting data is analyzed to determine if the assumptions made have resulted in an information-rich dataset that has a good learning potential.

Data visualization and feature importance analysis are performed to assess and revise the assumptions made so far, and determine improvement possibilities such as further eliminating uninformative features, possible relabeling opportunities for better clustering, dimensionality reduction to avoid sparsity, identify data patterns, etc.

Visualization

As a machine learning classification task can be defined as finding the optimal dimensional separation between all the label classes, some techniques that reduce the dimensionality of features are used for visualization purposes.

The objective of this step is to determine if there are any clearly visible clusters in low dimensions, which increases the probability of finding a better separability on higher dimensions.

With this purpose, two commonly used algorithms are tried; Principal Component Analysis, which is based on a feature variance linear algebra decomposition of the eigenvalues and eigen-vectors for projection on new component planes; and Linear Discriminant Analysis, which is based on finding the plane which maximizes separability amongst classes. An example of the results obtained can be observed in Figure 3.

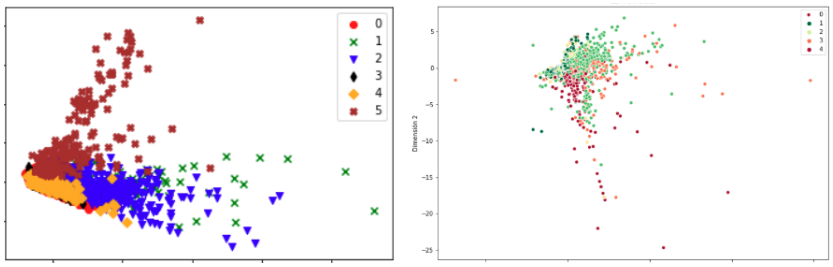

Figure 3: PCA (left) and LDA(right) Task Data Plots

Dimensionality Reduction & Feature Importance Exploration

For the stage of assessing the possibility of performing a dimensionality reduction by discarding some uninformative features, several tests are performed over the dataset.

These analysis aim to find and quantify the relation amongst features and/or weigh their influence on the decision-making process undertaken by a specific model to generate a prediction. This allows to map the hierarchical relations from most-important to least-important features, allowing the justified removal or keeping of features.

The analysis performed on the database are as it follows:

- Pearson Linear Correlation: Measures the linear correlation between two features
- Chi-Squared: Measures the independence between two features
- Recursive Feature Elimination : Sequentially fits the model and then removes the most uninformative feature, saving the performance results.
- Select from Model: Fits the model and then eliminates the features under an influence threshold.
- Random-Forest Based Dendrogram: Evaluates the decision tree feature hierarchy followed by a model to generate predictions, sorted by feature influence in the process.

After finishing this step, the feasibility analysis should have provided enough evidence and justification to generate an idea of the potential of the dataset to train the models, offering insights regarding which models should be used to get results based on data volume, data sparcity, cluster definition and feature importance.

2.4 Model Training

In the training stage, the models are programmed and built, along with their different data-feeding formats. During this stage hyper-parameter tweaking and optimization is performed, with care of not optimizing the model to the specific data introducing a bias that ultimately leads to overfitting the models and poor performance.

To avoid the stated above, the models are trained with a cross-validation of 3 folds during hyper-parameter optimization, meaning that the train set is divided in 3 and 3 models trained with each division, choosing the one with the best average performance in the end.

The Training stage only contemplates 5 out of the 6 fault states, holding out the Oil Contamination, which envelopes several unrelated faults, to simulate as Out of Training Distribution (OOD) events that might happen in real-world operations.

Models

The models which are trained for evaluation are presented in Table 2

Model	Base Concept
Decision Tree	Tree-Based
Random Forest	Tree-Based Bootstrap Aggregating Ensemble
Extra Randomized Trees	Tree-Based Random Split Bootstrap Aggregating Ensemble
XGBoost	Tree-Based Histogram Split Regularized Gradient Boosting
Light Gradient Boosting1	Tree-Based Histogram-GOSS Split Gradient Boosting
Multi Layer Perceptron	Neural Network
Flipout Bayesian Multi-Layer Perceptron 1	Bayesian Neural Network
Convolutional 1D 1	Convolutional Neural Network
Flipout Convolutional 1D	Bayesian Convolutional Neural Network

Table 2: Models Evaluated for Performance Comparison with Model Base Intuition

Bayesian Hyper-parameter Optimization

Hyper-parameters are fixed values that control how the model works, like initial conditions and estimator architecture settings defined before the beginning of the training phase. These define the learning power and learning mechanisms of the model. The more user adjustable settings the model has, the search space for this values increases combinatorially, as most of the times they are not independent from other hyper-parameters. Depending on the values selected for the hyper-parameters will most likely result in different models.

Hyper-parameters are a mixture of Boolean, discrete and continuous values that set different initial conditions whose nature varies from model to model. Some usual techniques that are progressively more computationally expensive as more hyper-parameters are added are Grid Search or Random Search; which are respectively based on exploring an user-set number of hyper-parameters for a combinatorial number of iterations based on the possible user defined combinations amongst the desired search space, or explore randomly the hyper-parameters over a set number of iterations.

Both of this methods return the results of a score function defined by the user for each of these models along with the hyper-parameters of the respective iteration.

Bayesian hyper-parameter optimization takes advantage of previous information about the hyper-parameters effect on the target score obtained through exploration, in order to exploit specific relevant areas of the search space that have the potential to provide an increase on the model's performance. Bayesian Hyper-Parameter Optimization balances the trade-off of exploration vs. exploitation.

As it progressively evaluates more combinations, the Bayesian Optimization model learns

how the values of different hyper-parameters affect the target result, guiding the value selection for hyper-parameters through the search space by updating the prior belief based by the current result. This generates the ability for this method to to provide a much faster and precise approximation for the best model hyper-parameters.

The objective is to create a surrogate model approximation of the error as a function of the hyper-parameters, an iterate through the results until convergence into a maxima for the target function where the change in values is sufficiently close that the impact on the score is small.

Basically, the optimization process is performed as it follows, observed graphically in figure 4:

- Using previously evaluated points it computes a posterior expectation for the target score.
- Samples the values at the computed expectation
- Compares the previous model score with the current score and accepts the hyper-parameter for the prior probability update if the result improves.
- Repeat until user-set convergence tolerance between variations in the hyper-parameter values is achieved.

For the process is important to note that to allow further exploration after converging locally to a relevant area of the search space, and escape from local minima, a parameter is set in the optimization process to sample, with a user set probability, a random point anywhere in the search space not based in the prior.

In short, the method's objective is to first canvas the search space and rapidly identify the most relevant areas to explore, in order to converge to the hyper-parameters that generate the best results for the model. It predicts given a prior probability how hyper-parameters will behave, and then samples a point from the parameters space which is most likely to improve the results, making an update of the prior belief based on the ongoing samples.

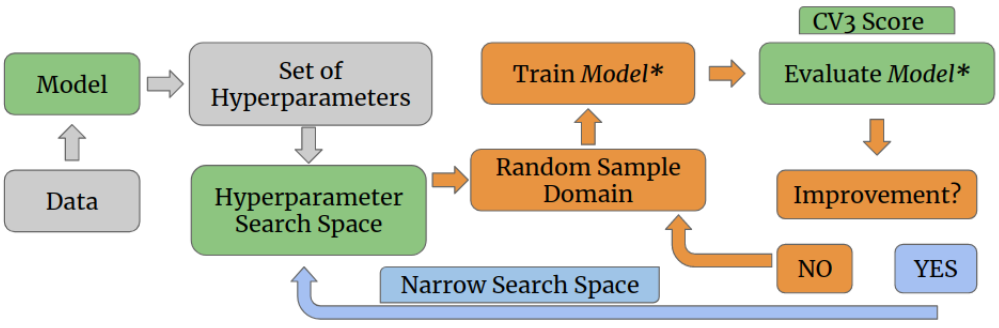

Figure 4: Flow Diagram of Bayesian Hyper-Parameter Optimization

Each model's Bayesian Optimization is exhaustive, meaning that it envelopes all the user defined hyper-parameters over a large search space with maximum of 1500 iterations if not stopped earlier, and each is run 4 times for each model. The target score function is defined to be the average of a 3-fold cross validation split.

Final Training & Early Evaluation

The best scoring model hyper-parameters are kept for definitive training. Each optimized hyper-parameter model is trained 5 times to asses stability and prediction consistency, with a validation set of 20% of the train data.

The holdout Test Set is scaled in relation to the Training fit preformed during the data preprocessing stage.

In Figure 5, the final optimized model training and evaluation loop is shown, which encompasses the final training phase and the early evaluation phase; mainly to detect if the model has a consistent generalization capability, or is prone to over/under-fitting. In this case, the confusion matrix and F1 score of the train, validation and test data are compared, hoping to have very similar results.

In the last case, the training process should begin again, while doing a detailed analysis of why the performance was not as expected.

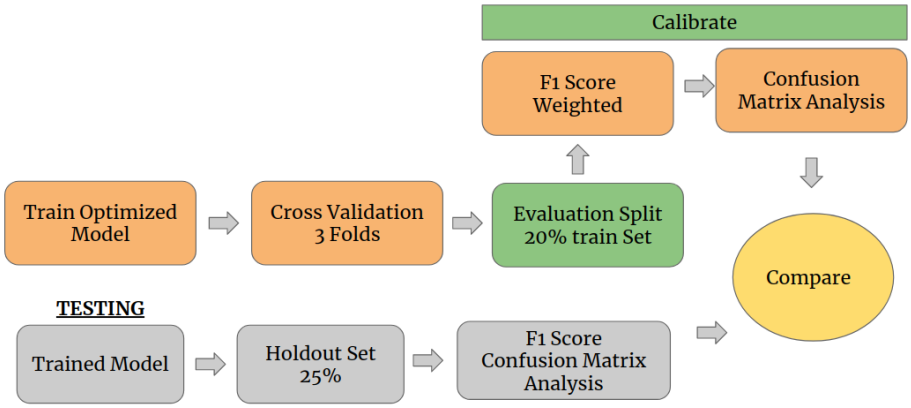

Figure 5: Flow Diagram of Training-Evaluation Loop

2.5 Evaluation & Interpretation

Evaluation General Metrics

In problems where the classes are imbalanced, the accuracy metric is not reliable enough to be applied confidently for model evaluation. Therefore, in order to compare between models, F1 Score, a metric based on the balance of Precision and Recall, is used instead.

Also, as a general comparison evaluation method, the Confusion Matrix is an invaluable tool to calibrate, evaluate and interpret the model's learning. It allows to interpret the reliability of each predicted class, providing insights about the mistakes made in the classification task, and if there is an underlying pattern in data that is inherent of the data or preprocessing stage.In other words, it allows to extract important insights from the quality of the model's learning, generating the opportunity to make decisions to improve the model performance in the next training iteration.

All the model's performance is compared in terms of F1 Score and Confusion Matrix Interpretation.

Probabilistic Interpretation

Along with the general evaluation metrics stated above, the probabilistic neural networks have extra capabilities for result interpretation, which are based on the Median and Standard Deviation issued with each prediction.

The reason for this is that each sample is passed 100 times through the model, which samples the learnt parametric distributions each time, getting a different answer. The more certain the prediction is, the closer the predictions will be, depicting the models knowledge about a specific sample, while in the opposite case, the model's lack of knowledge will be represented as a high variance of results.

Setting a threshold filter over the predictions, provides the opportunity of discarding any predictions that do not comply with the user-set values for any of these 2 statistical parameters. This prediction filtering affects the F1 Score and the confusion matrix obtained with the original predictions, where when calculated again as a F1 $\text{Score}_{\text{threshold}}$ and a Confusion $\text{Matrix}_{\text{threshold}}$, the results tend to improve, as the uncertain and under-confident predictions are set aside from the predictions pool.

The decision loop followed to calibrate the threshold values is shown in Figure 6, where the general prediction STD of the test set can be obtained and explored to set an according threshold. On the other hand, the median can be set based on an user defined criteria that depends solely in the trade off between model reliability vs. rejected samples.

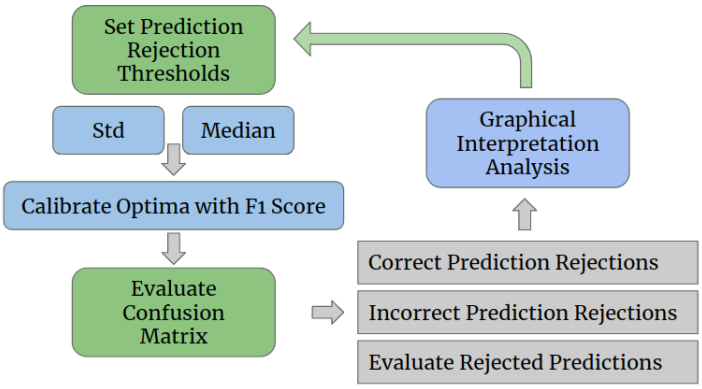

Figure 6: Flow Diagram of Threshold Value Setting and Analysis

Graphical Interpretations

The opportunity of graphical interpretation is generated with the probabilistic neural networks, which is a scatter plot based in all the 100 outputted predictions from the softmax[3] function, this characterizes graphically the median and variance distribution for each class, allowing to extract more in depth interpretability and insights from this kind of model.

[3]Mathematical function at the output of neural network classification problems that outputs the probability of belonging to each label class.

Abstract

In an increasingly more competitive global market that causes overall lower utility margins, lowering operational costs and ensuring operational stability of processes has jumped directly into the top priorities on almost all strategic management decisions related to production and manufacturing. A large part of the total operating cost on industrial scenarios is used in maintenance.

Machine/Deep Learning is an area of computer science that aims to use mathematical models to extract and learn patterns from data, being extremely useful to automate repetitive tasks and monitor certain patterns associated with complex behaviors. For Reliability Engineering, this specific field has presented innovative ways to build useful tools for tasks related to the diagnostics and prognostics of assets and process degradation monitoring.

Nevertheless, this models have some downsides due to their deterministic nature that have to be addressed before deployment in the real world. Applications that involve high risk decision making in real situations, crave for scalable robust models that are able to provide some sort of explanation over the confidence of their decision when deployed under uncertain conditions.

The Deep Learning research field, has introduced a scalable solution in the form of Bayesian Neural Networks [Blundell et al., 2015]. This allows to address key reservations on the application of these models in the real world, where a measure of the model's prediction lack of knowledge is a requirement for any robust model to be deployed.

This work proposes successfully a Bayesian Deep Neural Network as a feasible and advantageous alternative from an applied reliability engineering standpoint, studying the case of a high uncertainty fault diagnostics task related to off-line laboratory oil analysis of several equipment families of very different nature. It is compared with other available state-of-the art deterministic algorithms to expose the advantages of performing predictive maintenance over assets under uncertain conditions.

Section I: Introduction

1 Motivation

Machine and Deep Learning have quickly gotten widespread adoption through many industries that work with standardized information such as economics, banking, user experience, marketing and state-of-the-art automated monitoring.

But, relative to the chemical and mechanical industries, adoption has been slower due to lack of sensors, sub-optimal data acquisition, complicated data storing procedures, constantly changing operational conditions, and doubts regarding its stability, reliability & interpretability during the decision making process.

In recent years, as more complex processes and specialized machines are used in industrial environments with decreasing utility margins, preventive maintenance costs have been progressively rising due to its conservative approach. This generates frequent cases where a lot of components are not exploited to the full extent of their 'useful life', which in turn increases downtime scheduling frequency, both of which impact negatively on the production, causing major expenses that have become a main focus to address for many industries.

The above situation has transversely renewed the interest and focused the attention over the last decades towards Reliability Engineering, and the wide set of tools it proposes to manage machine and process variables to increase process reliability and efficiency. In this specific field, the use of machine/deep learning has been rapidly gaining momentum due to its capabilities to find patterns in a large number of variables to generate precision insights into processes stability and machinery degradation.

Still, Machine and Deep Learning have a couple downsides that hinders their adoption on real-world industrial contexts, which can be enveloped as the lack of interpretability,and concerns about the robustness when subjected to Out-of-Distribution(OOD) samples, which are low frequency or single-time events not contained on the training data. In other words, the result of combining both downsides, generates that mathematical models are prone to over-confident predictions which are not interpretable, and ergo, not suitable to support the decision making process where there exists an inherent risk, or potential cost, associated to making the wrong decision.

In order to tackle these specific problems, along with scalability for Big Data applications, the Deep Learning field has shifted towards the objective of developing a scalable and

reliable probabilistic approach for Neural Networks, which currently are one of the most powerful estimators. The approach used in this work corresponds to a type Bayesian Neural Network[Blundell et al., 2015], specifically Flipout Neural Network[Wen et al., 2018], which rely on replacing the network's deterministic parameters with parametric distributions (Gaussian) trainable via Variational Inference with pseudo-independent weight perturbations.

This allows the model to output a metric regarding uncertainty (lack of knowledge) over a certain prediction, which can help greatly to interpret and assimilate the model into a decision making process, while discarding the prediction when the confidence threshold is not met, proving extremely useful in real-world context industrial operations.

2 Context

2.1 Condition Based Maintenance

'Machine condition monitoring definition is the general topic of health monitoring of machines, which includes fault detection and diagnosis, as well as fault progress tracking' - [Ebersbach and Peng, 2013]

Condition Based Maintenance (CBM) refers to the decision process and scheduling of maintenance tasks based on progressively monitoring information collected through a sensor or indicator of the physical asset operational condition.

The main goal of CBM is to minimize unnecessary maintenance tasks by taking action; hopefully before the asset's condition or behavior is abnormal, through anomaly detection or health state diagnosis, and in cases where feasible, prognosis of the remaining useful life of the asset before need for maintenance.

When successfully applied, it can significantly reduce the maintenance cost by reducing downtime frequency and length, unnecessary preventive operations, while also improving the usage life of machines and components.

More in depth information about this topic can be found in the work of Jardine et al. on 'Mechanical Systems and Signal Processing 20'. [Jardine et al., 2006]

2.2 Oil Analysis and CBM

For machinery in general, lubrication is a critical variable of the mechanical system and is commonly achieved by adding or circling grease and/or oil through the system. Components rely on the protective and heat managing capabilities of the lubricant to reduce wear, minimise corrosion, dissipate heat and clean surfaces on moving components.

Given the core nature of the lubricant on the machine, which is in direct contact with wear-prone internal components, it presents an ideal indicator on which to perform exploration in order to aid in health state diagnostic of the machine.

By analyzing certain performance indicators related to changes in concentrations and physical properties of the lubricant; such as particle/element concentrations, presence of

external contaminants, and changes of viscosity or PH, very accurate insights can be inferred about the internal condition of the machine components to be used in the maintenance planning process.

Many organizations have included oil analysis as part of their CBM strategy, that when globally applied correctly, can signify benefits of cost reductions up to 30%, by minimising unscheduled downtime and catastrophic failures, all of which is achieved through the machinery oil CBM. [Newell, 1999]

2.3 Oil Analysis Process and Big Data

As a general overview of the classical oil analysis process, it starts with a sample taken from the asset on a regular time interval and sent to a laboratory where it is processed. The extent of the tests done on the sample vary depending on the scope and objective of the information that is required from the sample.

This results are then analysed and interpreted by an expert, usually aided with rule-based mathematical models that are either manual or automated, for anomaly detection. Then, a document is issued reporting the findings and the test data stored with its diagnostic.

Even though in the present there is an incipient field dedicated to explore and implement online lubricant monitoring techniques, the general approach to lubricant analysis is mostly done offline, following the same process explained in the paragraph above.

The process is highly precise and informative, producing highly valuable data, but has the downside of being time and human-resource consuming, requiring specific expertise that is highly on demand, where the insights obtained may not be always on time to aid on the decision making process.

This laboratory based analysis of the lubricant relies mainly on determining and assessing the following items [Newell, 1999] [Toms and Toms, 2010]:

- **Physical and Chemical Properties:** viscosity, oxidation, PH level, and additives.
- **External Contamination:** water, coolant, detergents, dust, fuel, carbon deposits, etc.
- **Wear Debris:** ferrous & non-ferrous metals, element concentration analysis, particle count.

These can be used, by comparing to the original lubricant or agreed thresholds, for anomaly detection, fault diagnostics, or prognostics tasks depending on the data acquisition and usage process.

On the modern era of Big Data, there has been an exponential increase on information management, storing, and services, generating an interest from the industry for ways to exploit the possibilities for analysis and automation of repetitive tasks relying on experts, such as oil analysis.

Recently, better data management policies of massive industries have allowed for the

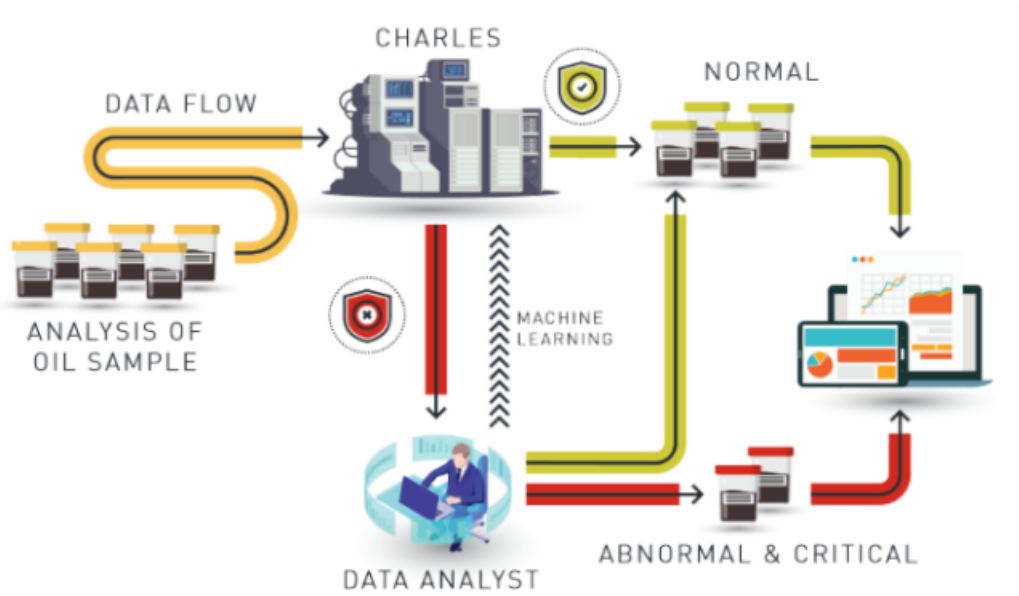

Figure 7: Oil samples Data Acquisition & Analysis Process Flow

collection and storage of significant databases that are very suitable to apply machine learning techniques, given the volume and quality of data they possess.

Specifically on maintenance, the databases, which are labeled and categorized by experts as shown in Figure 7[4], generate a unique chance to use this information to expedite and automate machine health related tasks. The application of these useful tools has the opportunity to add value and efficiency of the maintenance decision-making process, aiding in diagnostics and prognostics tasks.

As there has been a sort of natural filter of the features needed to diagnose the most frequent failures by humans or rule based systems over the long period of time this field has been applied, these features are rich of information that may allow to teach a learning algorithm how to automatically diagnose anomalies, or even the health state, of the oil and machine.

2.4 Advantages of Oil Analysis over Vibration Monitoring

As indicated by [Jingwei et al., 2012], some recent studies have brought to light that frequent monitoring of the condition of the lubricants is better able to predict some malfunctions than vibration based monitoring systems.

For instance, external contaminants, such as water presence on the lubricant, reduce component useful life significantly by decreasing lubricant effectiveness and increasing surface wear. As component wear is usually detected by vibration systems when its severity is high enough to cause perceivable physical signals, the fault has already happened where

[4]Image Source: $https://reliabilityweb.com/articles/entry/why-machine-learning-and-ai-are-the-future-of-oil-analysis$

preemptive action could have been taken to ensure component integrity.

The ideal case, is monitoring both, along with other process variables to make more robust system control, insuring process reliability and efficiency through variable guided effective control and maintenance scheduling.

3 Machine/Deep Learning Approach

Over the last 10 years, due to democratization of AI/ML knowledge platforms and a dramatic increase of computational power, pattern learning algorithms have experienced an exponential increase on research, development and application. The main benefits over mathematical rule based hard-coded models is the ability to learn subtle complex patterns while observing multiple variables virtually impossible for effective human evaluation.

The principal uses of this technology on economic and industrial settings, is the automation of complex repetitive tasks, mainly on classification and regression.

In the area of reliability engineering and maintenance, these field has experimented a widespread adoption given the capabilities it provides for machinery and process management, through the analysis of multiple variables, feature extraction from signals and unrelated features, data visualization and ability to model complex degradation patterns for diagnostics and prognostics.

On asset prognostics and health state management (PHM), classification has proven to be very useful tool for automating and simplifying processes that otherwise required extensive expert knowledge, very specific resources, and were prone to be very time consuming; such as fault diagnosis, health state prognostics and anomaly detection.

This methods extend the opportunity to provide very valuable insights to decision-making process and generate value by efficiently distributing assets, causing an explosion of interest to be applied in different industrial contexts.

Oil analysis requires human experts to diagnose the samples in a time-consuming process, and ergo, expensive. The advantage of having expert automatic systems that can make this same tasks takes a direct impact on reducing the time required to complete them, freeing human assets for more complex tasks.

Being solely numbers that are analyzed in form of concentrations, sizes and flags, the problem is ideal for computers that can capture underlying patterns rather than humans, who are better at calibrating, reasoning and making decisions.

3.1 Challenges of Oil Analysis Automation with Machine Learning

Machine learning, and its sub-area Deep Learning, have a couple challenges compared to traditional algorithms to be successfully applied in real world scenarios, and introduced reliably as a tool to be used in decision making processes. These can be simplified mainly in three dominant areas which are Data, Interpretability and Reliability.

Data

The principal challenge for machine learning is having the proper volume and quality of data, where usually the latter suffers on real-world settings due to sub-optimal data acquisition policies or storage.

Directly related to oil analysis, there are very few massive databases of oil analysis, which are composed of multiple equipment of different operational conditions, given that the amount of samples gathered through time from only one equipment are not enough to build an adequate volume database. Also, the frequency of the fault events is not uniform, causing unbalanced health state diagnostics.

The latter, leads to databases composed of multiple equipment operational families, using different oils and additives, but that are all housed under similar diagnosed faults; and most of them are not completely reliable, as they are based on human, or maintenance policy, subjective criteria to label the samples.

This diagnostic portrayed as subjective, because is left to the specific analyst criteria, who might make the decision based on a hierarchical fault tree, missing multiple failures that might be present, or fail to see an underlying pattern, and mislabeling the sample. This makes the quality of the database degrade as incorrect information is introduced, hence, the patterns and conclusions learnt from the algorithms are not entirely representative of the true phenomena.

To get a learning model that generalizes well across all machinery oil samples, having reliable databases is key, and if done right, comparable performance is very hard to be matched by rule-based models. But, there is a direct relationship between the quality of the classic oil analysis and the performance of the methods applied.

Interpretability and Explainability

Despite most Deep Learning models outperform human experts in specialized tasks at current times, they usually work as "black boxes" where the better their generalization capability is, the less explainable and interpretable the model is. [Gilpin et al., 2018]

For most machine learning algorithms, this is detrimental to their learning as they depend on data to find meaningful patterns. And, as most are 'black box', meaning that they do not deliver fully interpretable results, are undesirable for decision processes which involve risking assets and may have a high cost.

When irreversible decision making processes are subjected to risks and costs, having a decision making model that does not provide information about how it arrived to the respective conclusion is highly undesirable.

Loosely, *Interpretability* can be defined as the extent on which the result can be predicted with a certain amount of intuition without knowing the internal mechanics. It helps directly the process of calibration of the model, which guarantees to a certain extent that the results obtained can be trusted with an acceptable degree of confidence.

On the other hand, *Explainability*, is a concept that relies in trying to dig the mechanisms that deliver a certain result, with the objective of dissecting the core mechanisms being used by the model into human understandable terms.

As the tasks grows on data volume, multi-variables, and complexity; the models are more robust, but simultaneously harder to explain **why** they are making the respective decision.

Ergo, *Interpretability* takes a very important place on model deployment in the real-world, as it allows to calibrate the results on a way that they are suitable to be included in the decision making process.

But, if the volume data is not large enough, and/or the quality of it is not particularly good, the model looses credibility on its results, as they lack the arguments of the process in which they arrived to a conclusion.

Model Reliability on Real-World Scenarios

There is another factor that adds up to the challenges mentioned above. This is related to the fact that on real-world scenarios, singular one-time events are usual and, also, a plethora of low frequency events are likely to happen, specially on industrial contexts where the operational conditions are continuously evolving.

When faced with these events, that were most likely not present ion the training data from which the learning algorithm extracted the patterns, the model is prone to extrapolate and make over-confident predictions over situations that it has no knowledge at all about what to do, without any way of alerting about this situations.

Most Machine Learning and Deep Learning algorithms do not have a way of outputting the model's lack of knowledge (confidence) about a certain result, making them susceptible to incur in this kind of over-confident mistakes that may be very costly to emend.

A possible solution to this challenge, is to use probabilistic learning models such as Gaussian Processes, which output a metric of the models lack of knowledge for its predictions, allowing for a more robust prediction. Sadly, GP's are not scalable to large data, narrowing their use to very well defined problems with very few data points and features.

Lately, the highest levels in the field of Deep Learning research have shifted their attention to develop scalable and stable probabilistic neural networks that are able to output a metric of the lack of knowledge over a prediction in order to resolve the Training Out-Of-Distribution problem which affects the deterministic networks, adding an extra layer of interpretability that aids in increasing the potential value of the model in the decision processes.

4 Proposed Work

The goal is to develop a Bayesian Deep Neural Network framework dedicated to machine fault diagnostics via off-line oil analysis. The model should be able to predict the stage of component wear of the equipment in a situation where not all the possible fault states are known, there is a high probability of OOD events, and data training volume is sub optimal.

The specific task to be tackled corresponds to a supervised machine fault diagnostics problem, involving off-line oil samples laboratory analysis from several different real-world equipment families and operational conditions, acquired by a big mining company over a period of 1.5 years of operation.

The objective of the model is to perform an accurate diagnostic of several different stages that lead to Component Wear, that may be progressive or parallel, accelerating the degradation process of the machines. While, outputting a metric of its 'uncertainty' over its own predictions, which allows to discern between reliable answers and guesses.

Overall, an important assumption made by this work, is that the analysis can be performed over a large number of different equipment families (Pumps, Hydraulics, Gears & Shafts), in different operational conditions, using different oils and additives, as the indicators of faults that lead to lubricant degradation, and consequently wear, are applicable for lubricants in general.

The dataset is particularly complex, as it is riddled with improper data acquisition policies due to the fact that it was not intended for Data Analysis and, as mentioned before, is a compilation of very different equipment families oil samples which generates a high variance of the data. Also, the labeling on it is subjective, as many diagnostic decisions were based the maintenance policy of that time, that aimed to maintain production continuity rather than individual equipment health.

Given this, common learning algorithms performance tends to be hinged by the uncertainty over its results, which, to be deployed successfully, would need a high degree of interpretability to overcome this. Hence, the proposed model presents a clear opportunity to facilitate the deployment for automation as a differential diagnosis tool to support the decision process, where high certainty predictions are automated, and low certainty predictions which indicate a lack of knowledge are rerouted to an expert for analysis.

The particular version of Bayesian Neural Network used in this work corresponds to a variant method called Flipout[Wen et al., 2018] that build over the principle introduced by Charles Blundell[Blundell et al., 2015] on 2015 that replaced scalar parameters (weights) for parametric probability distributions. Flipout allows to de-correlate gradients through pseudo-independent weight perturbations over each training mini-batch, that since they can be written and operated in vectorized form, allow for a considerable speedup in training speed and more stable results compared to previous methods.

The performance of the model then will then be compared with several regular and state-of-the-art learning algorithms, allowing to point the advantages and disadvantages of using learning algorithms for this tasks.

In addition, up until now, there is little to no research on machine fault diagnostics through offline oil samples using machine/deep learning, and we hope this work shows that there is a good opportunity for development in the area, as we take a plethora of mixed equipment and manage to get good results on accurate wear and contamination fault diagnostics, results that will improve if a finer selection is made by equipment operational family and better data acquisition policies are implemented. Ultimately, this generates an opportunity for massive value to be added to the maintenance process in the industries.

5 Scope

The scope of this work is to build and explore the capabilities of probabilistic neural networks applied to a real world scenario, tackling a complex problem for the general of learning algorithms, and what advantages introduces in comparison.

The objective is to generate a reliable model based on Probabilistic Deep Learning to aid in the tasks of classifying the machine's fault diagnosis through off-line oil analysis, working as a differential diagnostic tool that can automate repetitive diagnostics, or give a starting point for more complex ones. Making that not only suggests the fault, but also the level of certainty of said decision scalable to big data, allowing for a more efficient use of the resources of the industry.

Then, these results will be compared to several regular tree-based models on the same task commonly used in the industry, such as Decision Tree, Random Forest, Extra Randomized Trees, and also, presently state-of-the-art gradient-based trees Light Gradient Boosted Machine and Extra Gradient Boosted. On the other corner, it will be contended versus regular deterministic deep neural networks of similar architectures to the Bayesian option, in order to establish direct evidence of the advantages and disadvantages of the proposed model.

A more in depth analysis will be also presented on the interpretability of the outputs of the network, with the objective of showing the interpretation process, the threshold introduced and how the results obtained by the the Bayesian Deep Neural Network can be calibrated and validated.

Ultimately, the advantage of this network's output of the 'lack of knowledge', should allow to reroute the under-confident and uncertain predictions to a human analyst for further exploration, optimizing the asset workload and providing useful insights into the maintenance decision processes.

The main issue with deterministic models, is that an Out Of Distribution sample will be predicted as any of the classes it was shown during training, and have no way of handling these events in a reliable manner. Ergo, a model under real-world deployment must have a way of handling OOD's, at least partially in order to be successfully added into the decision making process.

6 Future Potential

With a correct data acquisition policies, preprocessing and inference pipeline in place, there is a huge opportunity for more complex learning models that could add more value to the maintenance process, such as prognosis tools for estimating the remaining useful life based on oil Condition Based Monitoring, and generating optimal automated maintenance scheduling tools.

7 Section Overview

On section Section I, the motivation and general context is introduced to exhibit the general problem and challenges in the industry. On Section II, Related Work in the field discussed, along with similar approaches that aim to work with oil samples for condition estimation and machine fault diagnosis. In Section III, the Background the mathematical intuition of the algorithms is explained along with their strengths and weaknesses.In section IV the general case study and problem outline is detailed , reasons why the machine learning approach is appropriate for the task are presented, along with the challenges found and their proposed solutions. In section V, results and findings are presented. Section VI will correspond to the discussion of the results found on the previous section, along with recommendations of good practices regarding data acquisition and future research that might improve the results obtained. Finally, on Section VII, the conclusions are presented regarding the advantages of the proposed method.

Section II: Related Work

1 Classical Analysis & Rule Based Systems

In terms of lubricant analysis, there has been a long-standing relation between classical analysis and rule based systems to aid, by hand or automatically, the time consuming process which relies on human expert analysis of multiple related variables which are needed to infer the condition of the lubricant.

The analysis is mainly performed by evaluating a combination of indicators that give insights into the condition of the oil, the machine, or both. Nevertheless, most indicators are shared by some faults, progressive on severity of a certain fault, or derived from different simultaneous fault conditions; meaning that an asset might have several simultaneous faults of different severity, increasing the problem's complexity.

This leads that data interpretation by a human expert about the specific fault, without pivoting on known patterns that lead to certain hierarchical important faults, is extremely difficult.

As a consequence, lubricant analysis is mainly performed by comparing if sets of indicators are between certain ranges, or there is a flag on some variable that exhibits an anomalous behavior, while comparing to the base lubricant sample. For further information regarding classical lubricant analysis procedure and rule-based intuition for reliability purposes [Ebersbach and Peng, 2013] has a very complete explanation of the factors and methods that are considered for these analysis in general.

A lot of work has been done in this regard, but usually is respective of the equipment being analyzed locally based on the findings of the maintenance team or the factory maker rather than a generalized method for detection.

Recently, with aid of mathematical models and computers, some automatic computer software rule-based models for lubricant spectrometer oil analysis have been proposed, such as Proportional model [Jingwei et al., 2012] and the Concentration Model[Ma, 1993].

[Jingwei et al., 2012] proposes a rule-based model to diagnose wear and leak faults, validated on an experimental bench engine, where the lubricant was sampled every 5 hours. This system would ultimately allow to perform on-line and off-line condition based monitoring on the asset, minimizing the time of faulty operation, consequently allowing for a fuller and stable operation of the physical asset.

Nevertheless, rule-based models depend on analyzing specific features or concentrations previously observed and identified by human experts based on experience, to aid in the task of estimating oil condition with the objective of determining if it is time for an oil change or an in-depth inspection. As a consequence, rule-based systems most of the times fail to capture , or directly disregard, implicit linear or non-linear relations amongst features that may prove invaluable to deliver a more precise information about the degradation of the oil and the machine.

2 Machine Learning Approaches

Machine learning can capture most of the mentioned linear patterns between variables, but the overall application has not been exploited to its full potential yet due to external factors that are only recently starting to be addressed. The lack of sizable industrial lubricant analysis databases, along with the subjectivity of the human analysis related to fault diagnostics further increase the complexity of automatizing and deploying the models.

The most recent work related to our approach focuses on the application of machine learning to achieve automation on oil analysis process, is centered in two main areas .

The first area relates to the use of machine learning algorithms to diagnose the quality of the oil, with the main objective of optimizing the oil change interval. [Mohamed et al., 2017] proposes a supervised machine learning decision tree-based method to diagnose oil quality on military vehicles with this same objective that achieves very high precision in determining if the oil is Normal, Unsuitable or Degraded. Adding value to the maintenance process by allowing to set up an optimal maintenance schedule that maximizes the useful life of the machine oil, while minimizing the operation with faults and/or degraded oil quality.

The second area uses lubricant analysis, generally based on multiple variables obtained by element spectrometer and particle concentration analysis, to diagnose frequent component faults that speed degradation of physical asset internal components, not the quality of the oil.

For instance, [Phillips et al., 2015] uses a supervised multivariate logistic regression based method and contrasts it against Artificial Neural Networks and Support Vector Machines, arguing about the downside regarding the lack of interpretability of the two previous models mentioned to be applied on industrial decision processes. The task is performed over a database of 400,000 mining truck engine lubrication oil samples to detect potential fault states as Healthy/Not Healthy. The results obtained are very promising considering around 90% accuracy on the model predictions, and even if this model is not the most powerful, it grounds the fact that most of the times lower accuracy but more reliable/interpretable models are preferred on risk subjected decision processes.

This immediately hints the power of inference machine learning holds in comparison rule-based systems by learning complex patterns, but, it also shows the main weakness of the method. More complex fault diagnosis learning algorithms were not found on the bibliographical survey (until the time of writing) as lubricant sample dataset building is an infeasible process to do in only one asset, only achievable by very big companies that have many

similar assets working in parallel, and only applicable to learning algorithms assuming they have the procedures in place to store and use the data correctly.

3 Deep Learning Approaches

Deep Learning, is not only able to capture linear patterns, but also most non-linear patterns on the data; however, deep learning models also need a considerably larger volume of data to be trained successfully.

Over all, there is one industry that has generated a large amount of research in the are of lubricant condition based automatic monitoring with neural networks, which is wind energy generation via massive wind turbines. The main drive for the research is caused by the complexity of performing maintenance work on the mechanical systems associated with energy generation tens of meters over the ground, dealing with massive components in reduced spaces and a very expensive downtime cost.

The goal to maintain stable, efficient and fault-free operation of the turbines is crucial for the operators, ergo, the monitoring of the conditions that might indicate incipient failure with easy fixes are extremely important in order to avoid more significant repairs.

Most of the research focuses specifically on the wind turbine gearbox on-line condition based monitoring and fault diagnostics, which has a lot of moving parts making it probably the most complicated to repair. Most of them focus on monitoring operational conditions and inferring the machine health through either human or automatic analysis.

Methods for automatic anomaly detection and fault diagnostics through neural networks have been proposed and adopted with success due to the large amount of monitoring done in wind turbines through Supervisory Control and Data Acquisition (SCADA) system, allowing to tailor the methods to each specific turbine like in [Helbing and Ritter, 2018] and [Wang et al., 2017]; nevertheless, these systems rely not solely in oil on-line monitoring, but also oil temperature, component temperatures, rotor pitch, generated power and various vibrations to further generate precise health management insights following a fault tree. [Zhao et al., 2018]

The extra information provided by the extra variables makes the health monitoring automation easier, as fault indicators present themselves in many different forms that are related. Adding extra information to the deep learning models, generates more precise insights as it is able to characterize and differentiate specific faults.

4 Overall

Most papers make the argument that the adoption of machine learning for industrial practices has been mainly hindered by the fact that the more powerful the method, the less interpretable it is. This generates distrust from management, specially during the starting stages of implementation and deployment not reaching a level of automation, but rather taking the role of a suggestion.

This approaches are sparsely related to the work proposed, as our work focuses on using very different sources off-line laboratory oil samples to learn patterns that are able to diagnose the assets specific fault. Nevertheless, the wind energy generation field could also benefit from our work to make further improvements current diagnostics capability and add even more robustness to the automatic systems.

Section III: Theoretical Background

1 Machine Learning

1.1 What is Machine Learning?

Machine learning is a field of computer science which studies and applies mathematical models that learn patterns automatically from data without being explicitly programmed. Even though it has been brought back to public mainstream knowledge over the last couple decades, it has existed since the early 1940's, with one of the most popular examples being Alan Touring's 'Touring's Machine' used to crack the pattern of German radio codes during the Second World War.

Over the last few years, the exponential growth of connectivity and information technologies has allowed this area to widen its reach into many previously unexplored areas, achieving state-of-the-art results in many tasks when compared to hard-coded methods.

This has generated major breakthroughs on research, development and application in a plethora of tasks, such as image recognition, natural language processing, signal processing, industrial applications, big data analytics, social network relational graphs analysis, population clustering, behavioral analysis, speech recognition, targeted marketing, business intelligence, etc.

1.2 Tasks

First, it is important to note that machine learning tasks can be divided in 3 main areas: Supervised Learning, Unsupervised Learning and Reinforced Learning.

Supervised Learning is applied when the data has labeled samples, meaning that each vector or matrix of data X has a target value/class Y corresponding to it. On the other hand, Unsupervised Learning is when the available data X does not present a target class Y associated, requiring different approaches in order to determine the proper data segmentation.

Finally, Reinforcement Learning is a field that is based on modelling an environment for a model, called an 'agent', to learn to interact with the environment solving an optimization function based on rewards for good actions, and punishment for bad actions towards the final goal. On this case study reinforcement learning is not used at all.

In the machine learning field, the approach to different problems can be also categorized

in several tasks, regardless of the Supervised/Unsupervised nature, where the ones most relevant to this work are summarized in the following sections[5].

Feature Selection

The feature selection task is predominantly used when building models, selecting the right subset of features that hold the highest amount of information and minimise redundancies by performing statistical, wrapper or regularization techniques that characterize the features interaction amongst themselves.

For example, Chi-Squared feature selection statistical test is used to test the independence of two events, where given two variables, O_i is the observed samples frequency and E_i is the Expected frequency. This test measures how E_i and O_i deviate from one another as shown in Equation1.

$$\chi_c^2 = \sum \frac{(O_\mathrm{i} - E_\mathrm{i})^2}{E_\mathrm{i}} \tag{1}$$

Hereby, the method aims to capture the features that are highly dependent on the response observed, where the higher the χ^2 score the feature is more dependant on the response and more likely to contain significant information useful for pattern extraction by a machine learning model.

Another commonly used statistical test is the Pearson's correlation coefficient, which determines a measure of the linear correlation between two variables. If two variables have a very close correlation coefficient, they can be deemed redundant and one can be dropped since they are transmitting similar information to the dependent variable. the formula is shown below in Equation 2, where the means and standard deviations are portrayed as μ's and σ's, while X and Y are the features tested.

$$\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \cdot \sigma_Y} \tag{2}$$

On the other hand, Wrapper Methods train a machine learning model using a subset of features iteratively and based on the results obtained it is decided to add or remove features from the subset depending on the criteria used on the algorithm. this method is computationally expensive with large feature spaces.

Classification

This is one of the core task for Machine/Deep Learning. Predominantly, this task is of supervised nature as it relies on labels to teach the model to identify patterns that set aside different classes.

The learning mechanism used for the classification relies in the mathematical principles used by different models which have different ways of minimising the error function. But, in

[5]https://vitalflux.com/7-common-machine-learning-tasks-related-methods/

simple terms, a Classification Task is a prediction of a categorical target value Y from the features of a respective sample of features X.

Technically, what the model is learning is the separation hyper-plane boundaries between different classes that maximise the segregation.

Clustering

Clustering is a task dedicated to find patterns in the data that allow to group similar samples(labeled, unlabeled or both), generating a data segmentation for visualization purposes, dimensionality reduction or 'generalized labeling' [6].

On this work, the Clustering task is not used particularly to work clusters in the data, but to take advantage of a technique named Hierarchical Clustering to measure the level of incidence of the dataset's features on the machine learning model's decision making process.

The graphical visualization output of a Hierarchical Clustering task is usually through a Dendrogram, which is a tree-like diagram that shows the sequential merges or splits.

The method used corresponds to Agglomerative Hierarchical Clustering, which starts by considering each feature as a separate cluster, and by measures of similarity between each of them sequentially merges the most similar ones updating the proximity matrix in each iteration. [Park, 2013]

The algebraic principle of this clustering can be expressed as shown in Equation 3. Specifically, the one used for the feature hierarchical clustering was considering Ward's Method, which calculates the sum of the square distances between the two points, shown in Equation 4 .

$$Sim(C_1, C_2) = Max[Sim(P_{\mathrm{i}}, P_j)] \mid P_{\mathrm{i}} \in C_1 \ \& \ P_j \in C_2 \tag{3}$$

$$Ward Sim(C_1, C_2) = \sum \frac{[dist(P_{\mathrm{i}}, P_j)]^2}{|C_1| * |C_2|} \tag{4}$$

This effectively allows to provide a good reference to observe the features impact over the classification performance, and generating the possibility to eliminate redundant or uninformative features in order to perform a dimensionality reduction of the training data.

Regression

Regression tasks are based in the estimation of a continuous target variable value based on the the features available.

Even though this area is not directly related to the work performed, it is a major area in machine that should be mentioned, as eventually the target for more sophisticated models lies on Regression Tasks.

[6] Provided a few labeled samples, label the entire respective cluster with that label to turn an unsupervised to a supervised task

Specially in maintenance, as the ultimate goal of the application of Machine/deep Learning field is to approximate the remaining useful life of a physical asset in order to have very precise maintenance scheduling, cost saving and minimise downtime.

The basic example is Linear Regression, where the aim is to learn the value for coefficients α and β that approximate the best fit to the available data, in order to solve $\alpha \cdot X + \beta \cdot Y = Z$ with the objective to approximate a good Z value over a continuous space of values for X and Y.

1.3 Feature Scaling

Scaling is a transformation of the data preformed in the pre-processing stage, which is also called normalization.

The principle of this step is taking features independently and transforming them based on their characteristics into new ranges that retain the information, but shake-off the differences in magnitudes, units and ranges. This provides a way of attributing equal weight (magnitude) to each feature for the subsequent learning operations.

In models that use euclidean distance or gradient descent as an optimization method, the process can be sped up by scaling appropriately the data, as the sought parameter will descend slowly on large ranges, generating an inefficient minimisation towards the optima when variables are unevenly scaled.

On the other hand, tree-based algorithms are not distance-based models, weighting the features under their own node-split criteria, making them invariant weather the data is scaled or not.

Given than most machine/deep learning algorithms use the euclidean distance between data points, differences in magnitudes, units or ranges is detrimental for the learning process.

Scaling methods used in machine learning are mostly differentiated by their way of treating outliers, which are samples that deviate considerably from the mean of the specific feature distribution.

Some of the most used feature scaling methods are the following.

Standard Scaling

This method replaced each feature's X_{i} value for its Z score out of a standard distribution, as shown in Equation 5, where x' is the new value, x_{i} is the current value, μ_X is the feature's mean and σ_X is the feature's standard deviation.

$$x'_{\mathrm{i}} = \frac{x_{\mathrm{i}} - \mu_X}{\sigma_X} \tag{5}$$

This redistributes the features values with a mean $\mu_X = 0$ and a standard deviation $\sigma_X = 1$, subjected to outlier influence.

MinMax Scaling

In MinMax Scaling the values of the feature are scaled between a user defined range of values, usually [0,1] or [-1,1] as shown in Equation 6, giving full weight to outliers to define the scaling range, which is specially useful when features with hard boundaries are used, such as image data where colors have only 256 bits or a [0,255] range.

$$x'_{\mathrm{i}} = \frac{x_{\mathrm{i}} - min(X)}{max(X) - min(X)} \tag{6}$$

Robust Scaling

It corresponds to a inter-quantile range based scaling technique which is robust to outliers which may often mislead the scaling process in the previous methods.

The principle is based on not affecting the scaling results with large marginal outliers, while keeping this values in the data as in other scaling techniques the practice is to clip outliers before scaling.

As the percentiles move further from the majority of data, it decreases their impact on the normalization process.

1.4 Model Evaluation

In order to evaluate the models performance, a holdout test set is used to make predictions of the classes the model thinks they belong to, and then compare to the labeled samples.

To define the model has good generalization capabilities, several metrics such as Accuracy, Precision, Recall and F1 Score are used to quantify and analyze the model's performance.

Also graphical analysis about the predictions distribution issued like the Confusion Matrix are analyzed to gain further insights into the model's learning. This allows to detect problems like over-fitting or under fitting of the model during the training stage, and addressing this issues.

Overall, this classification metrics are a balance of the True Positives, False Positives, True Negatives and False Positives predictions of a model, and allow for it to be calibrated in certain ways that are dependent of the task to be solved.

A brief mathematical explanation is provided on the following subsection of the metrics and methods used to evaluate classification performance on the Case Study. Each of the Recall, Precision and F1 scores used in the Case Study has a weighted contribution from each of the classes given the dataset's class imbalance.

Accuracy

Is a basic classification metric for binary or multi-class classification. It is the ratio between correct predictions versus the total amount of predictions, and is specially sensible to class imbalance. The formula is shown in Equation 7.

$$Accuracy = \frac{TruePositives + TrueNegatives}{TotalPredictions} \tag{7}$$

Precision

It returns a metric related to the amount of True Positives out of all Positive Predictions. It is a specially good metric when the detection of a True Positive is critical, like in a task related to detecting a nuclear reactor failure, which cannot have the alarm not be raised, the result has to be very reliable in its detection. The formula is shown in Equation 8.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{8}$$

Recall

This metric measures the ratio of True Positives correctly classified. It is specifically useful to capture as many Positives as possible, disregarding potential false alarms, which in the example case of a nuclear reactor, false alarms are preferable to a reactor meltdown. The corresponding formula is shown in Equation 9.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{9}$$

F1 Score

F1 score is the harmonic mean of Precision and Recall, shown in Equation 10, this metric provides a measurement of the trade-off between both of the previous metrics. It is core in assessing the classification performance of a model, specially useful for identifying a well balanced model and comparing with other models in equality of conditions.

$$F1Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{10}$$

Confusion Matrix

Deeper insights can be gained through confusion matrix analysis, as it presents a graphical way of debugging certain predictions specially if the nature of the classes is known by the analyst.

A confusion matrix is basically a squared matrix which places the predicted class on pair with the real class; where, if they are equal, places the correct predictions on the diagonal, while wrong predictions reside outside the diagonal of the matrix like the example found in Figure 8.

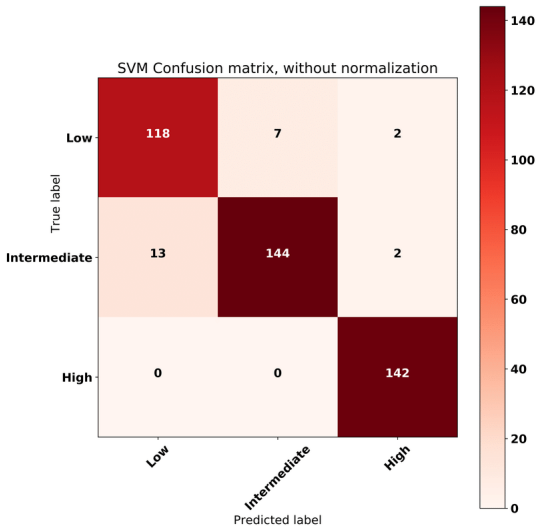

Figure 8: Example Confusion Matrix

For example, if two classes are very similar and overlapping in the mathematical space, the confusion matrix will show that the model predictions are mostly distributed amongst this two classes, as it couldn't find a suitable separation hyper-plane for this case.

So, either further steps have to be taken in the preprocessing stage to favor the successful separation, the classes could be collapsed into one for simplicity, or the model is not learning properly the patterns needed for that class and another model might be needed.

Also, if this pattern is repeated through the models, might mean that the classes are very similar and overlapping; ergo, no suitable boundary will be found to separate the classes.

This kind of insights make the confusion matrix a very useful tool to calibrate and validate the models performance in the desired task.

2 Machine Learning Model Ensembles

2.1 General Concepts

An Ensemble can be defined as a group of diverse[7] estimators which output predictions over the same input instance. This predictor aggregation often gets better prediction performance than a single predictor, as it allows to introduce additional information from different sources into the ultimate prediction, making for a more robust final model.

In classification, this predictors aggregation is usually done by establishing a voting process with each of the individual model's predictions. The voting process can be implemented in two major ways, Hard Voting and Soft Voting.

[7]As independent as possible: Different models altogether, same model on different data portions, or same model trained with the same data but different sub-sets of features.

Hard Voting is taking the predicted class from each of the estimators, where the final decision of the Classifier Ensemble is the class which the majority of the predictors voted.

Soft Voting, on the other hand, is done by averaging the class probabilities predicted by each estimator, and generating a new class probability vector using the average/median and the standard deviation as additional information about the classes. This method usually yields higher performance, as confident answers have more weight on the final decision.

Bagging: Bootstrap Aggregating

Bootstrap Aggregating is the concept belonging to the model averaging method (aggregating) but using a random sub-sample of the dataset (bootstrapping) to train each estimator. In other words, Bagging is performed by training several estimators of the same model using a random sub-set of the total data and/or features for each predictor **with**[8] replacement.

Also, this technique can be implemented by dividing the training data into a number of sub-sets, and training each estimator with a different sub-set. Models that which working principles are based on euclidean distances are the ones most benefited from this technique in contrast to tree-based models, which are mostly benefited by feature sub-sampling for model averaging.

The mathematical explanation for the variance reduction is based on the Central Limit Theorem, that states if there are n uncorrelated observations, each with variance σ^2, the sample mean variance can be noted as σ^2/n. Assuming the divisions or features of the dataset **D** of size **n** are mostly uncorrelated, and we sample sub-sets of features so that we get **D'** of size **n'** uncorrelated datasets, making it possible to train **n'** trees with each sub-set. After averaging the models, the result can be observed in Equation 11.

$$Trees_B(X) = \frac{1}{n'} \sum_{\mathrm{i}=1}^{n'} Tree_{\mathrm{i}}(X) \tag{11}$$

$$Var(Trees_B(X)) = \frac{1}{n'} Var(Tree_{\mathrm{i}}(X)) = \sigma^2/n' \tag{12}$$

Which eventually drives the variance towards 0 with **n'** -> ∞, as shown in Equation 12. Bagging is a way of simulation having this **n'** datasets from sampling with replacement of an original dataset, which approximates the theoretical behavior shown.

Finally, Bagging ensembles also can be used to estimate uncertainty without the need for Gaussian approximations, given that sampling from any-shape distributions randomly approaches a Gaussian-like distribution as the sample gets larger.

This allows for features to be sampled several times for the same predictor, usually obtaining considerably better results than the sole estimator. This is because each individual

[8] *Pasting* is the concept of training several estimators of the same model, using a random sub-set of the total features for each predictor **without** replacement.

estimator trained has a higher bias than if it was trained on the whole dataset, but aggregating the models into an ensemble helps to reduce the overall bias and variance by the estimators being less correlated.

Boosting

Boosting is another ensemble method that consists on combining weak-learners into a strong learner by training the models sequentially giving more weight the the predecessors errors. The two most popular techniques to perform Boosting are Adaptive Boosting and Gradient Boosting.

In the case of Adaptive Boosting [Freund and Schapire, 1997], it is similar to gradient descent boosting, but instead of adjusting the model's parameters aimed towards a optimizing a cost function, it sequentially adds predictors to the ensemble tweaking the sample weights in every new iteration.

In this work we will focus on Gradient Boosting [Breiman, 1997], which also is based on adding predecessor-correcting predictors sequentially to the ensemble; however, this method fits the new member of the ensemble on the residual error of its predecessor.

For general Gradient Boosting, the following process is followed in order to update the subsequent learner, and can later be modified to work with tree-based or other weak-learners depending on their working principles.

With X the features and Y the respective true class labels, Y' is the predicted class label, models $M(X, Y)$ and $H(X)$ are built as shown in Equation 13. The generalization formula of the previous model and subsequent model is as shown in 14, where alpha is the proportion of contribution of the new model H_{i} to the predecessor $M_{\mathrm{i}-1}$.

$$Model(X) = argmin[Loss(Y, Y')] \tag{13}$$

$$M_{\mathrm{i}}(X) = M_{\mathrm{i}-1}(X) + \alpha H_{\mathrm{i}}(X) \tag{14}$$

This proportion is calculated by incorporating the gradient to decide the value of α. This is done by calculating a pseudo-residual error R at each iteration i for every instance k, as shown in Equation 15, and the new model $H_{\mathrm{i}}(X_{\mathrm{i}}, R_{\mathrm{i}})$ is fitted to it.

$$R_{\mathrm{i},k} = \frac{-\delta}{\delta M(X_{\mathrm{i},k})} Loss(Y_{\mathrm{i},k}, M(X_{\mathrm{i},k})) \; with \; M(X) = M_{\mathrm{i}-1}(X) \tag{15}$$

Then, α is finally calculated as shown in Equation 16. An this process is performed iteratively until the reaching the end of the process.

$$\alpha = argmin_{\alpha}[Loss(Y_{\mathrm{i}}, M_{\mathrm{i}-1}(X) + \alpha H_{\mathrm{i}}(X))] \tag{16}$$

Stacked Generalization [Wolpert, 1992]

Stacking is a less frequent method of ensemble encountered in machine learning than the previous two cases discussed. Its basic concept is replacing the 'model averaging' stage where the predictions are combined, like Hard/Soft Voting, for a meta-learner which is fed the predictions of the previous models and trained to make a final prediction. Usually, this method is commonly used for regression tasks rather than classification tasks, given that the value outputs of the predecessor models in regression tasks is more suited for the training process of the meta-learner. Regardless, this still could be used for classification by having the meta-learner learn the weights of each contribution for optimal classification.

3 Case Study Machine Learning Models

3.1 Traditional Models

Decision Tree

The decision tree in one of the most popular and simple algorithms in supervised machine learning tasks. The core intuition behind it relies in splitting the data in sequential nodes representing each feature, by using a set of predefined rules that generate a decision based on the amount of information present in the samples provided inside each node. In layman's terms each node represents an attribute or feature, each link a decision and each leaf an outcome.

The information quality obtained from the samples in each node can be expressed a measure of purity given by the amount of mixture of classes in each node's binary classification. The most popular measures of impurity are Information Entropy or Gini Index.

'Information Entropy' can be defined as the amount of information that is required to accurately describe the observed samples, the mathematical expression is shown in Equation 17. If all samples are similar entropy will tend to 0, while a extremely heterogeneous mixture of samples will tend to an Entropy value of 1. Technically, Entropy is a measure of the uncertainty in the set of observed points $\mathbf{O}$, where $\rho_{\mathrm{i}}(C)$ is the ratio of the number of elements in a certain class $\mathbf{C}$ to the total number of elements in the dataset $\mathbf{O}$.

$$Entropy(O) = -\sum_{\mathrm{i}=1}^{k} \rho_{\mathrm{i}}(C) \cdot log(\rho_{\mathrm{i}}(C)) \tag{17}$$

From the entropy criteria, Information Gain, Equation 18, can be measured as the difference after the observations $\mathbf{O}$ are split on a specific feature $\mathbf{F}$ from a previous state, meaning how much uncertainty was reduced after splitting the observations on a certain feature.

$$InformationGain(F, O) = Entropy(O) - \sum_{c \in C} \rho_{\mathrm{i}}(c) \cdot Entropy(c) \tag{18}$$

On the other hand, Gini Index provides an insight about how good a split of the observations $\mathbf{O}$ is by measuring how heterogeneous are the classes in both sides of the split decision.

A perfect samples separation outputs a Gini Index of 0, with the worst case scenario having the same amount of samples of each class in each side of the split node, outputs a Gini Index of 1. The mathematical expression is shown in Equation 19, where $\rho_{\mathrm{i},k}$ is the proportion of class k samples among the training instances in node **i**.

$$GiniIndex_{\mathrm{i}} = 1 - \sum_{k=1}^{n} \rho_{\mathrm{i},k}^2 \tag{19}$$

Decision Tress are very powerful estimators with a high degree of interpretability and easy to use. Nevertheless, given their node split criteria, the generated decision boundaries are orthogonal on their division of data, which makes them very susceptible and sensible to small variations in data, features or orientation making the model unstable. [Geron, 2017]

Random Forest [Tin Kam Ho, 1995]

A Random Forest is a model averaging ensemble of Decision Trees usually trained by Bootstrap Aggregating. As an added feature, besides the ones inherited from Bagging and Decision Trees, Random Forest also look for the best feature while splitting a node on a random sub-set of features rather than the entirety of features available; therefore, it introduces an additional amount of randomness to the ensemble resulting on a more diverse estimator pool. This causes a higher bias but a lower variance overall, resulting in a more robust model with better performance.

Extremely Randomized Trees [Geurts et al., 2006]

Extra-Trees is also a model averaging ensemble very similar to a Random Forest, except for the fact that further extra-randomness is introduced to the model by using random thresholds for each feature split on the nodes, rather than using the optimal information-based split that Random Forest and Decision Splits use. This increases the computation speed considerably in comparison with Random Forests which look for the optimal split in every attribute. This allows to further decrease the variance present in the data by trading off a small increase of bias in return.

3.2 State of the Art Models

Sophisticated Gradient Boosting models have recently achieved state-of-the-art performance in several vectorized data machine learning tasks and competitions, becoming a direct competitor to the Deep Neural Networks. This models hold mostly the same core principles of Gradient Boosting, but are permeated by several layers of computational and mathematical methods that allow to achieve optimal learning capabilities while increasing the computational efficiency of the models.

The two models used in this work are amongst the most powerful and utilized tree-based models to achieve state-of-the-art performance in many tasks up to date. This models are Extreme Gradient Boosting and Light Gradient Boosting.

Extreme Gradient Boosting [Chen and Guestrin, 2016a]

XGBoost uses a histogram-based algorithm that groups the feature values into discrete bins further increasing training speeds and adding an additional amount of variance that acts as a regularizing factor to contain over-fitting. This technique additionally decreases the memory usage of the model, generating lighter models that require less computational resources. This generates a trade-off between speed and accuracy, as more bins are more accurate but also slower to compute.

This histogram-based split criteria also has the advantage of providing a way of dealing with data sparsity or missing values, where upon the binning of the continuous values, empty fields can be set automatically to 0;ego, handling the missing data without crashing the algorithm.

The inherent mathematical modeling of XGBoost's nature can be portrayed as a regularized version of Gradient Boosted Machines with a level-wise tree growth as shown in figure 9.

Figure 9: XGBoost Level-Wise Growth Mechanism

This means that the initial approach has built-in L1 and L2 regularization in its mathematical formula as observed in Equation 23.

Following the mathematical intuition presented in Equations 13 through 16, M(X) and H(X) are Classification And Regression decision Tree (CART) models, therefore the expression is rewritten as shown in Equation 20, where $H_{\mathrm{i}}(X)$ is a tree of L leaves.

$$H_{\mathrm{i}}(x) = \sum_{l=1}^{L} b_{l\mathrm{i}} I_{Rl\mathrm{i}}(x) \tag{20}$$

Introducing this equation into Equation 14, the expression obtained is the one presented in Equation 21, which in turn also derives into Equation 22 for the calculation of α_li for a regular Gradient Boosted Machine.

$$M_{\mathrm{i}}(x) = M_{\mathrm{i}-1}(x) + \sum_{l=1}^{L} \alpha_{l\mathrm{i}} I_{Rl\mathrm{i}}(x) \tag{21}$$

$$\alpha_{l\mathrm{i}} = argmin_{\alpha} \sum_{x_{\mathrm{i}} \in Rl\mathrm{i}} Loss(Y_{\mathrm{i}}, M_{\mathrm{i}-1}(x_{\mathrm{i}}) + \alpha) \tag{22}$$

But, the loss function of XGBoost has an additional term Ω concerning the regularization.

In other words, the equation presented in Equation 23 is a reformulation of the GBM loss, that calculates additionally the maximum gain and is used in the tree building process while each node is split, turning the two-step process of GBM's into a single step, which algorithmic solution method is adapted depending on the loss function selected in combination with Taylor Series Expansion for computation.

$$Loss(Y, M(X)_{\mathrm{i}}) = \sum_{\mathrm{i}=1}^{n} Loss(Y_{\mathrm{i}}, M_{\mathrm{i}-1}(x_{\mathrm{i}}) + H_{\mathrm{i}}(x_{\mathrm{i}})) + \sum_{l=1}^{L} \Omega \cdot H_{\mathrm{i}}(x_{\mathrm{i}}) \tag{23}$$

This allows for a more robust model to overfitting, with a substantial increase in accuracy and computational efficiency.

In addition to the mathematical advances regarding regular GBM's, XGBoost has been under community development for a while, generating the opportunity to implement several contributions that have optimized the algorithm's computational efficiency, such as parallel processing by multiple CPU cores, built-in cross validation on each iteration defining automatically the optimal number of iterations on a single run, and a built-in greedy effective tree pruning removing the branches when negative loss is encountered in the node split not contributing to a positive information gain.

Light Gradient Boosting [Ke et al., 2017b]

LGBM is a tree-based Gradient Boosted framework which aims primarily for computational speed and high performance. It achieves this by implementing distributed computing capabilities and complexity reducing techniques.

This models differentiates itself from other tree-based models by splitting the tree leaf-wise rather than level-wise for its growth process, allowing for the algorithm to reduce more loss in contrast with level-wise algorithms, as shown in Figure 10 [9].

This results in an increase of the model's accuracy without an increase in the model's overall size, achieving even sometimes better performance than more complex models in a fraction of the time. Nevertheless, this tree growth method also produces more complex trees that may lead to overfitting the data if a suitable maximum depth is not selected for the training stage.

[9] Figures 9 and 10 Images Source: https://towardsdatascience.com/lightgbm-vs-xgboost-which-algorithm-win-the-race-1ff7dd4917d

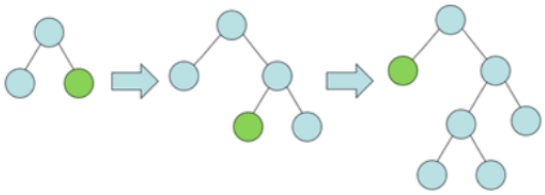

Figure 10: LGBM Leaf-Wise Growth Mechanism

In addition to the histogram-based optimal splitting of XGBoost, LGBM also uses a method that generates a 10x to 100x computation speed increase versus XGBoost, which is a novel technique called Gradient-based One-Side Sampling (GOSS). This algorithm consists in down-sampling the amount of instances computed based on each calculated gradient.

The principle is that instances with a small training error will present small gradients, as the variation between subsequent models will not be important, and can be considered stable and well trained, On the other hand instances with large calculated error are not stable yet; and therefore, under-fitted.

In order to speed up the training computation times, small error instances can be completely frozen or sub-sampled for subsequent calculations, retaining only the instances with large gradients for posterior computation. GOSS algorithm randomly sub-samples a fraction of the instances with smaller weights while retaining the instances that have larger gradients, allowing to save computational resources and increase computational speed dramatically.

The theoretical analysis of the GOSS algorithm applied to Decision-Tree Gradient Boosting, can be noted mathematically as the author [Ke et al., 2017b] explains on section 3.2 of the original paper 'LightGBM: A Highly Efficient Gradient Boosting Decision Tree'.

Definition of information gain on Tree-Based Gradient Boosting by 'amount of variance after splitting': *Let O be the training dataset on a fixed node of the decision tree. The variance gain of splitting a feature j at point d for this node can be defined as shown in Equation 24, where $n_o = \sum I[x_i \in O : x_{ij} \leqslant d]$ and $n^j_{r|O}(d) = x_i \in O : x_{ij} > d$.*

$$V_{j|O}(d) = \frac{1}{n_O}\left(\frac{(\sum_{\{x_i \in O : x_{ij} \leq d\}} g_i)^2}{n^j_{l|O}(d)} + \frac{(\sum_{\{x_i \in O : x_{ij} > d\}} g_i)^2}{n^j_{r|O}(d)}\right) \tag{24}$$

The optimal information gain is then calculated by selecting a $d^*_j = argmax_d[V_j(d)]$ by the decision tree algorithm, and then computing the largest information gain given by $V_j(d^*)$, effectively splitting the attribute j' at point d^* into the next two child-nodes of the decision tree.

The GOSS method then scores and sorts the training instances according to the absolute value of each of their gradients, from larger to smaller. A sampling parameter $a \in [0, 1)$ is introduced so that the top percentage of $a * 100\%$ instances with the largest gradients is kept, getting a sub-set noted A for large gradients and A^C for small gradients. Subsequently, another step randomly samples b samples out of A^c generating a new sub-set $B = bx|A^C|$ of

size b. From the set $C = A \cup B$ the optimal split of this sub-set of instances is calculated applying a reformulated version of the variance gain shown in Equation 24, resulting in Equation 25, where $\frac{1-a}{b}$ is a normalization coefficient over the sum of gradients over subset B to bring weight them the same as the size of subset A^c, and for notation simplicity $'Subset'_l = \{x_{\mathrm{i}} \in' Subset' : x_{\mathrm{i}j} \leq \mathrm{d}\}$ and $'Subset'_r = \{x_{\mathrm{i}} \in' Subset' : x_{\mathrm{i}j} > \mathrm{d}\}$.

$$\bar{V}_j(\mathrm{d}) = \frac{1}{n}\left(\frac{(\sum_{\{x_{\mathrm{i}} \in A_l} g_{\mathrm{i}} + \frac{1-a}{b}\sum_{\{x_{\mathrm{i}} \in B_l\}})^2}{n_l^j(\mathrm{d})} + \frac{(\sum_{\{x_{\mathrm{i}} \in A_r\}} g_{\mathrm{i}} + \frac{1-a}{b}\sum_{\{x_{\mathrm{i}} \in B_r\}})^2}{n_r^j(\mathrm{d})}\right) \tag{25}$$

Ergo, with this reduction of partially redundant computations to determine the optimal node split point over a reduced sub-set, without loss of generality by focusing primarily on the larger gradient instances, the calculation of $\bar{V}_j(\mathrm{d})$ results in a much more computationally efficient approach without significant loss of performance in comparison with the full calculation of $V_j(\mathrm{d})$.

4 Deep Learning

4.1 What is Deep Learning?

Deep Learning is a particular sub-field of Machine Learning that relies mainly on the use of Artificial Neural Networks as a learning model.

The Neural Network name is introduced from the conceptual ambition of modelling the human brain through a mathematical approach[Rosenblatt, 1957]. Each node in the model denominated a neuron is a feed-forward[10] mathematical model that takes a set of vectorized inputs x_{i}, multiplies them by a continuous-valued scalar weight w_{i} that symbolizes the synapse strength of the connection communication from the previous neuron and adds a bias b_{i}. Then, it sums up the weighted sum $w_{\mathrm{i}}x_{\mathrm{i}}$ which then passes through a non-linear **activation function** that squashes the value, in the original proposition a sigmoid function, outputting a response value to the next layer. Single neuron in Fig 11 [11].

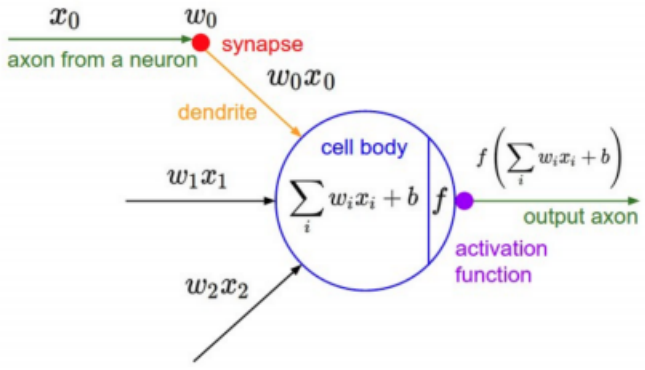

Fig. 11: Biologically Inspired Neuron - 'The Perceptron'

[10] a-cyclical graph where no feedback connections in which outputs connections are fed to itself
[11] Image Source: Karpathy, 2016 http://cs231n.github.io/neural-networks-1/

An Artificial Neural Network is a stacked connection of highly interconnected processing elements that may add up to hundreds of thousands units in hierarchical layers. Given the flexibility of each node to learn a parameter, also commonly called weight, makes them well-suited for scalable applications with large volumes of data.

Also, because of the previous point, ANN's are extremely powerful and scalable estimators that are able to process multidimensional vector or matrix data, and in contrast to most their Machine learning counterparts are able to map non-linear patterns present in data, that are far too complex to be taught to the machine explicitly by a human.

An Artificial Deep Neural Network has to be composed of at least an input layer, a hidden layer and an output layer. Presented in figure 12[12] is a basic example of the architecture with each node corresponding to a perceptron, also called a Multi-Layer Perceptron Neural Network.

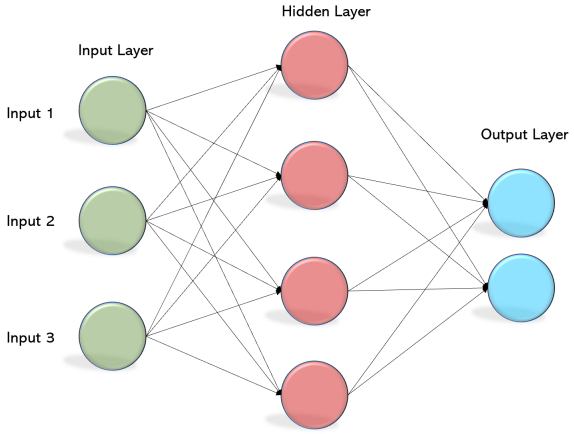

Figure 12: Basic Neural Network Example - Multi-Layer Perceptron

Given the exponential growth of the computational power of parallel processing units such as Graphical Processing Units (GPU's) and Tensor Processing Units (TPU's) since the early 2000's, the field has experienced an accelerated growth, given that now that very large models can be trained, for very complex tasks, with huge amounts of data with extreme efficiency.

With the explosion of research in the area, the neurons on each individual layer can be composed of several different mathematical modelling methods depending on the tasks desired to tackle, where extra features have been implemented to get more sophisticated task-specific units/layers.

Activation Functions

An activation function is a non-linear transformation f performed over the output of each layer, so that $y = f(z)$ with $z = \sum_{\mathrm{i}} x_{\mathrm{i}} \cdot w_{\mathrm{i}} + b$, allowing to control the output's magnitude

[12]Image Source: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

and sign by setting normalizing thresholds increasing the model's learning potential during training.

Some of the most popular and important activation functions are Sigmoid (Equation 26), Hyperbolic Tangent (Equation 30), ReLU (Equation 28), LeakyReLU (Equation 29) and Softmax (Equation **??**), amongst many others.

Sigmoid function was originally proposed on the perceptron inspired on the neuronal response of the biological nervous system, which tends to saturate at very high impulses and vanish at very low impulses. In computational uses, the sigmoid function had two major drawbacks, the fact that the data loses the sign, and that in very deep neural networks the gradient eventually vanish in the layers that are further away from the output, hindering the learning process of deeper layers.

$$y = f(z) = \sigma(z) = \frac{1}{1 + exp(-z)} \ \in \ (0, 1) \tag{26}$$

Ergo, as a solution for the first issue, hyperbolic tangent is a good alternative to be used without data losing the sign, and is still widely used for this reason in the initial layers of many Deep Learning models. But, for large values of z it still saturates proving detrimental for learning in deeper layers.

$$y = f(z) = tanh(z) = \frac{exp(z) - exp(z)}{exp(z) + exp(z)} \ \in \ (-1, 1) \tag{27}$$

Rectified Linear Unit was introduced experimentally detaching from the biological analogy, and simply restricts the output z to positive values, conserving the magnitude of the output. It is widely used and regarded as one of the activation functions that provide the most stable good results to this day regardless of the model architecture.

$$y = f(z) = \begin{cases} 0, & \text{for } z < 0 \\ z, & \text{for } z \geq 0 \end{cases} \ \in \ [0, \infty) \tag{28}$$

In order to keep ReLU advantages, Leaky ReLU was proposed to capture values which sign was on the negative domain, where as seen in the equation below, has a slow leakage towards negative values that adds a more information to the learning process.

$$y = f(z) = \begin{cases} \alpha \cdot (exp(-z) - 1), & \text{for } z < 0 \\ z, & \text{for } z \geq 0 \end{cases} \ \in \ (-\alpha, \infty) \tag{29}$$

And finally, the Softmax function that is used to output a probability for each unit in the last layer of a Deep Neural Network model, mainly used in classification tasks. In the great majority of cases, the amount of output units of the DNN is the same as the number of classes in the classification task.

$$y = f(z) = Softmax(z) = \frac{exp(z_{\mathrm{i}})}{\sum_{\mathrm{i}=1}^{units} exp(z_{\mathrm{i}})} \ \in \ (-1, 1) \tag{30}$$

Training: Loss, Optimization and Back-propagation

Taking as a general example the mathematical structure shown in Equation 31 of a sequential Artificial Neural Network introduced previously on Figure 12, where k is the number of layers, U_k are the units in layer k, b is the bias, x is the input data and y is the layer output.

$$y_j^k = \begin{cases} f\left(\sum_{\mathrm{i}=1}^{U_k} w_{\mathrm{i}j} \cdot x_{\mathrm{i}} + b_{\mathrm{i}}\right), & \text{for layer } k = 1 \\ f\left(\sum_{\mathrm{i}=1}^{U_k} w_{\mathrm{i}j} \cdot y_{\mathrm{i}}^{k-1} + b_{\mathrm{i}}\right), & \text{for } k > 1 \end{cases} \tag{31}$$

The process to train each of the parameters w efficiently to converge to a solution is very straightforward; the network is inputted a training sample or a training-batch of samples forcing it to make predictions, then the results are compared with the target output and the difference is computed. Finally, the parameter w contributions are weighted by propagating the gradient of the error back through the network adjusting the parameters with larger gradients.

Loss or Target Function

The loss function, also called target function, is the responsible for measuring the similarity of the predicted result to the actual result. The function selected can vary greatly depending the type of task aiming to be solved.

In most cases it follows the same basic concept: compare network outputs to the target values. The most basic loss function used widely on Machine Learning, nevertheless powerful, is the mean squared error, which provides a metric between the predicted value from the coefficients.

Minimizing this target allows to fin the optimal parameters θ to fit the available data.

$$argmin(Loss) = argmin(MSE) = \nabla J = argmin((1/n) \sum_{\mathrm{i}=1}^{n} (Y_{\mathrm{i}} - \hat{Y}_{\mathrm{i}})^2) \tag{32}$$

On the other hand, the equation widely used for classification tasks is Cross Entropy which receives as an input the predicted class probability y' from the Softmax function, shown in Equation 33. Where δ is a Dirac which is 1 if the class c of observation i is the real class, or else 0.

$$argmin(Loss) = argmin(CE) = \nabla J = argmin(-\sum_{c=1}^{C} \delta_{\mathrm{i},c} \cdot log(p_{\mathrm{i},c}(y'))) \tag{33}$$

The problem with optimizing this functions the classical manner, is that it is not possible just by differentiating and equaling to 0 due to the non-linear dependency of all the parameters w on the neural network , which generates the existence of a wide array of local minima and saddle points.

Optimization

Gradient descent is a widely used algorithm to try and tackle this problem on machine learning algorithms. The basic workings of it is the following: Starts from initialization points either known or random, and then performs a sequential approximation to the true parameters by iterating over the gradient of the weights, following the slope that maximises the descent.

In particular, gradient descent can be expressed in the following general formula, where, in the case of neural networks η is the step size and the neural network weights w are the parameters θ:

$$\theta^{(j+1)} = \theta^{(j)} + \eta \nabla J(\theta^{(j)}) \tag{34}$$

The specific challenge risen from neural networks with this method, is that ANN's are analog to a collection of many linear regression machine learning algorithms, and this algorithm calculates one gradient for each model for each data point; therefore, this algorithm is inefficient to calculate a great number of parameters.

A modification that allows batch size calculations by storing in a variable the cumulative sum of the error on the batch predictions ε and starting from a random initialization in each step is called Stochastic Gradient Descent (SGD). This algorithm is important because it allows to train machine learning algorithms on an efficient manner, and avoid some of the complications risen from the non linear dependency of the θ parameters, as not all points will have the same local minima or saddle points.

In particular, for neural networks further mathematical adaptations have to be performed to train a neural network successfully, which were introduced in the concept of Back-Propagation.

Back-propagation

Once there has been a training instance inputted, the parameters of the network have to be updated. In order to do this in an efficient manner the concept of Back-propagation was introduced. [Rumelhart et al., 1986]

When the network has evaluated and predicted an input vector, a loss metric is generated from the target function. In order to propagate the SGD and update the values of each parameter individually, $\mathrm{J}(\theta_{\mathrm{i}})$ must be computed. By applying a trick using partial derivatives.

$$\frac{\delta J_n}{\delta w_{j\mathrm{i}}} = \frac{\delta J_n}{\delta \hat{z}_j} \cdot \frac{\delta \hat{z}_j}{\delta w_{j\mathrm{i}}} \tag{35}$$

Where defining $\delta J_n/\delta \hat{z}_j$ as the error ε_j, and thanks to the iterative concatenation of layers shown in Equation 31 the term $\delta \hat{z}_j/\delta w_{j\mathrm{i}}$ corresponds to the previous layer neuron z_{i}.

Ergo, this is analogue to multiply the output error for the input connection, obtaining an error δ_j value for each neuron on the network, from Equation 35 in component form [Nielsen, 2015]:

$$\delta_{\mathrm{i}}^{(l)} = \sum_k w_{kj}^{(l+1)} \delta_k^{(l+1)} \sigma(z_j^l) \tag{36}$$

Regularization Methods

Dropout [Srivastava et al., 2014]

Dropout is a neural network regularization technique, which allows to train deep neural networks with overall better generalization capabilities upon unseen predictions. Besides, it provides a boost in training speed by decreasing the amount of parameters being trained on a given epoch (iteration).

Dropout consists on dropping randomly a certain amount of connections between two layers on each training iteration with a user-defined probability reigned by a Bernoulli distribution, and, when applied, forces the network to learn redundant representations instead of relying on only some units to make certain predictions.

Overall, the result of applying this regularization method is analogue to train several neural network ensembles of different architectures, and therefore learning different representations, to then perform a model average over the final ensemble, that ultimately decreases the variance and allows for better prediction generalization based on a greater number of units contributing to the final result.

Batch Normalization [Ioffe and Szegedy, 2015]

Batch Normalization is a technique implemented between layers, based on shifting and scaling the activation's mean and variance on the hidden layers, getting an improvement in training efficiency. The objective is to reduce the co-variance shift of the weights produced by different magnitudes of the previous layer outputs, this in turn produces a pseudo-independence amongst layers that is beneficial for the trained model's generalization capabilities.

Also, this method ensures that no activation escapes to too high or too low values while adding some noise to the information flow; therefore having a regularizing effect that aids to the stability of the training process to avoid over-fitting. This is done by normalizing the output of the previous layer by subtracting the mini-batch mean and dividing by the standard deviation of the mini-batch.

To avoid network destabilization, the normalized output is multiplied by two learnable parameters γ and β, which act as a standard deviation and mean respectively. This allows for the back-propagation stochastic gradient descent algorithm to preform a de-normalization

of the layer in order to propagate the gradients successfully on its way.

4.2 Case Study Deep Learning Models

Multi-Layer Perceptron

The Multi-Layer Perceptron Neural Network is a very powerful model, that can be explained as a fully connected graph where each unit is connected to all the previous layer units and outputs their response to each of the sequential layer neurons. The hidden layer/s are responsible for the feature extraction from the training instances, while the last layer is configured for classification or regression tasks depending on the output form.

The links between neurons are defined by the linear combination of each of the single perceptrons. Having multiple inputs in a 3 layer MLP, where f,g,h are activation functions, x_{i} is the input data, b_{i} is the bias, and m,n,l are the number of neurons in each respective layer; then output z_{i} is modeled as the equation shown in Equation 31.

Here the clear iterative hierarchical nature of each layer with the previous is presented, and also shows that there is not a mathematical stack limit for the number of layers to be introduced, but performance-wise and computationally-wise there is a limit depending on the task and dataset. The main drawback of the fully-connected layers is that input data has to be vectorized in one dimension, also called flattened, to be processed by the input layer units.

The number of layers and units per layer define the parameter size of the model, and are strictly up to the user's decision, but, the larger is the model in number of parameters, the probability of over-fitting the data also increases.

4.3 Convolutional Neural Networks [LeCun et al., 1999]

Convolutional Neural Networks are a special type of layer aimed to solve image recognition problems by processing multidimensional matrix data, also called tensors, without performing a flattening operation to vectorize it, therefore, maintaining spatial relationships between objects.

In order to retain the spatial relationships in the tensor, Convolutional Networks are not fully connected to the next layer, but the outputs, called filters or kernels, are a stack of feature maps of a specific regions of the tensor extracted by the convolution layer, formula shown in Equation 37. Then, this feature maps are flattened and passed to a Multi-Layer Perceptron for classification or regression.

$$FM(\mathrm{i}, j) = \sum_{h=1}^{height} \sum_{w=1}^{width} 2DTensor(\mathrm{i} - h, j - w) Kernel(h, w) \tag{37}$$

This working principles of the filters being applied by sliding over a 2-dimensional tensor and its channels, make CNN layers a very powerful pattern extraction tool with lower complexity than the MLP, that can explore locally and extend the amount of information

locally in order to be interpreted generally by a fully connected layer. A general example is presented on Figure 13 [Coşkun et al., 2017].

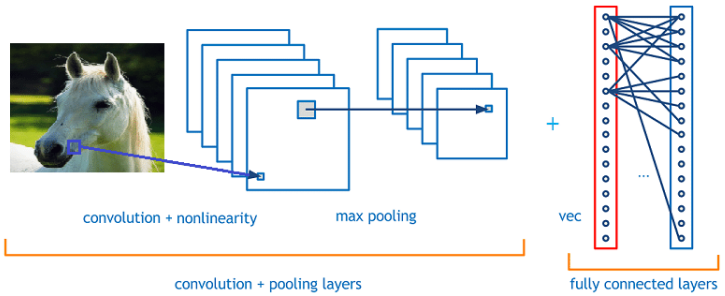

Figure 13: Basic Convolutional Neural Network Example

This technique has then been extrapolated to many areas such as signal processing, natural language processing, time series processing, etc. Basically, anywhere were local patterns contain a high degree of information that can be expressed as a tensor, and therefore subjected to patterns that are sometimes not clear for explicit extraction by humans.

The model used in this work is a 1-dimensional convolution, which is follows the same principle, with the difference that the sliding window acts over a vector of features capturing the local relationships, extracting the features, and then subsequently handing them into a MLP for classification.

5 Bayesian Modelling [Bayes, 1763]

5.1 Bayes Theorem

Bayes theorem,or 'Bayes Rule' shown in Equation 38, is a way of calculating conditional probability by exploiting the relation between a conditional probability and its reverse form. It was originally proposed by Father Thomas Bayes, and published posthumously by one of his students, as an algorithmic method to use available evidence to calculate an unknown parameter, by expressing/updating probability as a degree of belief in a proposition before and after accounting for the evidence.

$$P(B|A) = \frac{P(A|B) \cdot P(A)}{P(B)} \quad -> \quad Posterior = \frac{Likelihood \cdot Prior}{Evidence} \tag{38}$$

5.2 Bayesian Statistics

'Bayesian Statistics is a mathematical procedure that applies probabilities to statistical problems. It provides the tools to update the beliefs in the evidence of new data.' [13]

[13]Quote Source: https://www.analyticsvidhya.com/blog/2016/06/bayesian-statistics-beginners-simple-english/

This is the core of Bayesian Modelling, which applies Bayesian theory to solve statistical problems relying on the probabilistic modelling approach to develop tools that allow for approximate inference of a posterior belief. This essential specially on Machine Learning, as algorithms that learn from data are direct candidates for Bayesian Statistics to be applied in them, presenting several advantages over deterministic models as they mostly are.

The main ideas underlying the probabilistic modelling method of Bayesian theory is to find the parameters θ that are likely to have generated the outputs of a function $y = f(\theta, x)$, where the training input data is characterized by a vector of features x and the respective outputs as a vector y.

For this, using the Bayesian approximation approach, a *prior* distribution is chosen over the parameter space $p(\theta)$, where this *prior* represent the belief of the parameters θ that might have generated the data before observing any data samples.

Then, upon the observation of instances, the *prior* distribution will transform to represent the likelihood of each parameter given the observed training instances. This likelihood can be represented as a probability, following the conditional probability $P(A|B)$ shown above in Bayes Rule, in function of the parameters θ, data points x and known response y as $p(y|x, \theta)$.

This likelihood function can vary depending on the task or modelling, but for classification tasks a Softmax likelihood is commonly used, and is formulated as shown in Equation 39. While, for regression tasks, the likelihood is commonly formulated using a Gaussian likelihood as shown in Equation 40, where τ^{-1} is inverse of the model precision, that can be characterized as the observation noise variance .

$$P(y^*|x, \theta) = \frac{exp(f^{\theta}_{y^*}(x))}{\sum_{y^*} exp(f^{\theta}_{y'}(x))} \tag{39}$$

$$P(y^*|x, \theta) = (y^*; f^{\theta}(x), \tau^{-1}I) \tag{40}$$

Applying this expressions in the Bayes Rule equation, presented in Equation 41, with data x and the corresponding response y, the *posterior* can be updated based on the likelihood of the observations and their response, returning the most probable parameters given the observations x shown to the model.

$$P(\theta|x, y) = \frac{p(y|x, \theta)\ p(\theta)}{p(y|x)} \tag{41}$$

5.3 Bayesian Inference

With the new learnt distribution of parameters from observations, the output response for new observations x^* can be predicted by the process of Bayesian Inference; which refers to the process of integrating over the model's parameters to get an approximate **inference** of the response. The response prediction y^* for x^* observations is performed by solving Equation 42.

$$P(Y*|x*,x,y) = \int p(y^*|x^*,\theta)p(\theta|x,y)\delta\theta \tag{42}$$

In order to fully evaluate the *posterior*, the normalizing term known as *evidence* must be calculated as shown in Equation 43.

$$P(y,x) = \int p(y|x,\theta)p(\theta)\delta\theta \tag{43}$$

The ideal situation is to be able to marginalize over the entire parameter search space of known and unknown values for *theta*, each weighted by its own likelihood. Here, a challenge arises, as marginalizing the likelihood over θ is often times too complex to calculate analytically and is not computationally tractable, with the exception of some very simple models where the *evidence* can be calculated.

Ergo, given this complexity to calculate analytically and that the model *likelihood* is conjugate to the *prior*, approximation methods can be used to model the posterior without the need of direct calculation of the integral. Two of the most frequent methods that yield good results are Monte Carlo Approximation and Variational Inference.

Monte Carlo Methods [Metropolis and Ulam, 1949]

Monte Carlo methods are used frequently in modern statistics, the general intuition is based on drawing random samples from $p(\theta)$ non-trivial probability densities, which eventually help integration methods portray a reliable estimation of the *posterior* distribution after a large number of samples.

The basic concept is related to Equation 45, and is to draw randomly $\mathrm{d_i} \in \mathrm{d}$ samples from $\pi(\theta)$ when only a sample from $q(\theta)$, an auxiliary/proposal density, is available. Given enough random samples of size d, the integral can be approximated reliably, but as more definition is required the computational cost also rises.

$$E_\pi[h(\theta)] = \int h(\theta)\pi(\theta)\delta\theta \tag{44}$$

Several methods have been built on this principle over the years, developing refined algorithms mainly to concentrate the sampling in relevant areas of interest to obtain more accurate estimates in less time and using less computational resources such as Important Sampling [Geweke, 1989], Rejection Sampling [Gilks and Wild, 1992], SIR [Smith and Gelfand, 1992], Markov chain-based Metropolis-Hastings algorithm [Hastings, 1970], Gibbs Sampling [Gelfand and Smith, 1990], etc.

This methods are very useful and widely used to this day in small-to-medium data statistic modelling, but with larger dimensionality and data volumes, their performance is hindered by the requirement of massive computational resources.

Variational Inference [Jordan et al., 1999]

Variational Inference is a different approach to the intractability of the posterior distribution by replacing the marginalization of the likelihood for a optimization step over distributions. This approach does not provide as accurate results as Monte Carlo methods, but it scales better and is more efficient in terms of computational resources, while retaining the advantages of Bayesian modelling and results that capture model uncertainty.

The core principle of this approach is to define an approximate parametric 'variational distribution' parametrized by θ so $Q_\theta(\theta)$, which structure and nature is well-known and easy to evaluate. Then, using a measure of similarity between the variational distribution and the true posterior distribution $p(\theta|x, y)$, the proposed distribution is approximated as close as possible to the true posterior.

To achieve this, the Kullback-Leibler Divergence is minimised with respect to θ providing of a measure of similarity in information bits[14] between the true distribution and the proposed variational distribution.

Also, the similarity between P to Q is not always tractable; therefore, a simple reformulation trick is performed which consists on instead of measuring similarity from P to Q, the reverse is taken and the proposal Q goes to P to make the problem tractable as shown in 45.

$$KL[Q_\theta(\theta) \mid\mid p(\theta \mid x, y)] = \int Q_\theta(\theta) log \left(\frac{Q_\theta(\theta)}{p(\theta \mid x, y)} \right) \delta\theta \tag{45}$$

This makes the KL minimisation equivalent to the maximisation of the Evidence Lower Bound (ELBO) with respect to the variational parameters that define the proposed parametric variational distribution shown in Equation 46, where $log\,(p(y, x))$ is the logarithm of the evidence.

$$\mathcal{L}_{VI}(\theta) = \int Q_\theta(\theta) log\,[p(y|x, \theta)\delta\theta - KL(Q_\theta(\theta) \mid\mid p(\theta))] \leq log\,(p(y, x)) \tag{46}$$

The procedure is to maximise the first term of Equation 46 which is the Expected Log Likelihood, and minimize the difference between the true distribution and the proposed distribution often regarded as the *variational prior*, capturing the $Q_\theta(\theta)$ that that is as close as possible to the *variational prior*, penalizing complex distributions.

This procedure turns the problem tractable for a large array of models where MC methods are not particularly suited, but requires further tweaks in order to work in large-data settings, as it is still computationally expensive. Also, in models that are significantly more complex, there are challenges on calculating the last integral making the problem computationally intractable without further modifications to the method that are respective to the model utilised.

[14] It is not an euclidean distance, therefore is not bidirectional

5.4 Uncertainty is Key

The advantage of the probabilistic modelling, and ergo Bayesian modelling, is that it allows to determine the likelihood of certain responses of the system which are not known certainly from the observations provided in the data due to inherent variance present in the data.

Specially in engineering and other applied sciences, knowing the confidence intervals of a certain output response is necessary for reliable estimations regarding the variability of the phenomena under perturbations that happen upon predicting unseen data.

The deployment of models under real-world scenarios that are properly calibrated and provide reliable estimates for the uncertainty of a model is key for almost every precision industry in modern times.

This is why the probabilistic modelling approach is so important and necessary to be applied in scalable and reliable solutions in the Machine/Deep Learning fields, which work mostly with deterministic models that even though have statistical properties, have not adopted proper uncertainty quantification practices into the deployed models.

6 Bayesian Theory in Deep Learning

In the last subsection 5.4 'Uncertainty is Key' it was stated why uncertainty is important for any mathematical model, specially the data-learning models, such as Neural Networks, that are currently being applied in a wide arrange of fields that include sensitive and critical decisions, and, as for now, are mostly deterministic rather than probabilistic.

This generates problems concerning prediction over-confidence which is completely undesirable in risk-subjected decisions, given that there is already little intepretability of the model's decision making process. How uncertain a model is of its predictions is extremely important for sensitive applications.

6.1 Need for Uncertainties in Deep Learning

As Deep Learning Field grows in research and applications, such as self-driving cars or industrial automation, tackling larger and more complex tasks, the need for uncertainty in the deterministic models has awakened a resurgence of this concerns and methods of how to possibly solve the problem of 'knowing what the model doesn't know'.

Yarin Gal postulated Monte Carlo Dropout [Gal and Ghahramani, 2015] as a way of treating Neural Networks as a model averaging ensemble of many networks, performing predictions with the model using dropout to get a confidence interval on the predictions. Even though this approach is not completely Bayesian. Nevertheless, it extrapolates a lot of concepts from Bayesian Theory and gets approximate results fast with only one additional step over regular deterministic neural networks that does not change any of the current training infrastructure.

On the other hand, there has been a renewed interest on Variational Inference as a way of obtaining fully Bayesian Neural Networks, which were introduced in the 1990's when the first

concepts for Bayesian Deep Learning appeared. Charles Blundell in 2015 introduced in his paper 'Weight Uncertainty in Neural Networks' [Blundell et al., 2015] an effectively trainable Bayesian Neural Networks by using Variational Inference, which consisted on replacing the deterministic weight parameters of neural networks, by parametric distributions, such as Gaussian, which would limit the learning of parameters to the ones needed to successfully generate the target distribution shape for each node.

6.2 Bayesian Neural Networks with Variational Inference

Despite Regular Neural Networks being trained almost exclusively by back-propagation, this method has its downsides. Given the large amount of parameters and data volumes, finding the optimal values for the hyper-parameters by tuning is extremely difficult, as the complexity is proportional to the size of the problem, and additionally, a non-convex optimization problem. For regular neural networks, only single-point-estimate parameters are learned in the network, making it very susceptible to under/over-fitting, which in turn generates over-confident predictions that do not account for uncertainty in the model parameters.

Charles Blundell approach, popularly called 'Bayes-by-Backprop', allows to use most of the current training algorithms and infrastructure that regular neural networks use without change. It is basically a Variational Inference method that aims to learn the posterior distribution of a set of neural network weights $w = q_\theta(\theta|Data)$ that, in order to make the multiplications and differentiation during back-propagation, samples a single instance from each weight and tweaks it according to a loss function based in Variational Free Energy presented in Equation 46.

Hence, the optimal parameters from a proposed variational distribution $q_\theta(\theta|Data)$ and the true distribution $p(\theta|Data)$ can be approximated by KL divergence. Gradient Descent can be applied to a training phase and slowly modify the parameters of the proposal distribution, based on a loss function using the Kullback-Liebler Divergence which measures a unit of similarity (bits not distance) between two distributions, approximating the proposal over the true posterior distribution.

Trick: Instead of going from P to Q, the reverse is taken and the proposal Q goes to P to make the problem tractable.

$$KL[Q_\theta(w \mid Data) \mid\mid P(w \mid Data)] = \int Q_\theta(w \mid Data) log[\frac{Q_\theta(w \mid Data)}{P(w \mid Data)}]\mathrm{d}w \tag{47}$$

$$\theta_{opt} = arg_\theta min \sum_{\mathrm{i}=1}^{n} log[Q_\theta(w^{(\mathrm{i})} \mid Data)] - log[P(w^{(\mathrm{i})})] - log[P(Data \mid w^{(\mathrm{i})})] \tag{48}$$

Inference & Probabilistic Output

The main idea of Bayesian Neural Networks is that each prediction will be a collection of the parametric distributions $w_\mathrm{i} = q_\theta(\theta|x_\mathrm{i})$ that replace the scalar weights. Upon inference, the network will be fed an input instance to be predicted N times, where each time one Monte Carlo sample will be drawn of each learnt distributions $q_\theta(\theta|X)$, which produces a

collection of outputs. If the network is certain about its output, an therefore the distributions that contribute the most are consistent on the sampling process, the sample will show a median prediction probability for each class and a standard deviation of said prediction. This standard deviation is the confidence of the model in the answer given, and the median is simply the most frequent prediction value.

Both of this metrics can be successfully exploited to set minimum criteria for prediction reliability of the model.

6.3 Bayesian Neural Network Back-propagation

The Local Re-parametrization Trick

A reparametrization is basically a mathematical way to reformulate statistical problems into an equivalent form. The local reparametrization trick is when the global uncertainty of the parameters is formulated as a form of local uncertainty that is independent across examples.

The source of global noise is transferred to a local expression so that $\varepsilon -> f(\varepsilon)$. This is done by proposing an alternative estimator such that $Cov[L_{\mathrm{i}}, L_j] = 0$ so that the variance of the stochastic gradients scales proportionally as M^{-1}, then by sampling the intermediate variables $f(\varepsilon)$ and not ε the problem is made computationally efficient in comparison with the global estimation.

Reparametrization in Back-Propagation

Given than neural networks would have to sample posteriors from thousands of weight distributions and propagate those parameter updates, the Monte Carlo sampling methods, although used for some reparametrization tricks, are not efficient for the training process.

The idea behind Bayesian Neural Networks is that instead of having scalars as weights w, it has two parameters of μ and σ for the mean and variance of a Gaussian distribution that define a parametric distribution. Then, the overall contribution of the trained parametric distributions model a posterior 'knowledge' over the inferred prediction, and allow to get uncertainty estimates on predictions from the neural networks that may be interpreted also as 'risk of being wrong'.

Variational Inference has been proven to be able to scale to train a ANN's [Blundell et al., 2015] using several mathematical tricks to accomplish to successfully work well with Back-propagation, being subsequently extrapolated to many different architectures with Back-propagation-based training through the years.

Using the same equation shown on Eq. 48, a local reparametrization trick [Kingma et al., 2015] is performed to be able to calculate the derivatives of the distribution parameters being learned. It adds a little variation into the parameters that are targets to be learned in form of Gaussian noise.

For example, for a Gaussian Distribution, it creates two parameters of interest in every weight where the derivative can be calculated and updated:

$$\theta = (\mu, \sigma^2) \tag{49}$$

$$\varepsilon \; N(0,1) \tag{50}$$

$$w = f(\varepsilon) = \mu + \sigma \bullet \varepsilon \tag{51}$$

The update process takes place in the following manner:

$$\Delta\mu = \frac{\delta f}{\delta w} + \frac{\delta f}{\delta \mu} \tag{52}$$

$$\Delta\sigma = \frac{\delta f}{\delta w}\frac{\varepsilon}{\sigma} + \frac{\delta f}{\delta \sigma} \tag{53}$$

$$\mu <- - \mu - \alpha\Delta\mu \tag{54}$$

$$\sigma <- - \sigma - \alpha\Delta\sigma \tag{55}$$

$$\theta_o pt = (\mu_o pt, \sigma_o pt) \tag{56}$$

This is a general step that is useful for any neural network with minor tweaking.

Bayesian VI Convolutional Neural Network [Shridhar et al., 2018]

Bayesian Convolutional Neural Networks with Variational Inference are a special case, since the convolution algorithm and the filters require further development in order to replace the scalar weights for probability distributions and be able to apply Bayes-by-Backprop.

As shown in Figure 14 [Shridhar et al., 2018], the comparison of a regular CNN with a Bayesian CNN, where filters must be replaced with probability distributions over weights.

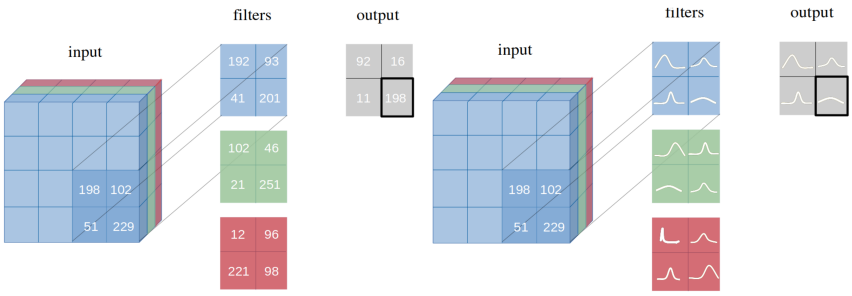

Figure 14: Graphical Comparison of Regular CNN(left) and Bayesian VI CNN (right)

This is done by using the local reparametrization trick [Kingma et al., 2015] by sampling layer activations rather than the weights. Then the variational posterior $q_\theta(\theta_{ijhw}|Data) = \nu(\mu_{\mathrm{i}jhw}, \alpha_{\mathrm{i}jhw}\mu^2_{ijhw})$ where i is the input layer, j is the output layer, and (w, h) are the dimensions of the input 2D-Tensor, resulting in the equation shown in Equation 57 for convolutional layer activation b, where A_i is the receptive field area and $\varepsilon_j \in (0, 1)$.

$$b_j = A_\mathrm{i} * \mu_\mathrm{i} + \varepsilon_j \cdot \sqrt{A^2_\mathrm{i} * (\alpha_\mathrm{i} \cdot \mu^2_\mathrm{i})} \tag{57}$$

Finally, the end result stacks with a Bayesian Multi Layer Perceptron to issue a probabilistic prediction as shown in Figure 15.

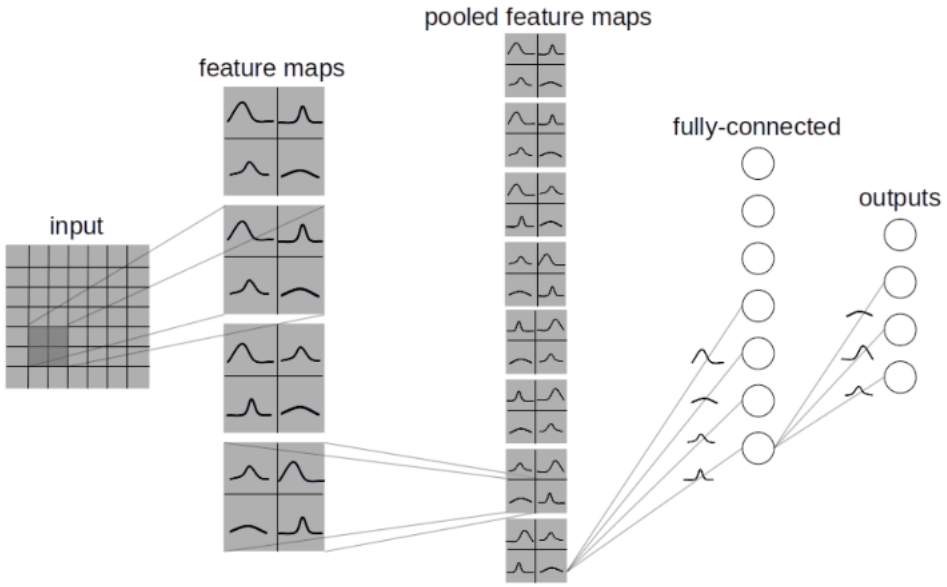

Figure 15: Graphical Example of Full Bayesian Variational Inference Convolutional Neural Network

6.4 Flipout[Wen et al., 2018]

Flipout is a novel method that build over Bayes-By-Backprop and aims to make Bayesian Neural Networks more efficient and perform better. Bayesian Neural Networks based on Bayes-By-Backprop Variational Inference are forced to use explicit weight perturbations, which generates an inefficient use of computational resources and a loss of performance, given that balancing the bias-vs-variance trade off becomes harder.

In order to achieve the ideal variance reduction, the gradients within a training mini-batch of instances should be uncorrelated. Overall the Flipout method aims de-correlate the gradient between instances, without biasing the stochastic gradients and obtaining a variance reduction between mini-batches, while using a vectorized form of perturbations that allows to parallel process through matrix multiplications achieving speed gains of over 40 times previous methods.

As explained in the paper, the work makes two basic assumptions about the weight distributions q_θ, where the perturbations of different weights are independent and their distribution is symmetric around 0. Given this, it proposes a random sign matrix E independent of the perturbation distribution q_θ. Then $W = q_\theta \cdot E$ is identically distributed as q_θ, and the loss gradient computed through it are also identically distributed.

What Flipout proposes is that by using a perturbation distribution q_θ shared by all the samples on the training mini-batch multiplied element-wise by a different rank-one sign matrix for each instance where n is the index within the training mini-batch, while r, n are random samples from a standard distribution $\in [-1, 1]$, so $W_\mathrm{i} = q_\theta \cdot r_\mathrm{i} s_\mathrm{i}^T$.

This allows to obtain an unbiased estimator for the loss gradients, also de-correlating the samples achieving a variance decrease on the sequential updates when averaging over a mini-batch during training. Extrapolated to multiple layer activations the formula obtained is observed in Equation 58 [Wen et al., 2018]. Where ϕ is the activation function, $r_\mathrm{i}, s_\mathrm{i}$ are the random sing vectors for all the examples of the training mini-batch.

$$y_\mathrm{i} = \phi(W^T x_\mathrm{i}) = \phi((\bar{W} + q_\theta \cdot r_\mathrm{i} s_\mathrm{i}^T)^T x_\mathrm{i}) = \phi(\bar{W}^T x_\mathrm{i} + (q_\theta^T (x_\mathrm{i} \cdot s_\mathrm{i})) \cdot r_\mathrm{i}) \tag{58}$$

Then the previous equation can be written in matrix form as shown in Equation 59. Where since R, S are sampled randomly and independently from q_theta, the gradients for $\bar{W}$, q_θ and X can be obtained through back-propagation.This allows the parallel processing in GPU and TPU units for efficient computation that allows the massive speed gain versus previous methods.

$$Y = \phi(X\bar{W} + ((X \cdot S)\ q_\theta)) \cdot R) \tag{59}$$

Section IV: Case Study - Machine Health Diagnostics based on Off-line Equipment Multi-Family Oil Analysis

1 Task Overview

The work done corresponds to a supervised classification task of machine fault diagnosis through an oil samples that has been provided kindly by MCM Ingenieria [15], a Chilean company that provides maintenance services for more than 120 companies on several industries. This particular data belongs to a big mining company water desalination, transport and supply system acquired over a period of 18 months. It comprehends several different equipment from the process that can be grouped in 4 major families, in which the machines are subjected to very different parts of the process, ergo, very different operational conditions.

The specific data used originates from a laboratory analysis performed over each particular oil sample, in which the chemical, physical and particular composition was analyzed by different methods such as element spectrometer, particle count, pH tests and burning tests. Then, these samples were tested and diagnosed by an expert on the laboratories from DICTUC [16], an external specialized company on the field that provide this service, where this diagnosis was later double-checked and confirmed by an expert from MCM.

There are 6 diagnostics that are **the most frequent ones**, but are not exclusive, as some times other very low frequency equipment faults may be found based on the oil samples, which are not less severe, and require different corrective actions.

The most frequent faults are related to component wear, as observed in Table 3, where they have different root causes that generate ultimately the same effect, while only one of them is related to unpredictable events that cause contamination by external agents such as seal leakage, faulty covers or other infrequent events caused by weather conditions or improper physical asset management.

Component wear is usually the cause of between 60% and 80% of all faults on machinery [Jingwei et al., 2012] progressively degrading the internal components and shortening their useful life, and, if unattended, in most occasions it involves very costly repairs due to catas-

[15] www.mcm.cl
[16] https://www.dictuc.cl/

Condition	Root Cause (Relation)
Normal	Healthy operational range
Silica Contamination & ISO-4406	Abnormal concentration of noxious particle sizes (Component Wear)
Silica Contamination & Wear	Incipient component Silica abnormal particle concentration (Component Wear)
Component Wear	Mechanical degradation of internal components (Component Wear)
Water Contamination	Abnormal water concentration in lubricant (Component Wear)
Oil contamination	Contamination by by external agents (Seal fault & Isolated events)

Table 3: Lubricant Health State Labels of the Physical Asset Condition

trophic failure or component wear propagation to other components due to lost lubricant effectiveness, requiring more downtime, safety concerns, specialized personnel and immediate availability of spare parts; therefore, most efforts to take the appropriate measures to predict the stage of wear of the internal components and take effective preemptive and corrective actions is fundamental for a proper maintenance policy focused on maximizing equipment reliability.

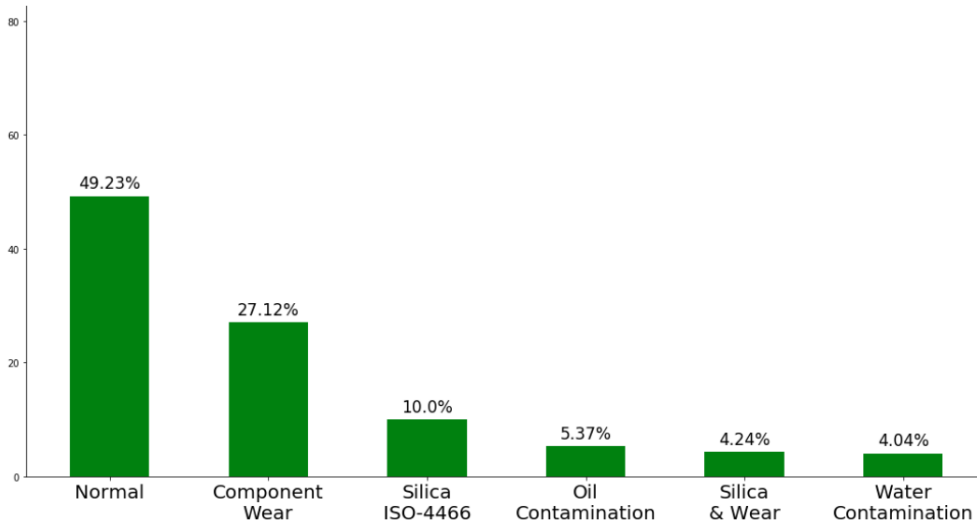

Figure 16: Fault State Distribution of the Dataset

Precisely because of the progressive nature of Component Wear faults and its related indicators, and as shown in Figure 16, 90.59% of the failures relate directly to component wear. Ergo, the case presents an excellent opportunity for data learning algorithms to be able to extract and learn exploitable patterns that can provide precise insights into the stages of severity related to this type of fault.

On the other hand, Oil Contamination Fault State, envelopes many different types of

contamination that can be caused by many different sources which degrade the oil and its lubrication efficiency like fuel, detergents, chemicals, etc. Given this, it is very difficult to extract a pattern from this specific fault as it is mostly comprised of faults that are infrequent and do not share common causes or indicators.

Nevertheless, the previously mentioned class is extremely important for diagnostics on an applied scenario, as it needs to be diagnosed correctly in order to take proper investigative actions to determine which corrective action should be applied; therefore, it needs a model that is able to either separate it properly as a class on its own or identify that it is not able to predict accurately this case and alert the human expert that the analyzed sample is beyond the model capabilities.

In the first case, it might be accomplished, but it would require a large volume of training data in order to identify the distinct patterns of different sub-faults present on the category to have a robust knowledge about the overall class.

A second, different, approach would require a metric that quantifies the lack of knowledge the model has regarding this specific class, in order to separate it and derive it to human analysis.

Considering that the task proposed in this work has a very high label imbalance due to the different frequency of occurrence of faults, as observed in Figure 16, and the fact that it should be applicable to a real world scenario where prediction reliability and interpretability should be key, this task is a complex task for traditional machine learning and deep learning.

Traditional models are not particularly robust to class imbalance, out of training distribution samples, not being presented with the entire universe of failures possible, and deal with multiple dimensions. The mentioned challenges, paint a complex scenario for traditional machine/deep learning models to be used and deployed [Hajizadeh, 2019], while probabilistic approaches seem more feasible as they model the probability patterns on the data, having more flexibility in their predictions and learnt representations.

The issue with deterministic models, is that an Out Of Distribution sample will be predicted as any of the classes it was shown during training, and have no way of handling these events in a reliable manner.

Therefore, the second approach mentioned has a lot more potential to be deployed in a task where the dataset and operating conditions are subjected to uncertainty over the input data and the results, and the maintenance process could benefit greatly from a model that only issues predictions when it is certain of the potential result.

2 Task Overview

The dataset is comprised of 12,504 samples with 29 features (362,616 data points), which is considered a sub-optimal amount of data for learning models, specially considering that the dataset of the case study has a very high variance, due to belonging to several different families of equipment subjected each to different operational conditions, having different

types of oils and having multiple fault states.

Amongst the most widely used option of a probabilistic models that allows to quantify the lack of knowledge of the predictions could be a Gaussian Process, but the large amount of features and data-points already exceeds the computational capability of the current models, generating major complexities on the development, training and deployment of the aforementioned model. Usually quantities over 10,000 data points are considered outside the ranges manageable by common Gaussian Processes, being automatically discarded from the potential candidates to be applied to this task.

From the stated above, this task is a very interesting case study to analyze and test the performance of scalable probabilistic models such as the Flipout Bayesian Deep Neural Networks, rising to be a feasible and interesting solution to deploy and actually aid in the diagnostics process as a differential diagnosis tool that can be used to automate and validate easy, certain predictions, while allowing for a starting point for more complex samples that the pattern has not been observed before, rejecting the prediction and deriving it to an analyst.

2.1 Dataset

As mentioned before, the dataset is comprised of 12,504 different samples of laboratory tests performed over lubricants extracted from sever different equipment subjected to different parts of the process and of different operational nature. The equipment can be grouped in 4 major families presented on Table 4.

Family	**Samples**	**Portion of Dataset**
Shafts	6670	53.33%
Gears	4662	37.72%
Hydraulics	641	5.11%
Pumps	531	4.22%

Table 4: Major Equipment Families in Dataset

Th fact that the dataset is comprised of the aforementioned equipment families, each of which has its own particular operational nature, and, where each equipment is subjected to its own specific operational conditions, specific oil types and special additives generates an undesirable high variance level of the data. Also, the samples can not be all considered reliable, as there is no way of determining if there is a degree of external interference, such as lubricant oil refilling or incorrect acquisition procedure, causing an decrease of the dataset's quality.

Important Assumption

Given the small size of data samples available to train the data learning algorithms, the samples will not be separated by equipment family, as this would reduce significantly the data available to train the model, degrading its robustness and generalization capabilities.

General intuition states that by introducing less data-inherent variance to the model, better performance could be achieved. But, general indicators of component wear in the lubricant

should be similar across moving machinery while independent of operational conditions and oil additives;therefore, the assumption that all equipment families share the indicators of component wear in the lubricant can be made without a loss of generality.

2.2 Features

Each sample has 29 feature columns regarding indicators of different laboratory results for particle, element, water concentration, physical properties.

A total of 22 of them correspond to measurements obtained by performing an Atomic Elemental Spectroscopy over the sample, which obtains the concentration in particles per million (ppm) of the metallic elements present in the oil. The type of elements studied during the analysis are categorized on 3 groups which have been previously defined based on knowledge of the effects they cause on the lubricant.

These 3 groups are Contaminant Metals, Additive Metals and Wear Metals. As their names state, Contaminant Metals are the most likely to be caused by external contaminants, Additive Metals are the most likely to belong to the oil which are depleted over time, and finally, Wear Metals are the most likely to indicate the internal component degradation which can be used also to identify which component specifically is failing from its composition.

The feature details can be seen in detail on Table 5 and Table 6. [17]

The spectroscopy test consists on using an external energy source to excite the atoms in a sample, causing them to release energy in the form of light, which in turn is acquired and sorted by the wavelength it presented. Each individual element has a specific wavelength, and the intensity helps quantify the amount of energy produced by the excitation, making it possible to determine the approximate concentration of the element present in the sample in parts per million. It is important to note that this test is usually only valid for particles below 7-10 microns, therefore it is important to have other tests to shine insights on particle concentration of larger sizes.

In this case, 3 features express insights into the particle size concentrations in the lubricant as shown on Table 7. Particle counting is used with this purpose, based on the [ISO 4406:2018(E), 2018] particle size count over certain thresholds which produce abrasion. Bigger particles give insights to potential external contamination or severe degradation from internal components.

[17] For more detailed general information: *en.oelcheck.com/wiki/The_elements_of_a_lubricant_check*

Indicator	Effect	Probable Origin
Iron	Wear Metals	Shafts, rolling-element bearings, cylinders, gears, piston rings
Chromium	Wear Metals	Piston rings
Aluminum	Wear Metals	Pistons, bearings, pumps, thrust washers
Copper	Wear Metals	Bearings, brass/bronze alloys, bushings, thrust washers, rotors
Lead	Wear Metals	Bearings, fuel additives, anti-wear additives, anodes
Nickel	Wear Metals	Bearings, valve train, turbine blades
Silver	Wear Metals	Bearing cage plating, gear teeth, shafts
Tin	Wear Metals	Journal bearings, bearing cages, solder
Titanium	Wear Metals	Bearing hub, shafts, rods, compressor blades
Vanadium	Wear Metals	valves, rods, rings and bearings, protective coating wear, rods, hard steel alloy wear
Cadmium	Wear Metals	Corrosion resistant cage protective plating

Table 5: Dataset Spectroscopy Analysis Features and Explanation Overview

PQ Index (Particle Quantifier Index) is a measurement that quantifies the mass of debris product of metallic wear in a sample and shows as an dimensionless coefficient based in Hall Effect[18] which disregards the particle size. It can be interpreted as a linear tendency in which the content of ferrous debris directly relates to the amount of component wear in the machine where the sample was taken; the higher the index more content of ferrous debris. It is specially useful to provide insights about particles above 10um that can generate serious wear.

Oxidation is the principal way of degradation of lubricant in service. Most lubricants are designed with additives that aim to control the oxidation by sacrifice, reacting and oxidizing before the base oil to provide protection increasing the lubricant's useful life. This additives are the only barrier of protection against premature degradation and failure of the lubricant, which in turn can have severe consequences for the machine components.

FTIR (Fourier-Transform Infrared Spectroscopy) is a test which aims to measure the concentration of the antioxidant additives present in the lubricant by the infrared emission or absorption of the particles suspended in the lubricant, and returns a percentage number when compared to the oil-when-new, quantifying the remaining useful life of the oil before is considered to be degraded.

[18]Voltage difference across an electrical conductor transverse to an electrical current and a perpendicular magnetic field

Indicator	Effect	Probable Origin
Manganese	Wear Metal	Steel, high-grade steel, non-ferrous-metal alloys, cathodes
Sodium	Contaminant Metals	Detergent or coolant additives
Potassium	Contaminant Metals	Coolant additives
Silica	Contaminant Metals	Dust, dirt, defoaming additives
Zinc	Additive Metals	Neoprene seals, grease, anti-wear additives
Barium	Additive Metals	Rust and oxidation inhibitor additives, grease
Boron	Additive/Contaminant Metals	Anti-corrosion additives in coolant, dust, hard water
Calcium	Additive Metals	Detergent/dispersant additives
Molybdenum	Additive Metals	Piston rings, electric motors, extreme-pressure additives
Magnesium	Additive Metals	Transmissions, detergent additives, hard water contamination
Phosphorus	Additive Metals	Anti-wear additives, extreme-pressure gear additives

Table 6: Dataset Spectroscopy Analysis Features and Explanation Overview (Continuation)

Indicator	Laboratory Test	Probable Faults
V40	Viscosity at 40 Celsius	fuel agents, oil degradation, water, oxidation
Water Content [%]	Karl Fischer Titration Moisture Analysis	Faulty seals, covers, plugs, o'rings
PQ Index	PQ Analyzer Hall Effect	Component Wear
Oxidation	Fourier Transform Infrared Spectroscopy	Additive depletion, overheating, overuse, external agents
Particles $>$ 4um	Particle Counting ISO-4406	External Contamination Wear/Impact
Particles $>$ 6um	Particle Counting ISO-4406	External Contamination Wear/Impact
Particles $>$ 14um	Particle Counting ISO-4406	External Contamination Severe Wear/Impact

Table 7: Dataset Features and Explanation Overview

Some effects of oxidation in lubricants are: sludge and sediment formation, increase in viscosity, base oil breakdown, PH increase, filter plugging & clogging, foaming, and corrosion.

Finally, water content in the lubricant is one of the most common contaminants that can cause problems in machinery. All lubricants have the capability of absorbing water into them up to a saturation point, ergo, the mere presence of water in the sample cannot be considered

as a fault. But if the lubricant emulsifies, very noxious effects can be unleashed given the capability of the water to free flow with the lubricant causing cavitation, surface impacts, surface wear, corrosion, hydrogen embittlement, chemical and physical changes in lubricant properties that lead to an important loss of useful life. All of which result in the need for immediate action to be taken, generally changing the oil and identifying the source of water. Therefore, it is a very important indicator to monitor the asset's condition.

Karl Fischer water test can be performed in on of two ways: Coulometric, which tends to be more accurate at lower concentration levels and Volumetric which is usually better at higher concentration levels. In the case presented case the first option is the one present in the features, expressed as a percentage.

The Coulometric test consists on generating a controled chemical reaction to react with the totality of water present in the sample, once the totality of water has reacted, the amount of water present is calculated by solving the stoichiometric equation respective to the chemical reaction. This allows to have very precise estimates that can detect water down to 1 part per million.

2.3 Machine Condition Label Characterization

Each feature vector of laboratory results pertaining to a sample, are accompanied with its respective label diagnosed by an expert from DICTUC and double-checked by an expert from MCM.

Even though the diagnostic process is certified, these diagnostics are based on the analyst criteria and the maintenance policy of the company; therefore, it is subjected to the possibility that the diagnostic might not be completely accurate, may present several different simultaneous faults that lie undetected, or are detected and disregarded due to a hierarchical fault tree. All of this favours the presence of uncertainty on the dataset, making it harder to isolate the specific fault patterns.

The dataset has been assumed to have only 6 different labels, which correspond each to a root fault that causes degradation of the machine diagnosed by an expert. As component wear is the principal and most frequent cause of faults in moving machinery, and directly related to the lubricant, most of the labels are related to this specific fault which is divided into different root causes according to the physical asset specific indicators of degradation. These faults along with the portion of the dataset they fill can be observed on Figure 17.

The figure above states the root causes of faults that are not considered to be related but most generate component wear. It is important to note that this are the most common faults that are regularly diagnosed in the analyzed machinery, and there are other fault states that are extremely infrequent such as degraded oil quality, changes in physical properties, etc. which did not have more than half a dozen samples and were disregarded from the dataset. Therefore, the complete universe of failures is not exactly known and present on the case study data.

In the subsection below, each fault will be characterized in detail about their cause, effects and severity.

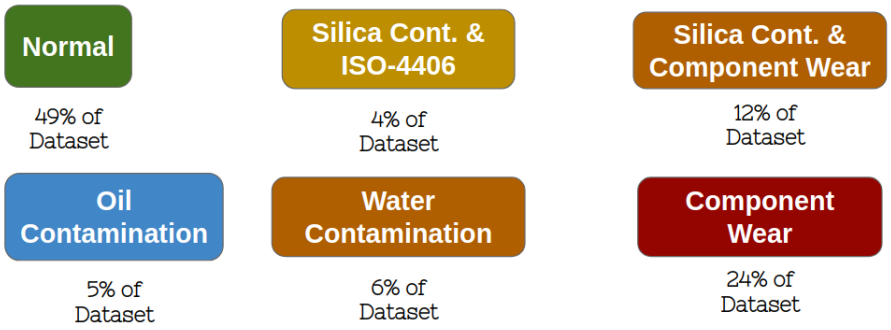

Figure 17: Labels Relation of the Fault States & Distribution in the Dataset

Healthy

Samples which condition is considered to be amongst normal operating ranges.

Water Contamination

Samples that are considered to be contaminated with a concentration of water beyond an acceptable level.

It is important to note that there is a particular allowance criteria for each equipment family on the concentration of water present in the sample, so the mere presence of water does not directly indicate directly this fault.

Water presence on lubricants is extremely undesirable, having many negative effects on components as it leads to component corrosion, protective film loss of effectiveness due to density difference, alcohols and acids formation due to lubricant corrosion, loss of heat dissipation effectiveness, loss of additive effectiveness, oil foaming causing cavitation, and, in piston engines or hydraulics, may cause catastrophic failure due to the incomprehensibility of water.

Most of the effects mentioned above relate directly the presence of water to early wear and failure of mechanical components. Nevertheless, it is easy to diagnose and repair if caught on time, but if left unattended it can dramatically decrease the useful life of the machine components.

Given the plethora of noxious effects it generates in the lubricant, the successful and timely diagnosis carries a high priority to carry an effective preventive maintenance plan. Ergo, it is regarded as one of the most important early indicators to prevent more severe degradation.

Silica Contamination & ISO 4406 [ISO 4406:2018(E), 2018]

Silica is a very hard material which is monitored due to its potential property to be able to scratch metal surfaces generating subsequent component wear. It is typically introduced into the system as very small particles through the air intakes or outlets, usually when the air filters fail to capture them due to filter physical limitations or damage. These particles stay suspended in the lubricating oil and, if the correct size, get pushed into the lubricating film formed in the clearance space between moving components. Here, they get lodged between

the surfaces and bridge them directly, cancelling out the effect of the lubricating film causing surface scratching, surface fatigue and breakup. In turn, this increases operating temperature, oil consumption and concentration of metal filings, which multiply and propagate the component wear effect by further hindering the lubrication effectiveness.

ISO-4406 is an international standard to report the level of solid contamination on fluids. There is a code for the amount of particles, categorized into 3 levels sorted by a size of 4, 6 and 14 microns. This code only reports the particle concentration concerning a certain size, which, depending on the criteria and machinery involved, qualifies as enough concentration to flag as contamination.

On this specific problem, Silica Contamination is an unavoidable problem since the mining site's operational conditions favor the presence of extremely fine silica particles generated by the various processes to break down minerals to very fine powders that facilitate transport and separation. Fine silica is suspended in the air, depositing inside the machinery involved in the breakdown processes, air filters and surfaces, while also present on the slurry and water transported by the pumps which may leak into lubricants of equipment involved.

The Silica Contamination &ISO-4466 label, indicates that there is an abnormal concentration of silica particles in the lubricant, therefore action must be taken to change the lubricant and determine the provenance of the particles if there is an abnormal point of entry before it turns into a situation that evolves into internal component degradation by wear.

Silica Contamination & Component Wear

On the other hand, the label referring to Silica Contamination & Wear, is related to the case that there is an abnormal concentration of noxious silica particles and wear debris mainly composed of metallic particles, which indicates directly that there is a degree of component wear along with silica contamination.

In this case, more in-depth corrective measures must be taken to ensure the reliability of the machinery, such as changing the damaged components along with the lubricant, and determining if there is an abnormal situation that caused the silica concentration.

The two different labels which involve Silica contamination can be interpreted as levels of severity of the contamination and how much it has contributed to the severity of the degradation, which result in different corrective actions for each diagnostic.

Both of the previous labels serve as a preventive approach to Component Wear, allowing to take corrective action before there is further damage to the asset, maximizing the useful life of its internal components.

Component Wear

Component Wear is a severe machine fault state which is determined when the indicators show high concentration of particles in the oil that were part of the machine internal components.

Higher contact tolerances left by the worn components may produce impact, lack of lubri-

cating film effectiveness, metal filings and particles that are suspended in the oil propagate and degrade other components of the machine, by speeding up the wear of other components, also, the physical properties of the lubricant are compromised reducing its effectiveness and heat dissipating characteristics generating a dramatic reduction the overall life of the complete machine.

To correct this, components and lubricants must be changed, and an extensive analysis of the damage produced to other components must be carried out being very costly in downtime, spare parts and the loss of net remaining useful life of the physical asset.

Oil Contamination

In general, this class is destined for different low frequency events such as contamination by burnt oil, detergents, fuel, and other external agents that may reduce oil effectiveness. The common solution is to change the oil and verify there are no failed valves, seals or covers.

The overall issue with this class, is that there is no underlying pattern shared among the different events of which it is composed. Therefore, extracting a pattern from this group is relatively hard for data learning models.

3 Training

3.1 Procedure, Challenges & Solutions

The dataset procured comes from real world operations, and was not initially thought out to be used for Data Analytics or machine learning purposes. This is a common problem found across the industry, specially in companies that are starting to adopt the digital transformation, and yet have not reached maturity in data acquisition and storing practices.

Given that there is no complete certainty about the diagnostics or the operational conditions the machine might have been subjected to when the sample was taken, as there is the possibility of multiple simultaneous faults being present, human error of the sample failing to recognize a diagnostic and unknown fault states never observed before. This causes an inherent uncertainty about the cleanliness[19] of the data, not feeding any samples that might cause add misinformation to the model.

Since machine learning algorithms learn from data, the data management assumptions made on the training stage is the core on which all the posterior work is based on.

Training and Test Set Building

The dataset is shuffled and then divided into 75% of data to be trained and a holdout set of 25% of the data for evaluation, which will only be seen by the model during the testing phase.

[19]Data is referred as clean when the information contained in it is clear for pattern recognition

Managing Outliers

Since the equipment families and operational conditions have a very high variance, outliers, which are samples that stray far away from the multidimensional clusters of each label class have are dealt with by either removing them from the dataset or reducing their influence during the process of training the model.

For this specific approach, two measures are taken that ensure a proper level of variance control over the training data.

First, if we assume that the samples of each class were generated from a normal distribution, all the samples that are within 4 standard deviations of the average (99,99% of the data) are kept for training phase. This allows to eliminate the singular extreme outliers that contaminate the pattern extraction during the training phase.

Second, the scaling of each of the dataset's features independently based only on the training data relevant statistics is performed using Scikit Learn's[20] RobustScaler, which is a percentile based scaler that removes the median and scales the data according to the inter-quantile[21] range of each specific feature.

Given the large amount of variance the dataset has and the fact that some laboratory measurements could be disproportionate or badly imputed to the database, generating many outliers which can dramatically influence the sample mean and variance in a way which affects the scaling process in a prejudicial way for machine learning applications, RobustScaler is chosen as it often provides better results by assigning less weight, or none at all, to outliers.

After this step is performed, a PCA analysis is done over the training data in order to determine visually if the problem is feasible, as shown in Figure 18, where it can be observed that the clusters present a very consistent definition for each label class, but present an overlapping rather than a clear separation. This is probably due to the high dimensional nature of the data, where all the samples share common indicators that remain constant through some faults and does not signify by any means that the classes are not separable in higher dimensions.

Class Imbalance

Class imbalance is mainly caused as not all faults present themselves with similar frequency, causing that over a limited span of time, the amount of faults acquired is not symmetrical. This presents a challenge during training of data learning models, as makes them prone to over-fit[22] causing them to generalize poorly on samples that have not been observed during training.

There are a number of approaches possible to get better results while working with a limited volume of data. Given the nature of this problem no data generation or augmentation can be performed as the non-linear relationships among variables is unknown. On the other

[20]Scikit Learn is a Free Machine Learning Python Library with high level functions for direct application

[21]Range between 25th and 75th quantile

[22]Maps noise patterns of data. Concept similar to memorizing the training data, preventing from generalizing the patterns to new cases.

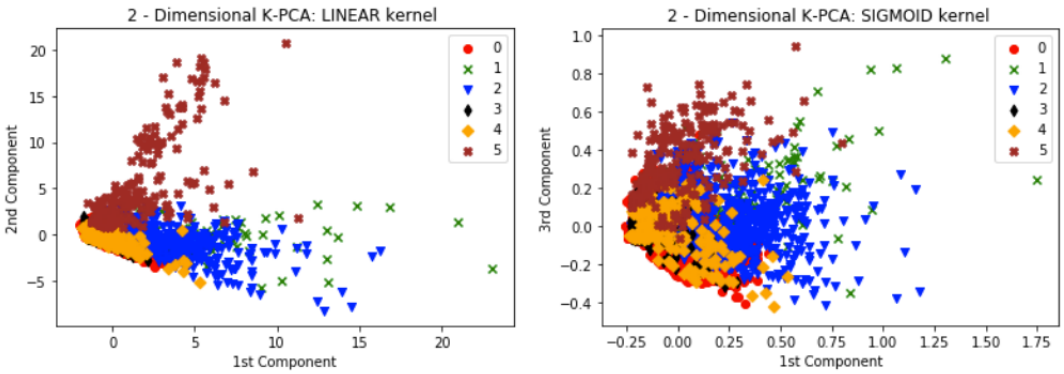

Figure 18: Kernel Principal Component Analysis 2D Visualization of the 6 classes pre-processed and scaled data with two methods: Linear Kernel (left) & Sigmoid Kernel (right).

hand, giving larger penalization to incorrect answers of infrequent classes opens a new set of challenges that can bias the results and favor overfitting. Therefore, it is concluded that the class imbalance will be left as is.

In spite of that, different models have different ways of dealing with class imbalance. Probabilistic models hold an advantage here, as they aim to learn the distribution that is most likely to have generated the data from which is learning, rather than extrapolating from observations. Tree-based models also are robust to class imbalance given their information-gain based leaf separations. Finally, Neural Networks should be the most affected as, given they are very flexible, they are very prone to over-fit with low volumes of data and high class imbalance.

Curse of Dimensionality

The 'Curse of Dimensionality' refers to an undesirable phenomena that can be described in simple words as: *'when the dimensionality of a problem increases, the combination of possible coordinates in said dimensions grows combinatorially, ultimately making the available data suffer from sparsity as the dimensional space grows'.*

In other words, as more dimensions are added to the samples of the data, more information is needed to fill the spaces between data samples as the euclidean distance between each sample increases. This sparcity generates a loss of statistical significance of the results, ergo, the volume of data needed to support the result reliably grows exponentially with the dimensions of the data.

As an intuitive example, the amount of data needed to know all the possible sample coordinates inside the dimensional space is proportional to the combinatorics of the dimensions. Using as a reference the case study presented, if we consider the previous statement, the amount of data needed to know the whole evidence space of 29 features it translates to a factorial combination of 29!, equalling over 8.8 x 10^{30} samples, which is an unthinkable amount of data, even impossible considering that this real world operational and takes a great amount of time to gather.

Sometimes, increasing the dimensionality through feature engineering[23] might be beneficial for more powerful machine/deep learning models as it can further contribute to separate samples that are not clearly separable in lower dimensions or add more clear and significant information about the phenomena.But, if dimensionality gets too high, due to sparsity the performance of the machine learning models tends to drop. This is commonly called Hughes Phenomenon[Hughes, 1968].

In the studied case, no feature engineering can be performed over the experimental oil samples as there is no further information that can be extracted from the values of the sample features.

On the other hand, reducing the dimensionality of the data is also sometimes performed when features are considered to be redundant or not contributing information to the problem. This can be done from a qualitative perspective in the cases where the feature nature is clearly unnecessary, or quantitative if the relationship between features is not clearly known.

In this case, multiple analysis were performed over the 29 features to determine if dimensionality could be reduced, and all of them deemed that there was an important loss of information by disregarding any feature. The two most representative analysis are presented below, allowing to visualize the feature correlations and hierarchical relations.

The Pearson Correlation Coefficient is the covariance of two variables divided by the product of their standard deviations. It is a metric that expresses the measure of linear correlation between two variables, and holds a value of o if the variables are uncorrelated or -1/1 if it is total negative/positive correlation.

The heat-map shown in Figure 19 shows the correlation amongst the 29 features present on the data, where lighter pink is total positive correlation, black is total negative correlation and bright red is no correlation. The heat-map helps to visualize in a more clear manner the relations between multiple features that are being compared versus all other.

As it is observed, most of the features present one or more, principally negative, correlations to other features. There are no two features that present exactly the same correlation pattern or that are completely decorrelated from the rest of the features. Also, some expected relations can be observed such as the positive correlation among the particle sizes, an as particle size increases, the correlation decreases between features (bottom right); and, the elemental metal concentrations that are also positively correlated (top left) as component wear progresses, more metal particles are present in the oil and speed up other metal surfaces degradation.

Product of this, and the fact that there is already a lack of information assumed due to the high dimensionality and low volume of data, this analysis suggests clearly that no feature can be discarded from the dataset.

The second test, shown in Figure 20, provides great insights into the hierarchical tree-like relations amongst the features and how they relate to each other in the learning process of

[23]Creating more informative features based on a single feature through mathematical or categorical transformations.

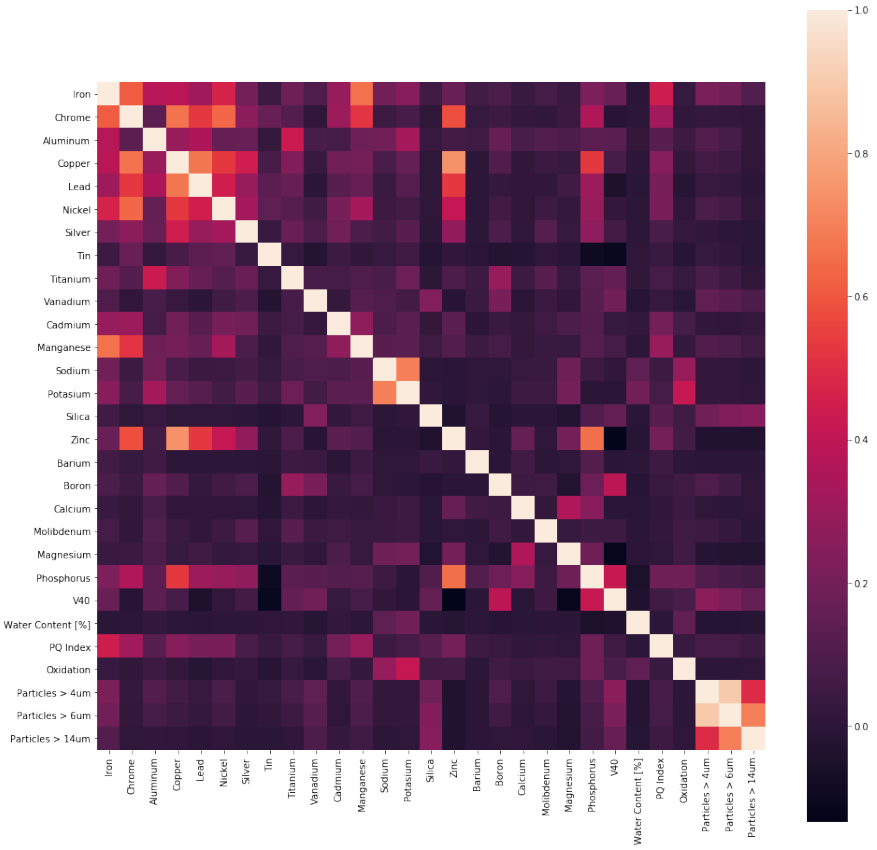

Figure 19: Pearson Correlation Heat-map

a machine learning model. Usually, the criteria to eliminate features based on this diagram is if the relations depicted between two features are very close to 0, meaning that they are redundant, or the whole 'branch' is close to 0, where the combination of features is not providing useful information to the model. As observed in the figure, most variables provide

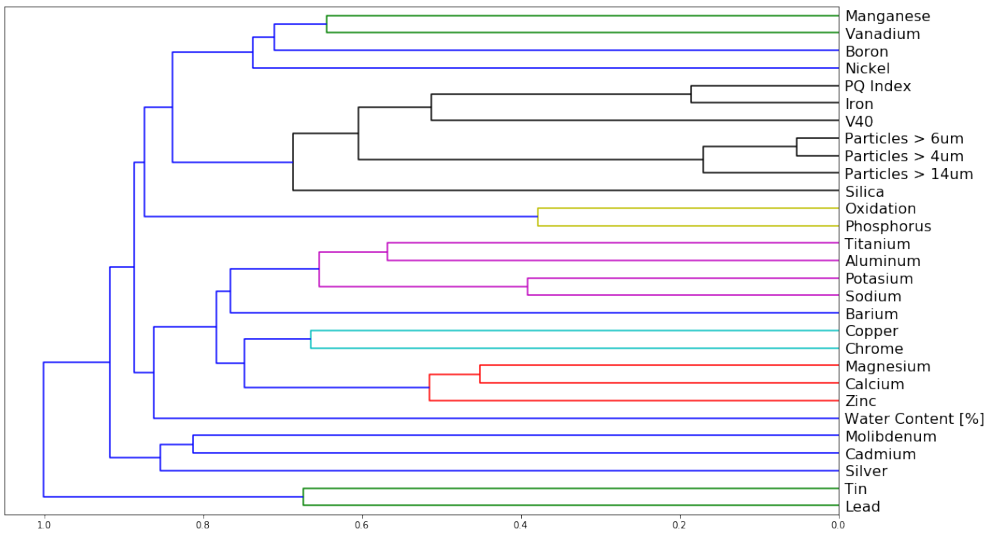

Figure 20: Hierarchical Clustering of Features Dendrogram

a very robust amount on information. The only features that are closer to 0 are the ones related to particle size concentration, where the more similar ones, which control a very narrow interval of sizes as 4 um and 6 um, are closer to be potentially redundant, but enough to consider disregarding one of them.

It is important to note that most all performed where to investigate the linear relation between features, whereas underlying non-linear relations can also be present between variables. But, as no feature was discarded through the linear analysis, posterior non-linear analysis were not required.

Out Of Distribution Samples

As stated before, on a real world setting Out Of Distributions (OOD) samples are very common for a number of reasons that are not identifiable directly from the analysis of the data. These usually can be grouped in single-time events, infrequent events, or acquisition error.

An advantage of probabilistic models such as Gaussian Processes and Bayesian Neural Networks in this realm, is the ability to estimate the model's lack of knowledge over each prediction, being specially useful to detect out-of-distribution samples and avoid an inaccurate overconfident prediction like other models, rerouting them to a human analyst.

The Training stage only contemplates 5 out of the 6 fault states, holding out the Oil Contamination, which envelopes several unrelated faults, to simulate as Out of Training Distribution (OOD) events that might happen in real-world operations.

4 Models & Parameters

The models that are trained using the case study data correspond to a mixture of the most commonly known, and state-of-the-art models. They are currently used in machine competitions and industrial applications with very good results, where each model has its own mathematical basis and interacts differently with the data learning process in its own manner, having advantages and disadvantages depending solely on the task and desired scope.

4.1 Tree-Based Models

These deterministic models are based on splitting the data in a hierarchy that goes from general-to-particular, imitating a tree growth were general similar classes are part of the trunk, and particular sub-classes are progressively divided further into branches and leaves. Their simple principles and interpretability makes them one of the most used algorithms for machine learning tasks, being also very powerful for supervised classification or regression tasks.

Different variations tend to use different criteria to split the data on each node, and given they use node splitting criteria, they are particularly good at managing unbalanced classes. On the downside, this same advantage makes the models very rigid, not performing well

against OOD events.

Decision Tree

The decision tree is the most basic estimator for this category, composed of a single tree that splits the data based on Gini coefficient or information entropy criteria. This makes it very simple to visualize and interpret, but tends to be very easy to bias and over-fit. Also, they do not handle well on non-numeric data, and are not scalable to big data.

Ensemble Models - Random Forest & Extra Randomized Trees

In order to handle the bias vs. variance trade-off[24], a combination of **N** decision trees, each trained on a random combination of a fraction of the total features of the data, is assembled together via averaging their predictions. This generates a model based on a collection of decorrelated trees that follow the bootstrap aggregating principle, allowing for more robust results as they are able to capture more complex structures and interactions amongst features than a single Decision Tree.

The only difference between Random Forest and Extra Randomized Trees is the splitting criteria for the trees, where Random Forest models calculate the data's optimal-split using Gini coefficient or information entropy to split its nodes, while Extra Randomized Trees uses a random value to split its nodes relying in the trees averaging to converge to the optimal splits. This generates that ERT needs more estimators but is considerably faster, while random forest is slower but relies in less trees.

Ultimately, both estimators usually perform very similar on most tasks being scalable, efficient, robust and easy to use. On the downside is very easy to over-fit in noisy tasks, it cant handle OOD's reliably, cannot deal with categorical or non-numerical data, and even though it provides valuable insights regarding feature importance it is very hard to interpret.

Gradient Boosted Models - XGBoost & LGBM

Boosting is a concept that bases itself on stacking sequentially several weak learners, such as Decision Trees, into a stronger one by training each model to fix the errors of its predecessor. Gradient Boosting is a method that builds upon the previously mentioned method by fitting the next predictor to the residual errors made by the previous predictor by the Gradient Descent Algorithm.

Usually this implementations have better performance than plain tree-ensembles and a huge comparable scalability, but, given the sequential fitting of models they have to perform, take much longer in computation time than ensemble algorithms; nevertheless, modern techniques have been developed lately that speed up training enough to make it very similar, or even faster, than ensemble algorithms.

Two state-of-the-art models based on this method have had extremely good performance on Kaggle competitions lately, Extreme Gradient Boosting [Chen and Guestrin, 2016b] and

[24]Referred as finding the learning equilibrium optimal point between over-fitting and under-fitting

Light Gradient Boosted Machine [Ke et al., 2017a]. Both this algorithms have ways of dealing with missing data, regularization to avoid overfitting and have superb scalable capabilities.

XGBoost is based in a level-wise tree growth and posterior pruning to keep the model as shown in Figure 21[25], also, it has several optimized features such as parallel task-processing tree building, depth-first tree pruning , regularization, missing data handling, built in cross validation.

Figure 21: XGBoost Level-Wise Growth Mechanism

On the other hand, as shown in Figure 22 LightGBM uses a leaf-wise tree growth, and its main feature is that it uses a node split algorithm based on discrete histogram bins of the data, which increases the speed of the model dramatically during training and inference, while introducing an amount of randomness that decreases overfitting. Also, the feature that makes this model even faster than XGBoost, is an algorithms called Gradient Based One Side Sampling (GOSS) which retains instances with large gradients (large error nodes) and samples a small portion of the small gradients (low error nodes) which it assumes they have 'learned'. This gives a dramatic speedup on training the sequential models, making it one of the strongest and most robust tree-based models in the present.

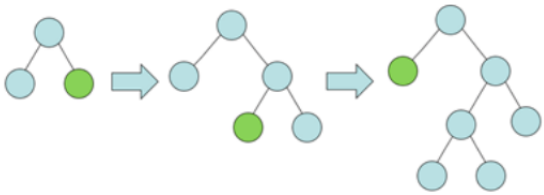

Figure 22: LGBM Leaf-Wise Growth Mechanism

4.2 Deep Neural Networks

Deep Neural Networks are currently the most flexible and powerful deterministic models in the field of artificial intelligence. These are extremely efficient scalable models that are able to capture linear and non-linear relationships, perform complex feature extraction, process non-numerical data, deal with extreme high dimensional data and parallel process millions of samples; while achieving exceptional performance on most, supervised and unsupervised, classification and regression tasks.

Nevertheless, this flexibility has the downside of finding the right architecture for the learning task and a complete non-interpretability of the decision process followed by the

[25]Image Source: $https://towardsdatascience.com/lightgbm-vs-xgboost-which-algorithm-win-the-race-1ff7dd4917d$

model. This generates that the training and calibration of the models has to be very rigorous to apply in real-world settings or risk related tasks, and even then they are susceptible to over-confident predictions on out of training distribution samples .

Multi-Layer Perceptron[Rosenblatt, 1957]

MLP's are the most basic neural network architecture used to process vectorized non-time sequenced data. They are a keystone in most complex architectures, which are mainly based in feature extraction layers with the objective to feed suitable data vectors to a MLP in the intermediate or final layers.

This neural network architecture is specially suitable for a simple, vectorized and supervised task like the case study, as there is no time sequences, multidimensional arrays or non-numerical data.

Convolutional Neural Network[LeCun et al., 1989]

This type of neural network was designed with the objective of processing data in multidimensional arrays, and has evolved to the point of being able to process as RGB video samples which has up to 5 dimensions (arrays, height, width, channels, time).

This network is probably one of the most powerful architectures to this day, focusing mainly in high dimensional feature extraction, allowing to learn very intricate and complex patterns on structured, and unstructured, data; while lower in complexity than multi-layer perceptrons.

On this case a CNN1D which convolves over the one dimensional vector for feature extraction is appropriate to aid in the supervised classification task.

4.3 Bayesian Neural Networks

Bayesian Neural Networks is a research field of deep learning that has recently exploded due to the need for interpretability and uncertainty estimates over scalable models, specially in those that search for application on real-world conditions.

The intuition behind the most accepted approach, Bayes-by-Backprop, is based on taking advantage of the mechanisms that have been widely studied for conventional neural networks, making minor mathematical modifications which allow to train and make inference as proposed on 'Weight Uncertainty in Neural Networks' [Blundell et al., 2015].

Mainly the technique is based on replacing the network parameters for parametric probability distributions, as shown in Figure 23[26], trainable via Variational Inference using a loss function based on an Elementary Lower Evidence Bound and reverse Kullback-Leibler Divergence, which measures the similarity between probability distributions (Figure 24[27]). The back-propagation training is made possible by sampling the parametric distribution one

[26]Image Source:[Blundell et al., 2015]

[27]Image Source:$https://blog.evjang.com/2016/08/variational-bayes.html$

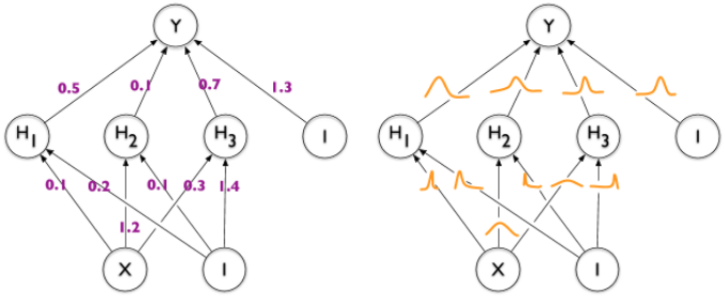

Figure 23: Intuition of Conventional Neural Network (left) vs Bayesian Neural Network (right)

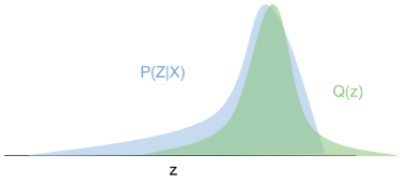

Figure 24: KL-Divergence measures the similarity between P(x) and Q(x)

time during training and adding a weight perturbation into the parameters to update the parametric distribution's mean and the variance of each parameter during each iteration.

For inference, the model is fed the same sample **n** times, taking one Monte Carlo sample of each learnt parametric distribution per run. This allows to build an output which depicts prediction certainty as the median and the confidence as the standard deviation observed. As shown in Figure 25, running a sample 100 times, the model's predictions with higher confidence will have lower STD, while over-confident uncertain predictions will appear as with a very high standard deviation.

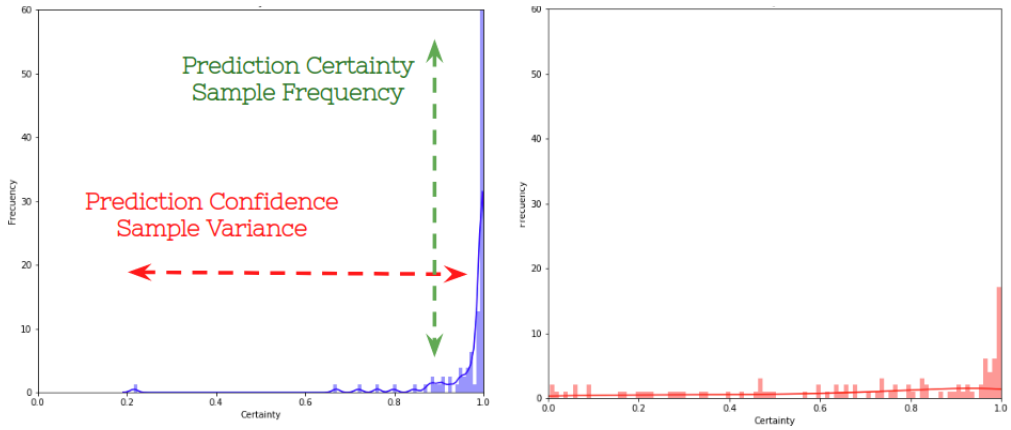

Figure 25: Certain and Confident Prediction (left) vs Over-Confident Prediction (right)

Flipout

Flipout is a method that builds over the previously stated model, by proposing an efficient way of de-correlating the gradients by pseudo-independent perturbation of the weights in vectorized form on a training mini-batch.

This method achieves an unbiased gradient and lower variance updates between training-mini batches, achieving a regularizing effect over the distributions; and also, a massive increase of computational GPU operations throughput through the reformulation of perturbations in matrix form versus the previous explicit perturbation approach, allowing for parallel processing making networks dramatically faster to train.

Models Evaluated for the Task

Model	**Base Concept**
Decision Tree	Tree-Based
Random Forest	Tree-Based Bootstrap Aggregating Ensemble
Extra Randomized Trees	Tree-Based Random Split Bootstrap Aggregating Ensemble
XGBoost	Tree-Based Histogram Split Regularized Gradient Boosting
Light Gradient Boosting1	Tree-Based Histogram-GOSS Split Gradient Boosting
Multi Layer Perceptron	Neural Network
Flipout Bayesian Multi-Layer Perceptron	Bayesian Neural Network
Convolutional 1D	Convolutional Neural Network
Flipout Convolutional 1D	Bayesian Convolutional Neural Network

Table 8: Models Evaluated for Comparison

In Table 8, a general overview of the models compared in the case study is shown along with their base model concept.

4.4 Hyper-parameter Tuning: Bayesian Expectancy Improvement

In order to find the closest to optimal results in each model, automatic hyper-parameter tuning through Bayesian Expectancy Improvement is performed over each of the models in Table 8 to optimize the models hyper-parameters to ensure the best individual classification performance.

Hyper-Parameter Tuning

Hyper-parameters are fixed values that control how the model works, such as initial conditions and estimator architecture settings, defined before the beginning of the training phase. These define the learning power and learning mechanisms of the model.

The more user-adjustable settings the model has, the search space for this values increases combinatorially due to the aforementioned 'Curse of Dimensionality', where changing one hyper-parameter affects the behavior of most others, increasing the computational time and complexity to find the optimal solution.

Most machine learning problems try to solve a multidimensional non-convex optimization problem by finding parameters that are able to map the behavior observed in the data, guided by minimizing an error function. Therefore, depending on the values selected for the hyper-parameters different models will be obtained.

Hyper-parameters are a mixture of Boolean, discrete and continuous values that set a plethora of different initial conditions which vary from model to model.

Some usual techniques, that are progressively more computationally expensive as more hyper-parameters are added, such as Grid Search or Random Search, are respectively based on exploring an user-set number of hyper-parameters for a combinatorial number of iterations based on the possible combinations amongst the desired search space, or explore randomly the hyper-parameters, over a user-defined interval, in a set number of iterations.

Both of this methods return the results of a score function defined by the user for each of the models, along with the hyper-parameters of the respective iteration.

Typically, to decrease the probability of biasing the model generalization capabilities by optimizing the model to the specific available dataset, the score function used to rank the best model is based on K-Fold Cross Validation, which splits the training set in K parts, trains K models, and then evaluates the accuracy of predictions with one of the K-Folds averaging the results in the final score.

Bayesian hyper-parameter optimization takes advantage of previous information about the hyper-parameters effect on the target score obtained through exploration, in order to exploit specific relevant areas of interest in the search space that have the potential to provide an increase on the model's performance.

Bayesian Hyper-Parameter Optimization balances the trade-off of exploration vs. exploitation[28], in contrast to the other methods mentioned before which favor almost completely exploration.

As it progressively evaluates more combinations, the Bayesian Optimization model learns how the values of different hyper-parameters affect the target result, guiding the value selection for hyper-parameters through the search space by updating the prior belief based by the current result. This generates the ability for this method to to provide a much faster and precise approximation for the best model hyper-parameters combination.

Algorithm

The objective is to create a surrogate model approximation of the error as a function of the hyper-parameters, an iterate through the results until convergence into a maxima for the target function where the change in values is sufficiently close that the impact on the score is small.

Basically, the optimization process is performed as it follows, observed graphically in figure 26:

- Using previously evaluated points it computes a posterior expectation for the target score.
- Samples the values at the computed expectation
- Compares the previous model score with the current score and accepts the hyper-parameter for the prior probability update if the result improves.
- Repeat until user-set convergence tolerance between variations in the hyper-parameter values is achieved.

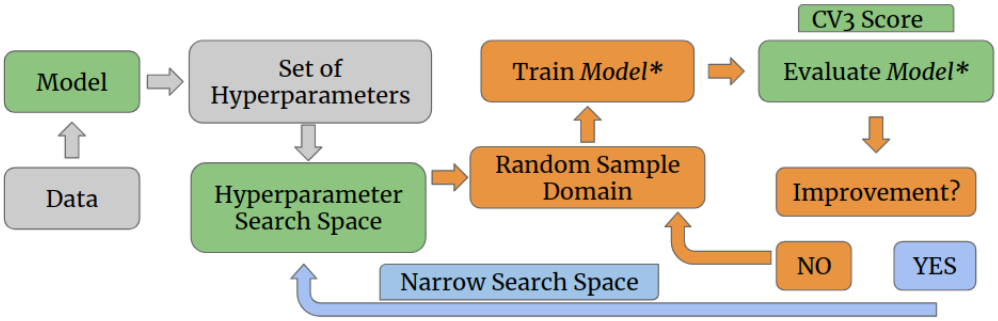

Figure 26: Flow Diagram of Bayesian Hyper-Parameter Optimization

For the process is important to note that to allow further exploration and escape from local minima, a parameter is set in the optimization process to sample, with a user set probability, a random point anywhere in the search space not based in the prior.

In order to calculate the Expectancy Improvement the prior distribution assumes that the

[28]The computational resources can be used either exploring a wide search space globally for an overview, or exploiting a local constrained search area for detailed results.

expectation function can be approximated via a Gaussian Process, where given an Acquisition Function shown in Equation 60, where x is the current optimal set of hyper-parameters, $\overline{x}$ is the current observed values, and GP is the Gaussian Process, The Expected Improvement can be calculated in order to draw a new guided sample.

$$EI(x) = \mathbb{E}[max\{0, GP(x) - GP(\overline{x})\}] \tag{60}$$

In short, the method's objective is to first canvas the search space and rapidly identify the most relevant areas to explore, in order to converge to the hyper-parameters that generate the best results for the model. It predicts given a prior probability how hyper-parameters will behave, and then samples a point from the parameters space which is most likely to improve the results, making an update of the prior belief based on the ongoing samples.

Model Hyper-Parameter Optimization on this Task

For this task, each model's Bayesian Optimization is exhaustive, meaning that it envelopes all the user defined hyper-parameters over a large search space with a maximum of 1500 iterations if it does not converge sooner, and each is run 4 times for each model. The target score function, given the class imbalance and low volume of data, is defined to be the average of a 3-fold cross validation split. Finally best scoring model hyper-parameters are kept for definitive training.

Section V: Case Study Results

1 Model Architecture & Test Results

In the following section, the model hyper-parameter and architecture results obtained with the Bayesian Optimization are presented along with the best model evaluation results over the holdout Test Set.

The Bayesian Optimization Plots and the Training Curve Plots (Accuracy/ Loss over Training & Validation) are shown in the Appendix 1 for each model where applicable.

1.1 Evaluation Metrics

In problems where the classes are imbalanced, the Accuracy metric is not reliable enough to be applied confidently for model evaluation.

Therefore, in order to compare between models, the F1 Score, a metric based on the balance of Precision and Recall, is used instead.

In addition, as a general comparison evaluation method, the Confusion Matrix is an invaluable tool to calibrate, evaluate and interpret the model's learning. It allows to extract important insights from the quality of the model's learning through the predictions made versus the real labels. This makes evident any underlying pattern on a classification task specially within similar classes.

All the models performance is compared in terms of F1 Score and Confusion Matrix Interpretation. Also, the number of parameters, the training and inference times are evaluated for external model comparison.

1.2 Threshold Selection

The respective median and standard deviation minimum and maximum value thresholds for the selected class of the Flipout Neural Networks concerning the rejection of the model uncertain predictions, are respectively calibrated and selected using a criteria that implicates a 25% rejection of the total Test Set predictions.

This returns a minimum Median prediction value and a maximum Standard Deviation that is admissible for a prediction to be deemed reliable under the knowledge perspective of the model.

Model: Decision Tree	
Hyper-parameter	**Result**
Node Split Criterion	Gini Coefficient
Maximum Depth	25
Class Weight	Balanced
Iterations to Convergence (Bayesian Optimization)	43
Evaluation	**Results**
Average Training Time	120 ms
Average Inference Time (2,959 samples)	2 ms
Accuracy Score	82.27% ± 0.21%
Precision Score	82.09% ± 0.24%
Recall Score	82.27% ± 0.21%
F1 Score	82.17% ± 0.23%

Table 9: Decision Tree Hyper-Parameter & Evaluation Results

1.3 Decision Tree Full Results

In Table 9 and Figure 27 the results for this estimator metrics and predictions are shown respectively.

Figure 27: Decision Tree - Confusion Matrix of Best Model

Random Forest	
Hyper-parameter	**Result**
Number of Estimators	941
Node Split Criterion	Gini Coefficient
Maximum Depth	75
Maximum Features	Auto
Warm Start	True
Class Weight	Balanced
Iterations to Convergence (Bayesian Optimization)	54
Evaluation	**Results**
Average Training Time	4,980 ms
Average Inference Time (2,959 samples)	204 ms
Accuracy Score	88.23% ± 0.05%
Precision Score	88.18% ± 0.06%
Recall Score	88.23% ± 0.05%
F1 Score	87.85% ± 0.05%

Table 10: Random Forest Hyper-Parameter & Evaluation Results

1.4 Random Forest Full Results

In Table 10 and Figure 28 the results for this estimator metrics and predictions are shown respectively.

Figure 28: Random Forest - Confusion Matrix of Best Model

Extra-Randomized Trees	
Hyper-parameter	**Result**
Number of Estimators	982
Maximum Depth	50
Maximum Features	Auto
Bootstrap	False
Warm Start	True
Class Weight	Balanced
Iterations to Convergence (Bayesian Optimization)	158
Evaluation	**Results**
Average Training Time	3,080 ms
Average Inference Time (2,959 samples)	506 ms
Accuracy Score	86.77% ± 0.15%
Precision Score	86.56% ± 0.14%
Recall Score	86.77% ± 0.15%
F1 Score	86.3% ± 0.15%

Table 11: Extra-Randomized Trees Hyper-Parameter & Evaluation Results

1.5 Extra-Randomized Trees Full Results

In Table 11 and Figure 29 the results for this estimator metrics and predictions are shown respectively.

Figure 29: Extra Randomized Trees - Confusion Matrix of Best Model

Light Gradient Boosting Machine	
Hyper-parameter	**Result**
Boosting Type	Gradient Boosted Decision Tree
Learning Rate	0.464
Lambda L1	0.419
Number of Leaves	25
Maximum Depth	10
Maximum Features	Auto
Minimum Data	100
Sub-Feature Sample	0.6
Class Weight	Balanced
Iterations to Convergence (Bayesian Optimization)	155
Evaluation	**Results**
Average Training Time	226ms
Average Inference Time (2,959 samples)	19ms
Accuracy Score	87.97% ± 0.0%
Precision Score	87.74% ± 0.0%
Recall Score	87.97% ± 0.0%
F1 Score	87.77% ± 0.0%

Table 12: LGBM Hyper-Parameter & Evaluation Results

1.6 Light Gradient Boosted Machine Full Results

In Table 12 and Figure 30 the results for this estimator metrics and predictions are shown respectively.

Figure 30: LGBM - Confusion Matrix of Best Model

1.7 Extreme Gradient Boosting Full Results

In Table 13 and Figure 31 the results for this estimator metrics and predictions are shown respectively.

Figure 31: XGB - Confusion Matrix of Best Model

Extreme Gradient Boosting	
Hyper-parameter	**Result**
Boosting Type	Gradient Boosted Decision Tree
Learning Rate	0.031
Gamma	1.343
Number of Estimators	352
Number of Leaves	25
Maximum Depth	41
Minimum Child Weight	4
Column Sample	1
Objective	Binary-Logistic
Iterations to Convergence (Bayesian Optimization)	152
Evaluation (5 models)	**Results**
Average Training Time	13,300ms
Average Inference Time (2,959 samples)	73ms
Accuracy Score	89.02% ± 0.0%
Precision Score	88.86% ± 0.0%
Recall Score	89.02% ± 0.0%
F1 Score	88.85% ± 0.0%

Table 13: XGBoost Hyper-Parameter & Evaluation Results

Convolution 1D Neural Network	
Hyper-parameter	**User Set**
Loss	Cross Entropy
Optimizer	Adam
Epoch	100
Early Stopping Patience	5
Validation Split	0.2
Hyper-parameter	**Results**
Architecture	Table 15
Batch Size	32
Dropout	0.3
Total Parameters	189,949
Trainable Parameters	186,643
Iterations to Convergence (Bayesian Optimization)	152
Evaluation (5 models)	**Results**
Average Training Time	16,100ms
Average Inference Time (2,959 samples)	106ms
Accuracy Score	85.33% ± 0.57%
Precision Score	85.39% ± 0.46%
Recall Score	85.33% ± 0.57%
F1 Score	85.28% ± 0.84%

Table 14: CNN1D Hyper-Parameter & Evaluation Results

1.8 Convolutional1D Neural Network Full Results

In Table 14, Table 15 and Figure 32 the results for this estimator metrics, model architecture and predictions are shown respectively.

Sequential CNN1D Architecture			
Input Layer			
Batch Normalization Layer			
Convolutional 1D Layers			
Layer ID	**Filters**	**Kernel**	**Activation**
1	13	1	ReLU
2	57	1	ReLU
Flatten			
Batch Normalization Layer			
Dense Layers			
Layer ID	**Neurons**	**Dropout**	**Activation**
3	108	0.3	ReLU
4	34	-	ReLU
Output	5	-	Softmax

Table 15: CNN1D Architecture Results

Figure 32: CNN1D - Confusion Matrix of Best Model

1.9 Multi Layer Perceptron Neural Network Full Results

In Table 16, Table 17 and Figure 33 the results for this estimator metrics, model architecture and predictions are shown respectively.

Figure 33: MLP - Confusion Matrix of Best Model

Multi Layer Perceptron	
Hyper-parameter	**User Set**
Loss	Cross Entropy
Optimizer	Adam
Epoch	100
Early Stopping Patience	5
Validation Split	0.2
Hyper-parameter	**Results**
Architecture	Table 17
Batch Size	36
Dropout	0.25
Total Parameters	41,372
Trainable Parameters	41,372
Iterations to Convergence (Bayesian Optimization)	10
Evaluation (5 models)	**Results**
Average Training Time	15,200ms
Average Inference Time (2,959 samples)	115ms
Accuracy Score	84.12% ± 0.44%
Precision Score	83.96% ± 0.29%
Recall Score	84.12% ± 0.44%
F1 Score	83.86% ± 0.39%

Table 16: MLP Hyper-Parameter & Evaluation Results

Sequential MLP Architecture			
Input Layer			
Batch Normalization Layer			
Dense Layers			
Layer ID	**Neurons**	**Dropout**	**Activation**
1	116	0.25	ReLU
2	119	0.25	ReLU
3	126	0.25	ReLU
4	67	-	ReLU
Output	5	-	Softmax

Table 17: Multi-Layer Perceptron Architecture Results

1.10 Flipout Convolutional1D Neural Network Network Full Results

In Table 18, Table 19 Figure 34 and Figure 35 the results for this estimator metrics, model architecture, predictions and predictions after threshold are shown respectively.

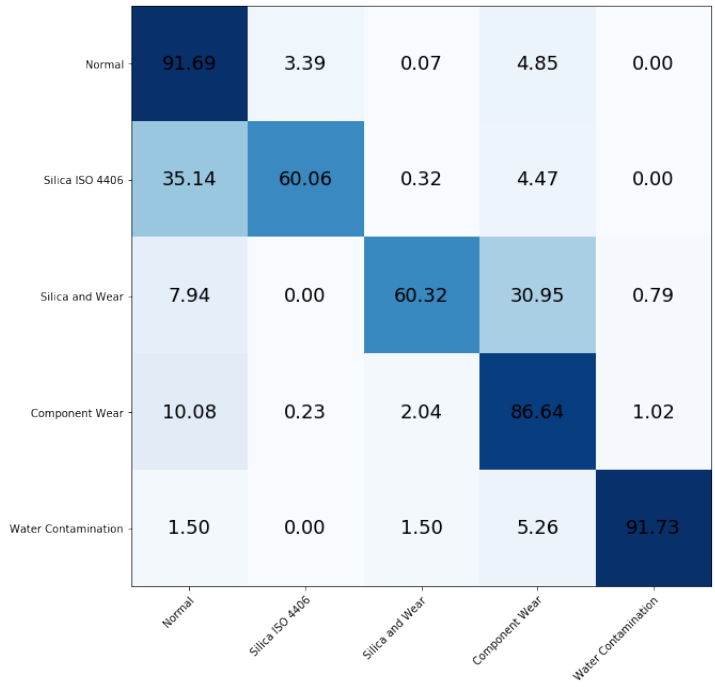

Figure 34: Flipout CNN1D - Confusion Matrix of Best Model

Flipout CNN1D	
Hyper-parameter	**User Set**
Loss	-LogLikelihood + KL Divergence
Optimizer	RMSprop
Learning Rate	0.001
KL Complexity Cost	10
Epoch	550
Early Stopping Patience	-
Validation Split	0.2
Hyper-parameter	**Results**
Architecture	Table 19
Batch Size	441
Dropout	-
Total Parameters	1,150,233
Trainable Parameters	1,143,513
Iterations to Convergence (Bayesian Optimization)	33
Evaluation (5 models)	**Results**
Average Training Time	36,100ms
Average Inference Time (2,959 samples)	4,840ms
Accuracy Score	85.14% ± 0.22%
Precision Score	84.86% ± 0.22%
Recall Score	85.14% ± 0.22%
F1 Score	84.67% ± 0.24%
Kappa Score	76.14% ± 0.34%
Kappa Verdict	Substantial Agreement
Lack-of-Knowledge Filter	
Threshold: 25% Samples Rejection	
Minimum Certainty (Median)	70% Threshold
Minimum Confidence (Std)	±25% Threshold
Rejected Predictions	750 ± 32
Rejected Portion	25.3% ± 1%
Accuracy Score	92.72% ± 0.22%
Precision Score	92.54 % ± 0.22%
Recall Score	92.72% ± 0.22%
F1 Score	92.36% ± 0.24%
Kappa Score	87.23% ± 0.34%
Kappa Verdict	Substantial Agreement

Table 18: Flipout CNN1D Hyper-Parameter & Evaluation Results

Figure 35: Flipout CNN1D - Confusion Matrix after-Filter

Sequential Flipout CNN1D Architecture			
Input Layer			
Convolutional 1D Layers			
Layer ID	**Filters**	**Kernel**	**Activation**
1	56	2	ReLU
2	72	1	ReLU
3	120	1	ReLU
Flatten			
Batch Normalization Layer			
Dense Layers			
Layer ID	**Neurons**	**Dropout**	**Activation**
4	161	-	ReLU
5	67	-	ReLU
6	50	-	ReLU
Output	5	-	Softmax

Table 19: Flipout CNN1D Architecture Results

1.11 Flipout MLP Neural Network Network Full Results

In Table 20, Table 21 Figure 36 and Figure 37 the results for this estimator metrics, model architecture, predictions and predictions after threshold are shown respectively.

Figure 36: Flipout MLP - Confusion Matrix of Best Model

Figure 37: Flipout MLP - Confusion Matrix after-Filter

Flipout Multi-Layer Perceptron	
Hyper-parameter	**User Set**
Loss	-LogLikelihood + KL Divergence
Optimizer	RMSprop
Learning Rate	0.001
KL Complexity Cost	10
Epoch	800
Early Stopping Patience	-
Validation Split	0.2
Hyper-parameter	**Results**
Architecture	Table 21
Batch Size	502
Dropout	-
Total Parameters	66,599
Trainable Parameters	66,541
Iterations to Convergence (Bayesian Optimization)	33
Evaluation (5 models)	**Results**
Average Training Time	8,200ms
Average Inference Time (2,959 samples)	3,150ms
Accuracy Score	83.22% ± 0.26%
Precision Score	82.99% ± 0.26%
Recall Score	83.22% ± 0.26%
F1 Score	82.19% ± 0.39%
Kappa Score	72.55% ± 0.5%
Kappa Verdict	Substantial Agreement
Lack-of-Knowledge Filter	
Threshold: 25% Samples Rejection	
Minimum Certainty (Median)	65% Threshold
Minimum Confidence (Std)	±30% Threshold
Rejected Predictions	772 ± 11
Rejected Portion	26.1% ± 0.4%
Accuracy Score	91.00% ± 0.20%
Precision Score	90.91 % ± 0.24%
Recall Score	90.00% ± 0.20%
F1 Score	90.16% ± 0.23%
Kappa Score	83.86% ± 0.3%
Kappa Verdict	Substantial Agreement

Table 20: Flipout Multi Layer Perceptron Hyper-Parameter & Evaluation Results

Sequential Flipout MLP Architecture			
Input Layer			
Batch Normalization Layer			
Dense Layers			
Layer ID	**Neurons**	**Dropout**	**Activation**
1	189	-	ReLU
2	94	-	ReLU
3	99	-	ReLU
Output	5	-	Softmax

Table 21: Flipout MLP Architecture Results

2 Overall Model Performance Comparison

2.1 F1 Score Model Comparison

Model	F1 Score
Decision Tree	82.17% ± 0.23%
Random Forest	87.85% ± 0.05%
Extra Randomized Trees	86.30% ± 0.15%
Light Gradient Boosting	87.77% ± 0.00%
XGBoost	88.85% ± 0.00%
Multi Layer Perceptron	83.77% ± 0.53%
Convolutional 1D	84.64% ± 0.44%
Flipout MLP	82.19% ± 0.39%
Flipout MLP 25% Threshold	90.16% ± 0.23%
Flipout Convolutional 1D	84.67% ± 0.24%
Flipout Convolutional 1D Threshold	92.36% ± 0.24%

Table 22: F1 Score Model Comparison

2.2 Time & Number of Parameters Comparison

Model	Parameters/ Estimators	Training Time	Inference Time (2,958 samples)
[model]	[number]	[ms]	[ms]
Decision Tree	1	120	2
Random Forest	941	4,980	204
Extra Randomized Trees	982	3,080	506
Light Gradient Boosting	-	226	19
XGBoost	-	13,300	73
Multi Layer Perceptron	41,372	15,200	115
Convolutional 1D	189,949	16,100	106
Flipout MLP	66,599	8,200	3,150
Flipout Convolutional 1D	1,150,233	36,100	4,840

Table 23: Model Parameters & Time Comparison

Model	Predicted Class Min Median	Predicted Class Max Std
Flipout MLP	65%	30%
Flipout CNN1D	70%	25%

Table 24: User Defined Thresholds for 25% Test Rejection

Test Set 25% Rejection Threshold Calibration			
Class	**Predictions**	**Rejected**	**Accepted**
Normal	1,504	241	1,263
Silica & ISO-4406	313	177	136
Silica & Wear	126	78	48
Component Wear	883	197	687
Water Contamination	133	37	96
Total	2,959	729	2230
Total Predictions Filtered: 24.64%			

Table 25: Lack-of-Knowledge Filtered Samples Detail - Flipout CNN1D

2.3 Simulated OOD Response

The initially separated class of 'Oil Contamination' is used as a Out of Training Distribution Sample (OOD).

It is run through the models simulating a real world scenario where a sample could be affected by incorrect sample acquisition, external contamination an unknown/infrequent health state; where the optimal behavior of the model is rejecting to support that specific prediction due to lack of knowledge over the correct outcome.

Through any of the deterministic models the capability of reproducing this behavior is restricted, classifying the sample as any of the classes observed during training, while in the best probabilistic neural network (CNN1D), the result is presented on Table 26.

External 'Oil Contamination' as OOD			
Class	**Predictions**	**Rejected**	**Accepted**
Normal	463	217	246
Silica & ISO-4406	88	88	0
Silica & Wear	76	56	20
Component Wear	30	10	20
Water Contamination	14	4	10
Total	671	375	296
Total Predictions Rejected: 55.89%			

Table 26: Rejected OOD

2.4 Graphical Results Examples from Flipout Neural Networks

The output of 100 Monte Carlo samples for each network can be used to generate a distribution plot for each sample that generates a prediction.

This plots can be used to interpret the results in a way that can't be done with deterministic models.

The following distribution plots are taken from the Flipout CNN1D model to exemplify several cases. Only the correct label will be shown along with the 'second best' option. For

the OOD plots, the default class is set to 'Normal' as in an operational context the prediction, even though uncertain, should not raise alarm until a fault is detected.

Confident Prediction Example

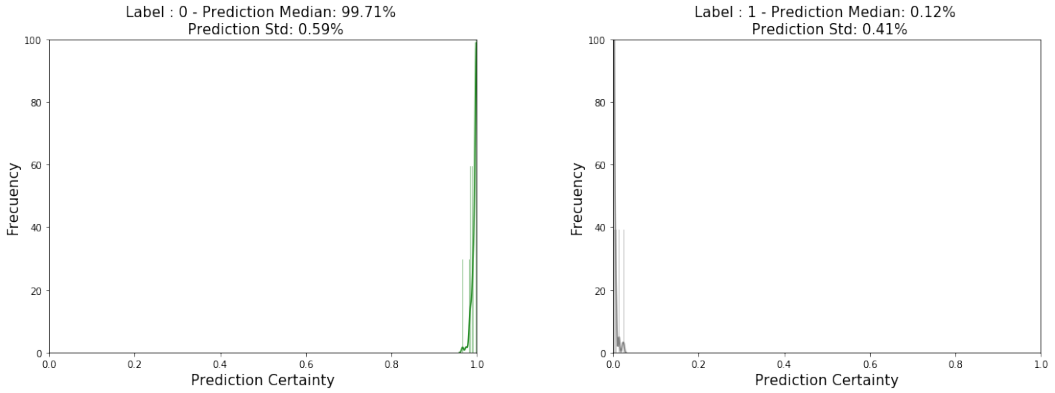

Figure 38: Confident Class Predictions Example

Wrong Under-confident Prediction Example - Median Rejected

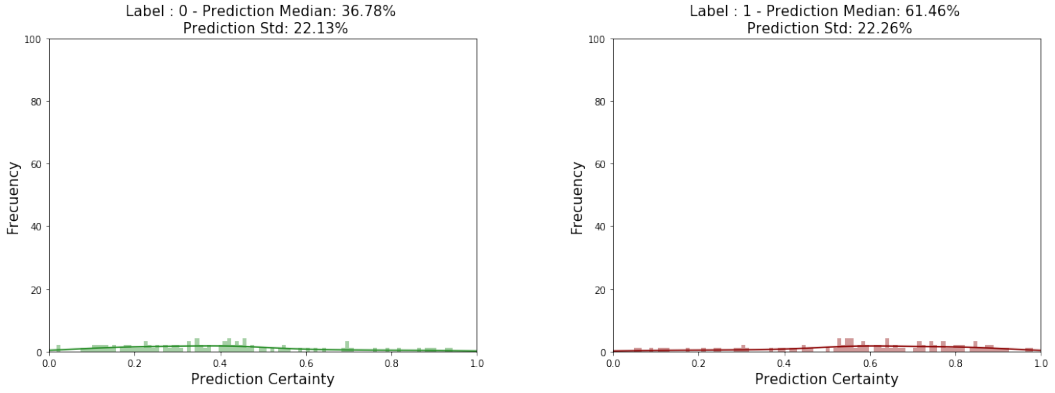

Figure 39: Wrong Prediction Rejected Example

Multiple Fault or Transition Fault State Identification

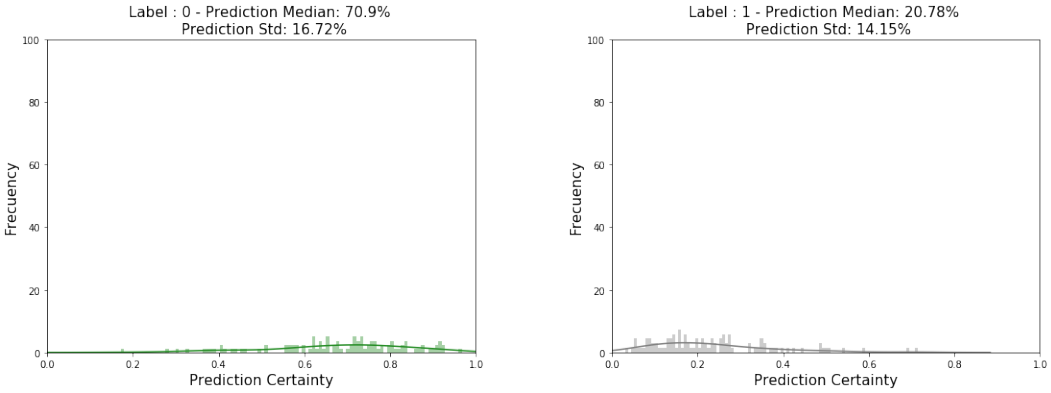

Figure 40: Model Might be Capturing a Multiple/Incipient Fault State

Accepted Simulated OOD

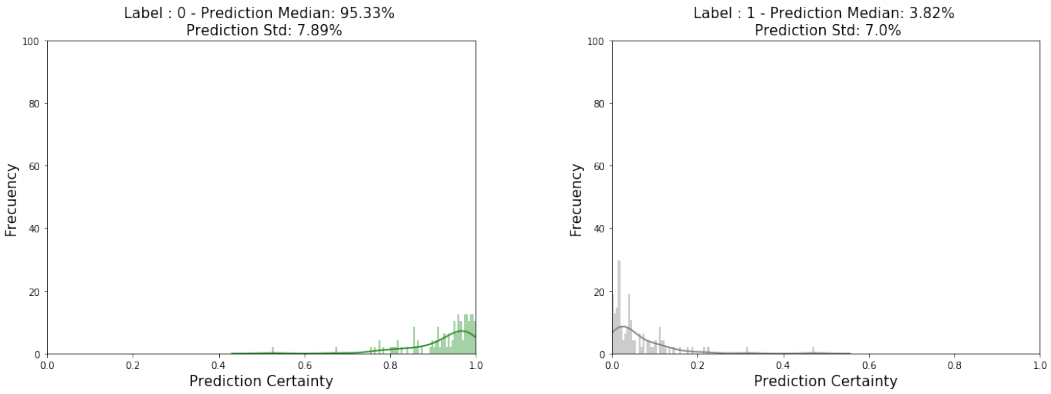

Figure 41: Accepted Sample from the OOD Simulated Example

Rejected Simulated OOD - STD Rejected (Over-confident)

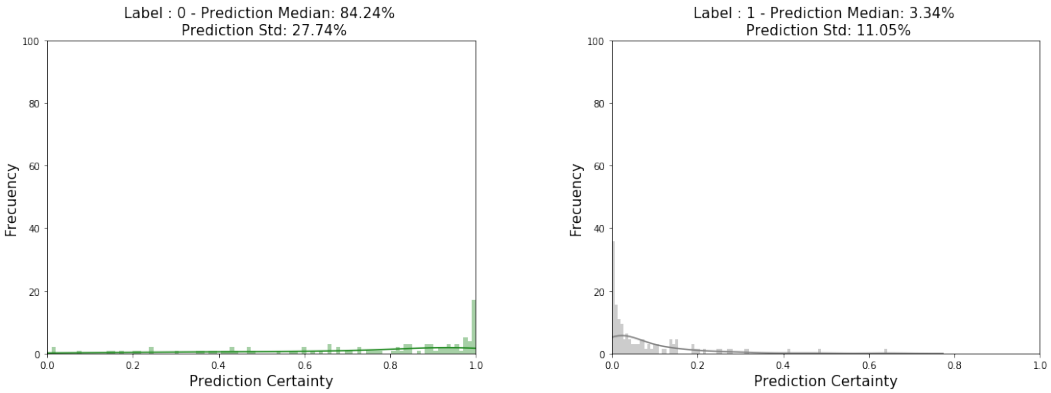

Figure 42: STD Rejected Example

Rejected Simulated OOD - Median Rejected (Over-confident)

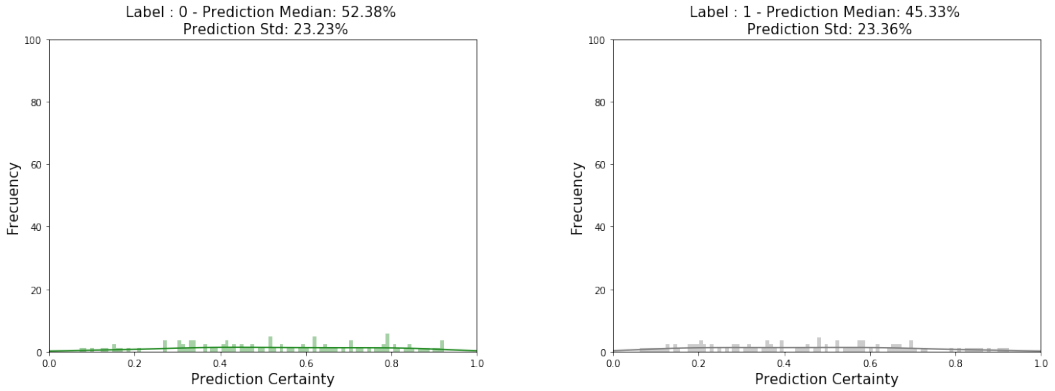

Figure 43: Median & STD Rejected Example

Section VI: Discussion

1 Bayesian Hyper-Parameter Optimization

The optimization of hyper-parameters through this method has several advantages over other methods such as Grid Search or Random Search, exploring and exploiting the hyper-parameter search space in a guided fashion, insuring fewer overall evaluations of the objective function to find an acceptable optima, resulting in less computational cost and time.

The full results for each model can be found on the Tables in Section V and Appendix I.

The criteria to determine a successful hyper-parameter optimization for each model does not rely on the convergence to the same set of hyper-parameters every time, but rather the objective function output stability on a certain vicinity, implying that a set of optimal solutions for the model learning has been reached consistently, and the model is not likely to improve its performance beyond that point.

The fact that the results for the Test data F1 Score, shown in Table 22, have a low and similar standard deviation across models, where the most observed variance lies in the most flexible models, hints that the set of hyper-parameters found for each model is consistent through multiple training's, resulting in model stability.

An important factor that hints a successful optimization on Table 22, is the fact that all models, based on very different principles, converge to a F1 score vicinity where the outcomes of each model is congruent with what is expected based on their own learning attributes and capabilities, resumed in Table 8, given the case study data.

A clear example of the stated in the last paragraph, is the fact that models with similar principles share very similar performance, while the significant differences in performance are observed while comparing to other models based on different principles, such as ensemble models as Random Forest with Extra Randomized Trees, or gradient boosted models like XGBoost with LGBM in comparison with the tree-based Decision Tree or any of the Neural Networks. For this, the models can be assumed to be working close to their best learning hyper-parameter configuration by balancing the 'bias vs variance' trade-off.

This strongly evidences that the differences observed on the models performance on Table 22 can be attributed to their inherent working principles, independent of the effect of hyper-parameter variation, generating the opportunity for the models to be evaluated solely and reliably for their learning working principles.

2 Model Comparison

2.1 Time

As it is expected and shown in Table 23, models with more parameters have larger training times. For the structured nature and amount of data relevant to the case study presented, time concerns are not crucial, but is an important variable to be taken under consideration for extrapolation to similar data-heavy tasks.

The most impressive time difference is observed between both of the Gradient Boosted models, which are sequential and not parallel processed, where both having similar working principles have a massive Training time and Inference time difference.

This is the product of the design of LGBM, aimed towards computational speed on big data settings, exceeding speed gains of 10x, or even 100x in some cases, over XGBoost with similar performance. And, as observed in Table 23, there is a impressive x58 speed increase between XGBoost and LGBM with extremely similar performance, which in larger tasks may account for a significant amount of savings on computational resources and time.

On the other hand, as Neural Networks have larger numbers of parameters, they present the highest training times. Flipout Neural Networks are expected to have even larger training times than regular networks with the same architecture, as for every single parameter of a deterministic network, the probabilistic alternative has two, doubling the number of parameters. Also, during training there are two back-propagation steps, one for mean and other for variance.

In this specific case, the Flipout Neural Network training times have a faster training time for a larger number of parameters, as the networks were programmed bare-bones[29], in contrast to the regular Neural Networks that use high level libraries, generating a considerable speed boost on the training. This is why the Flipout networks seem to train faster, while in equality of conditions they are slower.

Regarding Inference times, these are very low in average for trained models with the exception of Flipout Neural Networks. This time x30 increase is caused by each sample passing through the model 100 times (user-set) to be able to get the classes prediction distribution, which is analogue to performing inference over 295,800 samples rather than the original 2,958.

This causes that Flipout Neural Networks have a significant downside on tasks that rely on fast inference processing, having a trade-off between the prediction probability distribution definition and the inference time, where a more defined posterior needs more samples , and ergo, more time. This specially impacts its application, and must be taken into consideration for more complex tasks which are real time or time critical, and require larger networks or are very data-intensive.

For this specific case study, the diagnostics are not remotely time critical and the inference time of one sample is still under one second, but it is important to highlight the networks downside for extrapolation into bigger tasks.

[29]Most of the code is contained within the script without RaiseWarnings, RaiseError or external sources.

Parameters

An important note on the parameters, beside the fact that Flipout Neural Networks have double the parameters than a regular neural network, is that most of the learnt distributions upon finishing training are mostly redundant or null, where subsequent model weight pruning can be performed up to 95% of the network without loss of performance as shown in Bayes-by-Backprop paper [Blundell et al., 2015]. This same feature that allows a network to, given a gigantic number of parameters such as the Flipout CNN1D in Table 23, not over-fit the data as a regular neural network would probably do.

This presents an advantage for Bayesian Neural Networks to work with little volumes of data, being able to generalize the same or better than a deterministic Neural Network.

Confusion Matrix

Important insights about the data and training process are gathered through the analysis of the confusion matrices obtained in conjunction from all the models.

First, it is observed that all of the models have successfully learnt the patterns to identify the 'Normal', 'Component Wear' and 'Water Contamination' classes, obtaining a high success rate on predicting these classes. These specific classes have an acceptable data volume and/or several marked indicators that allow to find the specific patterns that separate the classes almost optimally.

Given the large variability of operational conditions, equipment and equipment families of the dataset, the minor miss-classifications on these classes can be explained by the use of different thresholds to diagnose the faults specific to each machine by the human analysts, generating a lot of dispersion on the potential boundary hyper-planes.

As the boundaries of class separation are learnt where the class separability is maximized, due to the inherent variance and sample overlapping not all samples are expected to fall within the correct side of the division. This means that the samples closer to the boundaries experiment a higher degree of prediction uncertainty from the model, which is exactly what it is observed in the transitions from the Confusion Matrix in Figure 34 (FCNN1D) and Figure36 (FMLP), to the 'lack-of-knowledge' Filtered Confusion Matrix in Figure 35 (FCNN1D) and Figure 37(FMLP). Here, it is appreciated that an important number of samples that were initially miss-classified by the Bayesian Flipout Neural Network are filtered from the results product of the reason stated above; the closer they are to a division hyper-plane between two classes, the uncertainty towards which class it belongs increases.

On the contrary, the 'Silica & ISO4406' and 'Silica & Component Wear' have the highest rate of miss-classification consistent in magnitude throughout the models. This is attributable to the fact that, as shown in Figure 16, this two classes are have only a few examples, and can be interpreted as conceptually overlapped, sharing many core indicators with other classes.

As explained in the Label Characterization sub-section of Section V, ISO 4406 is a fault state solely based on a particle size count, which due to the variability of the dataset it generates a cluster overlapping with 'Normal' class; as different equipment families, or sin-

gular equipment's, can have each different specific criteria[30] to define the particle/.Silica size and/or count that implies the fault state. And, the model's prediction being based differentiating on only 3 or 4 features related to particle count, which most likely are overlapped with most or all of the 29 features of the 'Normal' class, it becomes clear why most of this class samples are being miss-classified as 'Normal'.

The same issue can be attributed for the 'Silica & Component Wear' miss-classifications towards the 'Component Wear' class, as the wear feature indicators, shown in Table 5 and Table 6, carry more weight at the time of making a prediction that the Silica concentration feature alone. This adds up to the fact that 'Component Wear' is the second largest class, 5 times larger than 'Silica & Component Wear' as shown in Figure 16, generating that the 'Component Wear' class has more 'gravity'[31], 'pulling' the predictions towards itself.

Therefore, it can be concluded that these miss-classifications are inherent of the data, and have to be resolved by either increasing the affected classes data volume in order to extract more meaningful patterns that allow for the correct separation of the class, or diminishing the dataset variance by training the models in individual equipment families. Sadly, adding more data is not possible at the moment but might be done in the future when more samples are provided by the client. And regarding the second possible solution, even though it is possible and would probably improve the models performance, the current volume of data is already low and it would further unbalance the classes and generate other problems for large-data models during training.

In short, the congruence between the Confusion Matrices of the models and the consistency of the miss-classifications hint that there are samples that are globally miss-classified across models, and actions over the dataset must be performed to correct this.This causes the models to find an asymptotic threshold at 90%, which is not passed due to the inherent uncertainty of the dataset, where high variance, multiple failures under one label, oil refills, overlapping classes and high class imbalance contribute to generate a very sub-optimal quality dataset. This hinders the learning abilities of data learning algorithms, and may not be resolved fully during training. Some models may be able to handle better the specific dataset's characteristics, but none will achieve perfect performance.

2.2 Model Analysis - F1 Score

As it is observed in the F1 Scores Comparison Table 22, the results are consistent and vary over an interval ranging from ($\sim$82%) to ($\sim$89%).

The interval's variation is explained by the principles used by each model to handle the 'bias versus variance' trade-off, where simpler models are more inclined to suffer from bias, such as the Decision Tree estimator, while models that are more powerful are inclined to suffer from variance, such as neural networks.

In other words bias is explained as simpler models might not be able to learn all the significant patterns to provide the best results, while variance can be explained as; the more

[30]Based on surface tolerances, materials, operation principle or operational conditions.

[31]Models evaluate the likelihood of a sample belonging to a certain class: for a boundary sample, the larger class will be deemed more likely

complex the model is, it will learn subtle variability patterns that are not necessarily useful and may interfere with its predictions.

Assuming the Bayesian Hyper-Parameter Optimization was successful, each of the models should be performing close to their best capability of minimizing 'bias vs variance' trade-off.

Given that the dataset's specific conditions in this case study is of very high variability, dimensionality, and it is not of optimal quality, the behavior observed is what is expected and supported by the theoretical background of the models; the models that are able to balance the 'bias vs. variance' trade-off better will be the best performing.

The extremes, like simple restricted models, such as Decision Tree (∼82%), and the more complex flexible models, such as Neural Networks (∼84%), will not obtain the best results overall, but models that have mechanisms to balance the 'bias vs variance' trade-off will have significantly better results, such as ensembles or gradient boosted state-of-the-art models. This is clearly depicted in the results of the F1 Score in Table 22.

The aforementioned is exactly what is appreciated in Table 22, where Random Forest (∼88%) and Extra Randomized Trees (∼86%) perform very similarly, only noticing a slight loss in performance of the second model due to the random node split criteria.

On the other hand, XGBoost (∼89%) and LGBM (∼88%) obtain similar performance, where the difference with ensemble models is mostly made by the sequential learning of the mistakes of the previous predictor, and the automatic regularization methods to aid in the minimization of over-fitting, allowing for a better handling of the 'bias vs. variance' trade-off.

The only dramatic difference amongst both of these models relies in the Training and Inference time, where LGBM sacrifices a tiny amount of performance for a significant amount of computational speedup and lightness. The main difference between both of these gradient boosted models, is that LGBM is built over principles aimed towards computational speed on big data settings, exceeding speed gains of 10x, or even 100x in some cases, over XGBoost with very similar performance.

Overall, for a task of this characteristics the difference is marginal, where the response does not have any urgency to be immediate, and there is a low amount of data. For the amount of data relevant to the case study presented, time concerns are not crucial, but is an important variable to be taken under consideration for extrapolation to more data heavy tasks. Still, as observed in Table 23, there is a x58 speed difference between XGBoost and LGBM with very similar performance, which in larger tasks may account for a significant amount of savings on computational resources and time.

Neural Networks are known not to handle optimally a high variance dataset, as their high flexibility favors the learning of many 'noise' patterns present in the data that do not contribute to make optimal quality predictions. In contrast with ensemble and gradient boosted, neural networks are considerably more powerful models, but do not have explicit mechanisms to handle high variance datasets, degrading their performance on this task. Also, gradient boosted models have a degree of interpretability by graphically outputting their decision process, which that neural networks lack, giving them the advantage head-to-head.

In the base case, deterministic neural networks and probabilistic neural networks share the same performance, meaning that there is no perceivable difference on the results obtained between the models.

Therefore, if the comparison was made upon solely deterministic models for a direct machine/deep learning solution, the Gradient Boosted algorithms would be the ones best suited for this task, with an exceptional $+\sim 4\%$ performance gain over neural networks and $+\sim 2\%$ over ensemble models, achieving almost 90% F1 Score.

Probabilistic Advantage

The main advantage of the probabilistic neural networks is the ability to output a measurement of uncertainty for each class prediction through the observed variability of the model collected class predictions of a specific input.

This prediction variability is obtained through the following process; instead of a learnt scalar parameter as in regular neural networks, there is a parametric distribution which is forward pass sampled one time by Monte Carlo sampling for the model prediction process to perform the matrix multiplications. The same input is passed through the model multiple times, where the variance contribution of each internal parametric distribution is captured into a measurable output after a certain number of passes, where it translates into a dispersion of the predicted values for each class through $Softmax$ function. When this collection of class outputs is obtained from sufficient samples of the same input, a median and standard deviation metrics are calculated for the prediction.

With this metrics, the median of all predictions is taken as the prediction certainty, while the standard deviation is taken as the confidence interval of the input. This aids on hinting the lack-of knowledge, or uncertainty, of a model over a specific prediction.

For example, if an outputted prediction has a very low standard deviation, it means that the learnt parametric distributions point the model consistently towards the same class prediction, implying that the $Softmax$ class probability outputs fall in a close vicinity; therefore, the prediction is confident of its result. On the contrary, with a high standard deviation, the class prediction probability has high variability, and the model lacks the knowledge to make a confident prediction on that class.

By pairing this mechanism with user-calibrated thresholds for the minimum median and maximum standard deviation admissible to consider a prediction reliable, a prediction filter is generated which can accept or reject predictions according to the user's desire for the model's 'minimum required knowledge'. In this case, the criteria selected for the case study is presented in Table 24, where the values were set by the median and standard deviation combination that caused the rejection of a portion of 25% of the Test Set.

As seen on Table 22, the Flipout Neural Networks F1 Score subjected to the filter criteria improves considerably, a steady +8% on both models, as all the samples that do not meet the determined knowledge criteria are deemed 'non reliable' due to the model's lack of knowledge, and separated from the 'reliable' prediction pool.

This is specially useful in a real-world context, where it allows for uncertain predictions to be rerouted to experts, while certain predictions are automated, increasing the impact of automation reliability when included in a decision making process.

When analyzing the samples rejected by the Flipout CNN1D on Table 25 that result in the Filtered Confusion Matrix presented on Figure 35, the first important thing to note is that the classes on the Test Set corresponding to 'Silica & ISO4406' and 'Silica & Wear' are the most affected by the lack-of-knowledge filter, where more than half of each samples are deemed 'unreliable' predictions.

The cause of this is the same as the reason why these labels are the most miss-classified classes by all the models. This classes are being affected by the fact of being the minority classes on the dataset, while also having to share core indicators with two of the larger classes. This generates that the most samples belonging to this classes will lie close to the decision boundary, being predicted as part of the larger class, or predicted with a high level of uncertainty. For most of the samples that are miss-classified after the filter, the model has a very high certainty that the samples belong to the larger classes.

In the case of the class 'Silica & Component Wear' the miss-classification towards 'Component Wear' it can be interpreted as a good fault classification in an operational-context, as the fault is still detected and closely related to the original fault. Thanks to the the warning provided by the model, in order to take corrective actions the sample will probably be re-analyzed by a human expert and confirmed.

Therefore, this miss-classification has less impact on the deployed model performance than the 'Silica & ISO4406' fault, which is miss-classified as 'Normal'. This can lead to more severe degradation of the equipment's on the long run, and even though the model identifies $\sim$60% of the faults, a large remaining portion is left without corrective actions. More samples regarding this fault are required for the model to learn a more robust separation.

The filter resulting on the Confusion Matrix on Figure 35 also causes the 3 remaining classes to increase their classification performance considerably, where if evaluated without the two problematic classes reach $\sim$97% of F1 Score. The filter also boosts the problematic classes, but given that these classes gravitate with high prediction certainty towards similar and larger classes, the remaining predictions after the filter are still grossly miss-classified.

This filter proves particularly useful to weed out any over-confident predictions that would be issued otherwise by a regular neural network or any other of the deterministic models reviewed in this case study, without a measurement of the model's knowledge over its own prediction. By separating the predictions via a filter, these predictions can be set apart in contrast with all the other models, proving an important advantage for an operational decision making process.

Eventually, as degradation is a continuous process, the classes that were miss-classified after the filter will become more severe and show indicators that will allow to classify the samples with more certainty into one of the classes allowing to take the required corrective action.

The latter it is not the optimal work flow for reliability engineering, but as more data is collected on the faults, the model's certainty over its predictions will continually grow as it identifies more specific class patterns increasing its classification performance. Eventually the filter will loose its overall impact on rejecting predictions related to known fault states, as with every new example in more data it will reaffirm its beliefs regarding each class.

For this specific case study, the probabilistic advantage of Bayesian Neural network versus regular neural networks relies on two main points: the ability to filter uncertain predictions gaining reliability for its inclusion in the decision making process , and to be more robust to over-fitting when in a situation of scarce data, providing better generalization capabilities to unseen data.

On the other hand, by comparing the best performing Flipout Convolutional 1D Neural Network to the best performing state-of-the art algorithm, XGBoost, the Flipout Neural Network barely has better performance than the gradient based model, considering it has rejected to predict 25% of the Test set. Performing the analysis under the premise that the most important miss-classifications are the ones of 'Silica & ISO4406' to 'Normal', comparing Figure 31 with Figure 35, this algorithm beats the Bayesian Neural Network's filtered Confusion Matrix in terms of lower false negative class identification.

If this was a machine learning task with perfect information, where all the class universe was known, the performance of XGBoost would certainly be considered superior to the Flipout Neural Network. Nevertheless, in real world conditions applicable to this specific Case Study, single time and infrequent events are a possibility that becomes very likely when the data comes from a large and wide range of different equipment subjected to 24/7 operation under variable load and external conditions such as extreme weather.

For this task, not all the fault classes are known, some of the faults have been left out due to insufficient data, and the 'Oil Contamination' class has been left out deliberately for evaluation purposes. The issue with deterministic models, is that an Out Of Distribution sample will be predicted as any of the classes it was shown during training, and have no way of handling these events in a reliable manner.

Here is where the prediction uncertainty filter becomes important and makes the difference when compared to deterministic models, by rejecting events where it lacks the knowledge to predict reliably.

Graphical Interpretation

Before discussing the Out of Distribution handling, An added advantage of the Bayesian Neural Networks is the ability of generating user-friendly graphical distribution plots, which show the median and standard deviation of each class predicted probability output in a simple and interpretable manner.

This is extremely helpful for information communication, explanation, and more intuitive than observing the raw metrics on a spreadsheet or report. This generates the opportunity to have a small degree of interpretability over the 'black box' that Neural Networks are in general. On an applied engineering or operational context, the graphical representation of

results has many upsides on the human-to-human interaction.

Also, several insights can be extracted from the distribution plots such as the ones exemplified in Figure 38 and figure 39 which are not trivial by observing the raw data metrics.

For simplicity, only two of the five class plots are shown[32]. In the first case, a correct prediction is presented along with the second most likely class prediction for the same input. Both exhibit a very low standard deviation, where the model is grouping the probability outputs in close vicinity meaning that the model is certain of the issued predictions.

On the second case, a wrong under-confident prediction is shown, where the true label is marked in green, and the predicted wrongly in red. In contrast to the first case, a very wide spread of the standard deviation is observed, implying that the model has no confidence of the issued prediction. Nevertheless, the prediction still narrowly falls within the maximum standard deviation allowed, but is rejected through the median filter. This specific case in the real world, if it was a regular neural network it would have issued the wrong prediction directly without any explanation or warning, while in the Bayesian Neural Network this sample would have been rerouted for expert analysis.

Another interesting interpretation for the graphical interpretation can be observed in Figure 40, specially for more robust models trained with larger volumes of data, is that the model could be right in its prediction, and the observed uncertainty could pertain to a transitional fault state that resides in the boundary between the two classes, or correspond to multiple faults identified on one sample.

This is a specially interesting concept useful for the identification of incipient faults, aiding in the preventive task management, planning and executing. Also, at the very least, it can provide a starting point to guide the human analyst to pay attention to certain fault indicators, expediting the oil analysis process saving time and resources.

2.3 Out of Distribution Simulation

The main advantage of the Flipout Neural Networks is the capability of applying the 'lack-of-knowledge' filter mentioned before, not to identify and refuse predictions classes it knows, as a robust model trained with an acceptable volume of data will most likely be certain of the issued predictions; but, to identify and refuse predictions on events that are boundary between classes, single-time or infrequent, rerouting the input for human analysis.

Different to deterministic models, that will issue a prediction based on the training data solely, the probabilistic model can handle OOD's generating a massive advantage for real-world deployment where the environment is not entirely controlled. The model will be subjected to singular/infrequent events during its deployment which need to be addressed reliably in order to trust the prediction on a decision making process where the outcomes are subjected to risk.

The 'Oil Contamination' class was separated from the training data in order to simulate OOD's that will happen during the model's deployment. This were then passed through the

[32]The complete prediction plots are in the Appendix II

trained model.

Given that the samples do not have a known label, they were set as 'Normal' class, given that not raising a fault warning (classification) to a sample would mean the normal operation of the machine until the manifestation of fault indicators.

The results for this simulated response to OOD's by the Flipout Convolutional 1D Neural Network is presented in Table 26, where the detail of the class predictions of the network before and after filter are observed. Here, 55.86% of the simulated OOD's were rejected due to uncertainty over the predictions, which equates to half the predictions being sent for human analysis in an operational context.

Out of the remaining 296 samples that passed through the 'lack-of-knowledge' filter, 246 (83.1%) of the predictions were classified as 'Normal' with an acceptable degree of certainty as seen in the example of Figure 41, while 50 were similarly miss-classified as another fault state.

This implies that either the fault was not severe enough to be differentiated enough of the 'Normal' class, which as degradation is a continuous process, eventually will show indicators that allow to diagnose a fault, or there were multiple faults on a same sample that lead it to be classified as other fault. This insights could not be provided by a deterministic model.

On the other hand, the graphical analysis presentation advantages are clearly experienced when analyzing Figure 42 and Figure 43, both of which show samples that did not meet the criteria of the user-set minimum 'knowledge' filter and were deemed unreliable. In both cases it is seen that the standard deviation of the class predictions is high.

In the first case, the model is certain the Label 1:'Silica & ISO4406' class is not the correct class with a 3% median and a standard deviation of 11%. But, it is observed that the median value for the Label 0:'Normal' class is close to 84%, while the confidence (standard deviation) is a $\pm$ 27% from the median, meaning that the spread falls outside the filter's thresholds and the sample is deemed unreliable and rejected.

Here the importance of having a measure for the variance of the class prediction probability as the model's uncertainty about a specific prediction is clear, the model lacked the knowledge to provide a confident answer that otherwise it would have been issued which was filtered through the inconsistency of the model's probability estimates for the specific class.

On the second case, the sample still has a large standard deviation which falls barely within the threshold set for it. But, differently from the last case, here the models does not have certainty about which class is the sample more likely to belong. Ergo, the median values predicted are blow the set threshold and the sample is rejected.

As it can be seen in both cases, the prediction spread is so wide it visually resembles a uniform distribution correctly depicting the model's lack of knowledge over this prediction, while the model's class predictions can also be analyzed for deeper insights on the sample and the model's decisions. This also highlights the importance of the graphical interpretation.

Section VII: Conclusion

Hyper-Parameter Bayesian Optimization was successful to find hyper-parameter sets that allowed to develop stable and close to optimal models that allowed the evaluation process to be pseudo independent of the model settings, obtaining valuable insights about the model performance according to their core attributes.

The dataset has to much variance for neural networks to be able to balance the 'bias versus variance' trade-off, hindering their learning potential. Convolutional Neural Networks performed better, as expected, due to their local feature extraction capabilities versus the Multi-Layer Perceptron. Ensemble models performed slightly better given their natural attributes for variance management and feature sub-sampling.

Bayesian Flipout Neural Networks had the same raw performance to their counterpart deterministic Neural Networks, with less than 0.03% for Convolutional and 1.2% for Fully-Connected. The main downside of Bayesian head-to-head is a larger number of trainable parameters and a massive increase on inference times, which on larger or time critical tasks could prove detrimental to their feasibility.

The model that overall had the best classification performance over the data in every way was the Extreme Gradient Boosted model with 88.85% F1 Score, followed by the Light Gradient Boosted Machine with marginally worse performance with 87.77% F1 Score, but training 58 times faster than XGBoost.

For maintenance real-world operation settings, the Bayesian Neural Network attribute to output a measure of uncertainty, and the posterior filter that can be calibrated to rule out the classifications where the model lacks the knowledge to make a reliable prediction, proves extremely useful. This positive externality makes the model very desirable for its use under uncertain conditions where the data is scarce and of sub-optimal quality, leading it to be a good contender to deterministic models that can not perform the filter or the graphical interpretation plots.

The main downsides of the Bayesian Neural Networks are the complexity of the training process, having more hyper-parameters than need to be optimized, double the model parameters and a few extra algorithmic steps, equating in considerably more training time. Also, as mentioned before, the inference time increases massively in comparison with regular neural networks due to the multiple sampling of the prediction, hindering their performance in time critical applications.

Compared to the other models, the filter is a useful tool that does not rely on 100% of automation, but rather making contributing to make the process easier and faster than what it is it is commonly to this day, freeing time an resources from expert assets.

As more and better quality data is available to train the Bayesian Neural Network models, by the law of large numbers[33], class predictions become more certain and the filter looses importance on an internal scale within the classification of the classes.

The issue with the deterministic models is that an external event might affect a sample (Out Of Distribution), which will be predicted as any of the classes it was shown during training, and have no way of handling these events in a reliable manner.

In contrast, the 'knowledge' filter can prove to be one of the key fortitude points of Bayesian Neural Networks to attack this point, being able to detect and handle singular/infrequent events by deeming the model's prediction unreliable and rerouting it to an expert for analysis.

The positive externality of the Bayesian Neural Networks capability to provide an OOD-robust model, with the added advantage of interpretation is one of the key points why this model should prevail for application in an real-world setting over the deterministic state-of-the-art XGBoost. The model is intended for deployment under uncertainties inherent to an uncontrolled environment, which should be addressed by any model deployed in order to grant enough reliability to be taken into account in any decision making processes.

In other words, the model is a very feasible alternative for deployment in real world settings, as it envelopes the performance of a neural network, with an extra layer of protection against over-confident predictions and out of distribution events. It may not have the best performance, but the positive externalities outweigh the performance gains by establishing a categorical trade-off between best performance and model robustness under uncertainty.

Even though the OOD filtering mechanism is not perfect by any means under high uncertainty, as the results of this work show, it is an extra layer that is not provided by any other scalable to 'Large/Big data' model to attack concerns that arise from applications in real-world and industrial settings. Overall, the results obtained show that the model knowledge filtering works even in sub-optimal[34] data conditions, and under better conditions it should improve significantly.

Finally, the Graphical Interpretation plots depicting the class predicted probability distributions can provide insights into the model predictions and expert interpretation about faults, being a very useful added feature for management and operational information flow.

In conclusion, this works shows that Bayesian Neural Networks are a feasible and desirable alternative for deployment under real-world settings for maintenance tasks, presenting several advantages over their deterministic counterparts, which might have better performance, but lack robustness against Out of Distribution events. This Bayesian Neural Networks, if

[33]Eventually, with enough data, any model will converge to the same result

[34]The dataset was assembled under the big assumption as if the combination of different equipment families under extremely different operational conditions should not change or impair the class recognition, introducing detrimental amounts of variance.

calibrated properly, have the potential of adding great value to the industrial maintenance decision processes as a tool to expedite machines diagnostic tasks, specially under uncertainty.

Bibliography

[Bayes, 1763] Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Soc. of London*, 53:370–418.

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight Uncertainty in Neural Networks. *arXiv e-prints*, page arXiv:1505.05424.

[Breiman, 1997] Breiman, L. (1997). Arcing the edge. Technical report.

[Chen and Guestrin, 2016a] Chen, T. and Guestrin, C. (2016a). Xgboost: A scalable tree boosting system. pages 785–794.

[Chen and Guestrin, 2016b] Chen, T. and Guestrin, C. (2016b). Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754.

[Coşkun et al., 2017] Coşkun, M., Uçar, A., yıldırım, , and demir, Y. (2017). Face recognition based on convolutional neural network.

[Ebersbach and Peng, 2013] Ebersbach, S. and Peng, Z. (2013). *Fault Diagnosis of Gearbox Based on Monitoring of Lubricants, Wear Debris, and Vibration*, pages 1059–1064. Springer US, Boston, MA.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139.

[Gal and Ghahramani, 2015] Gal, Y. and Ghahramani, Z. (2015). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *arXiv e-prints*, page arXiv:1506.02142.

[Geron, 2017] Geron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Sebastopol, CA.

[Geurts et al., 2006] Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63:3–42.

[Gilpin et al., 2018] Gilpin, L., Bau, D., Yuan, B., Bajwa, A., Specter, M., and Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning.

pages 80–89.

[Hajizadeh, 2019] Hajizadeh, Y. (2019). Machine learning in oil and gas; a swot analysis approach. *Journal of Petroleum Science and Engineering*, 176:661 – 663.

[Helbing and Ritter, 2018] Helbing, G. and Ritter, M. (2018). Deep learning for fault detection in wind turbines. *Renewable and Sustainable Energy Reviews*, 98:189 – 198.

[Hughes, 1968] Hughes, G. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

[ISO 4406:2018(E), 2018] ISO 4406:2018(E) (2018). Hydraulic fluid power — Fluids — Method for coding the level of contamination by solid particles. Standard, International Organization for Standardization, Geneva, CH.

[Jardine et al., 2006] Jardine, A. K., Lin, D., and Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483 – 1510.

[Jingwei et al., 2012] Jingwei, G., Niaoqin, H., Lehua, J., and Jianyi, F. (2012). A new condition monitoring and fault diagnosis method of engine based on spectrometric oil analysis. In Jiang, L., editor, *Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 19-20, 2011, Melbourne, Australia*, pages 117–124, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Jordan et al., 1999] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

[Ke et al., 2017a] Ke, G., Meng, Q., Finely, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017a). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30 (NIP 2017)*.

[Ke et al., 2017b] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017b). Lightgbm: A highly efficient gradient boosting decision tree. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc.

[Kingma et al., 2015] Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick.

[LeCun et al., 1989] LeCun, Boser, Denker, Henderson, Howard, Hubbard, and Jackel (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[LeCun et al., 1999] LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. (1999). *Object Recognition with Gradient-Based Learning*, pages 319–345. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Ma, 1993] Ma, L. (1993). Multivariate statistical analysis of spectrometric oil test data.

[Metropolis and Ulam, 1949] Metropolis, N. and Ulam, S. (1949). The monte carlo method. *J. Am. Stat. Assoc.*, 44:335.

[Miller, 2017] Miller, T. (2017). Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38.

[Mobley., 2002] Mobley., R. K. (2002). An introduction to predictive maintenance. pages 1–6, Woburn. Elsevier Science.

[Mohamed et al., 2017] Mohamed, S. M. K., Le, V., Lim, C. P., Nahavandi, S., Yen, L., Gallasch, G. E., Baker, S., Ludovici, D., Draper, N., and Wickramanayake, V. (2017). Condition monitoring of engine lubrication oil of military vehicles: a machine learning approach.

[Newell, 1999] Newell, G. (1999). *Oil analysis cost-effective machine condition monitoring technique*, pages 119–124.

[Park, 2013] Park, C. (2013). A feature selection method using hierarchical clustering. 8284:1–6.

[Phillips et al., 2015] Phillips, J., Cripps, E., Lau, J. W., and Hodkiewicz, M. (2015). Classifying machinery condition using oil samples and binary logistic regression. *Mechanical Systems and Signal Processing*, 60-61:316 – 325.

[Rosenblatt, 1957] Rosenblatt (1957). *The perceptron, a perceiving and recognizing automaton, Project Para.* Cornell Aeronautical Laboratory.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. , 323:533–536.

[Shridhar et al., 2018] Shridhar, K., Laumann, F., and Liwicki, M. (2018). Uncertainty estimations by softplus normalization in bayesian convolutional neural networks with variational inference.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

[Tin Kam Ho, 1995] Tin Kam Ho (1995). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1.

[Toms and Toms, 2010] Toms, A. and Toms, L. (2010). *Oil Analysis and Condition Moni-*

toring, pages 459–495.

[Wang et al., 2017] Wang, L., Zhang, Z., Long, H., Xu, J., and Liu, R. (2017). Wind turbine gearbox failure identification with deep neural networks. *IEEE Transactions on Industrial Informatics*, 13(3):1360–1368.

[Wen et al., 2018] Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. (2018). Flipout: Efficient pseudo-independent weight perturbations on mini-batches. In *International Conference on Learning Representations*.

[Wolpert, 1992] Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

[Zhao et al., 2018] Zhao, H., Liu, H., Hu, W., and Yan, X. (2018). Anomaly detection and fault analysis of wind turbine components based on deep learning network. *Renewable Energy*, 127:825 – 834.

Appendix

1 Appendix 1: Bayesian Hyper-parameter Optimization Process & Training Plots

1.1 Decision Tree

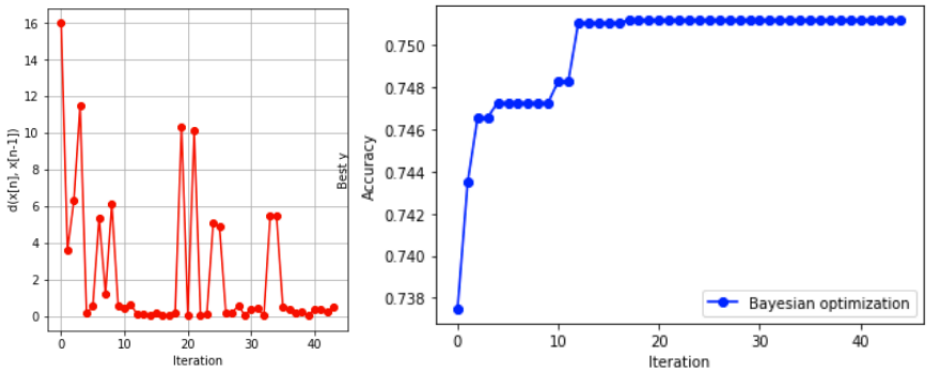

Figure 44: Decision Tree - Geometrical Distance Between Samples(left) & Score Result(right)

1.2 Random Forest

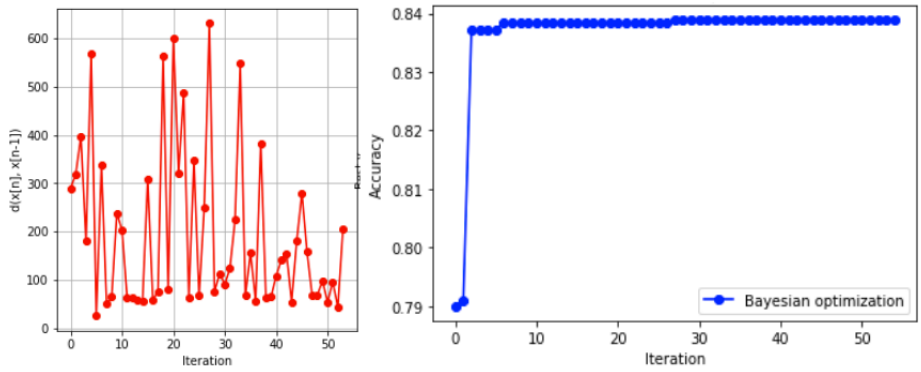

Figure 45: Random Forest - Geometrical Distance Between Samples(left) & Score Result(right)

1.3 Extra Randomized Trees

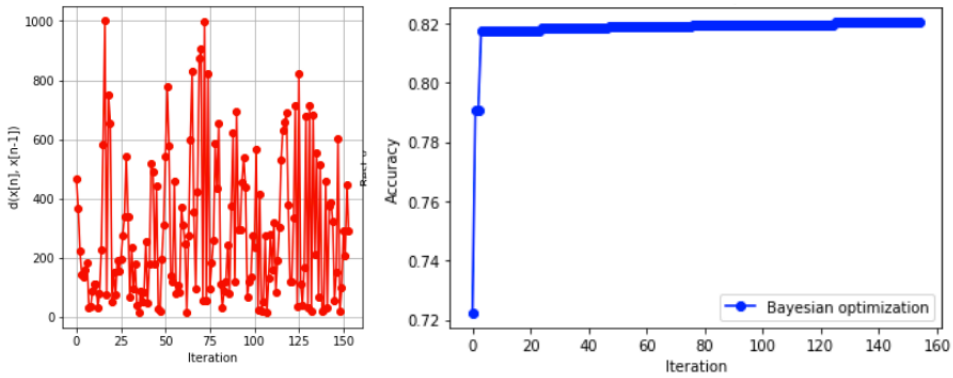

Figure 46: ERT - Geometrical Distance Between Samples(left) & Score Result(right)

1.4 Extra Gradient Boosting - XGBoost

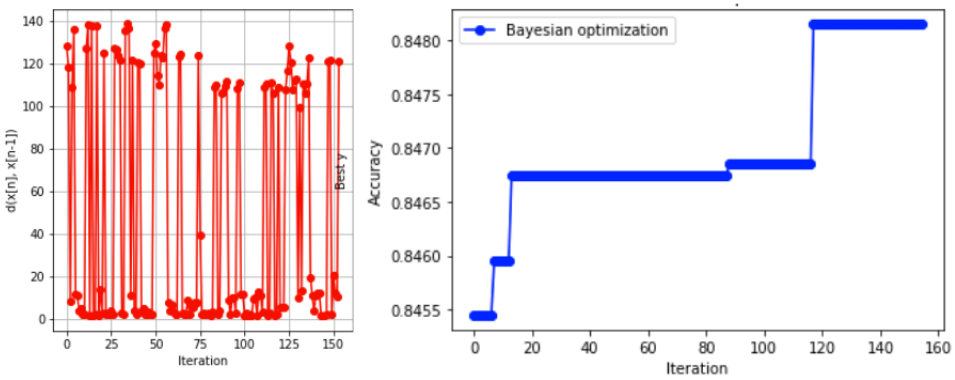

Figure 47: XGBoost - Geometrical Distance Between Samples(left) & Score Result(right)

1.5 Light Gradient Boosting - LGBM

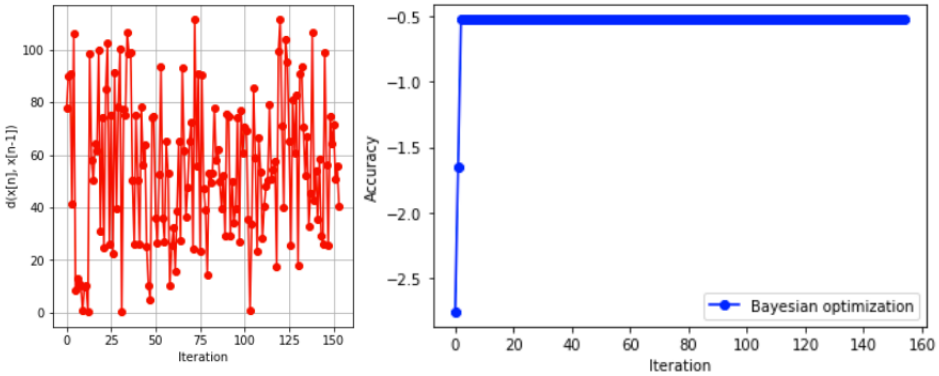

Figure 48: LGBM - Geometrical Distance Between Samples(left) & Score Result(right)

1.6 Neural Network - Multi Layer Perceptron

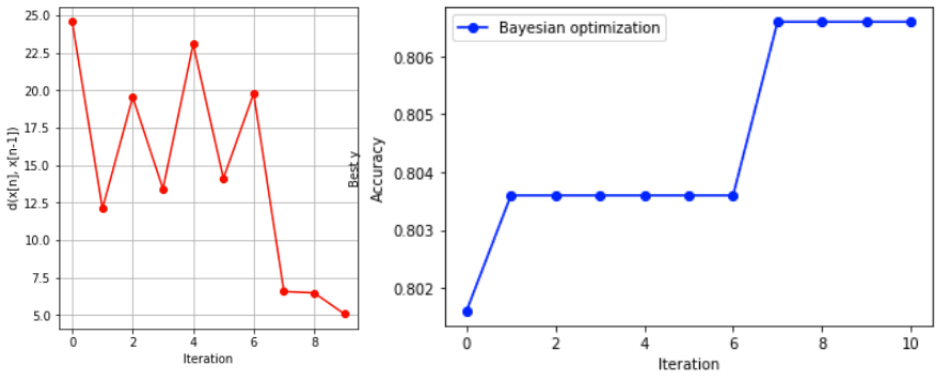

Figure 49: MLP - Geometrical Distance Between Samples(left) & Score Result(right)

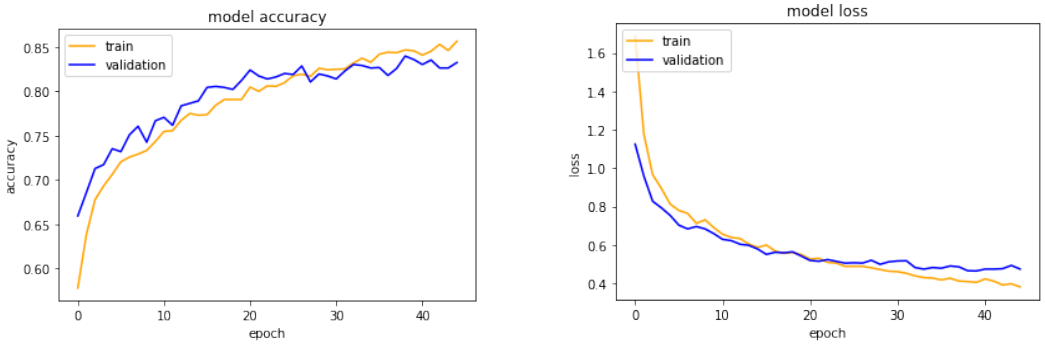

Figure 50: Train and Validation Curves MLP - Accuracy (left) and Loss (right)

1.7 Convolutional 1D Neural Network

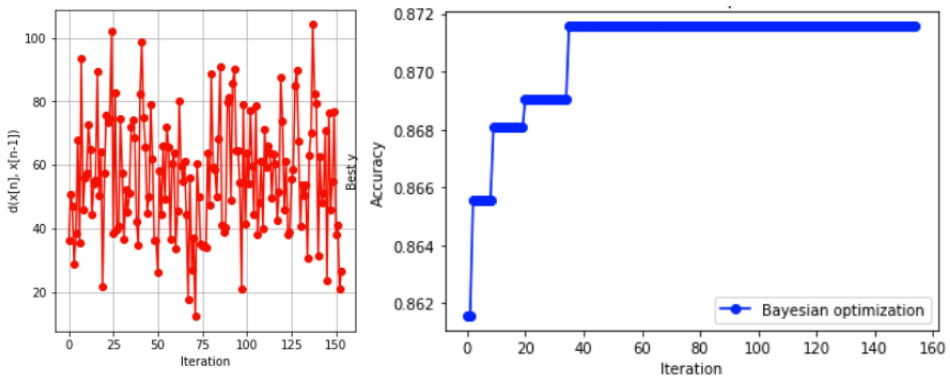

Figure 51: CNN1D - Geometrical Distance Between Samples(left) & Score Result(right)

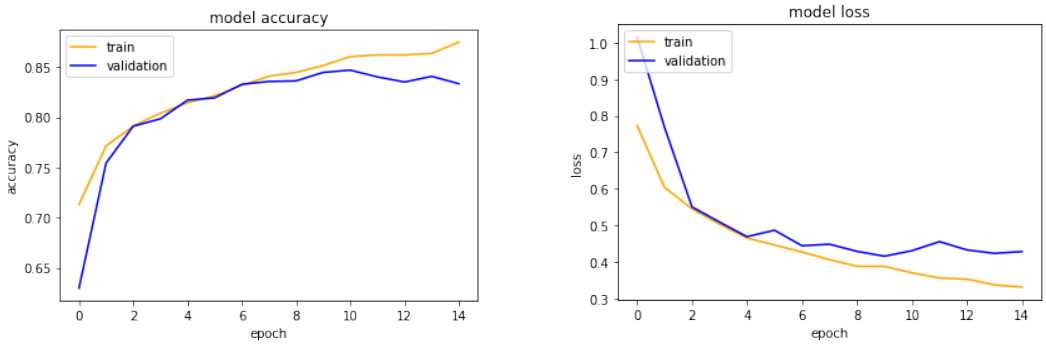

Figure 52: Train and Validation Curves CNN1D - Accuracy (left) and Loss (right)

1.8 Flipout Multi Layer Perceptron Neural Network

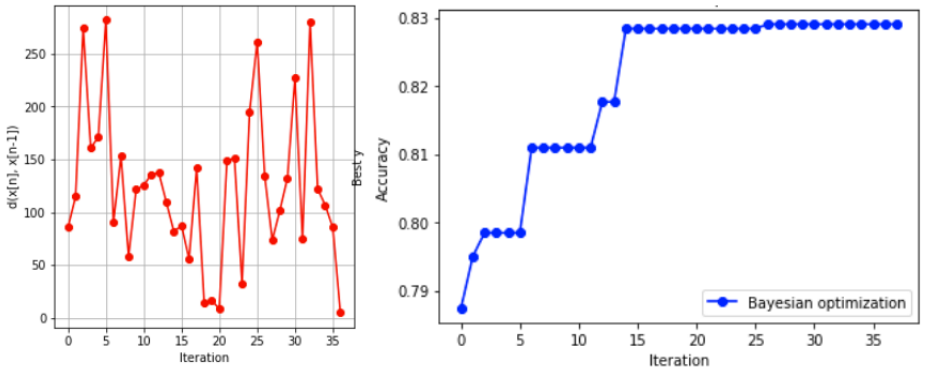

Figure 53: FMLP - Geometrical Distance Between Samples(left) & Score Result(right)

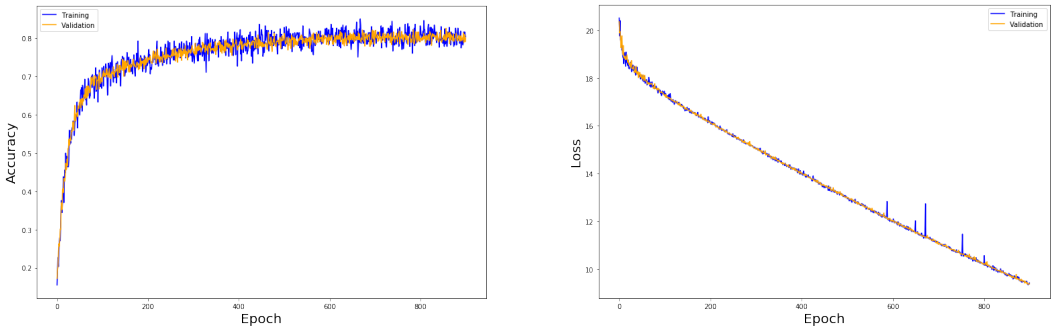

Figure 54: Train and Validation Curves FMLP - Accuracy (left) and Loss (right)

1.9 Flipout CNN1D Neural Network

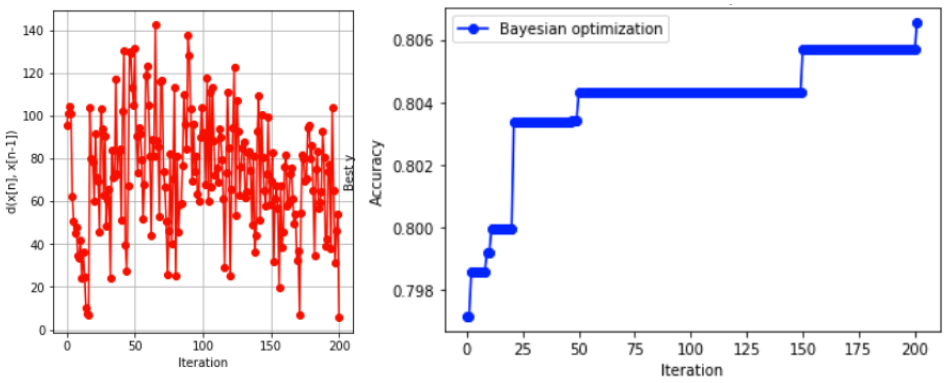

Figure 55: FCNN1D - Geometrical Distance Between Samples(left) & Score Result(right)

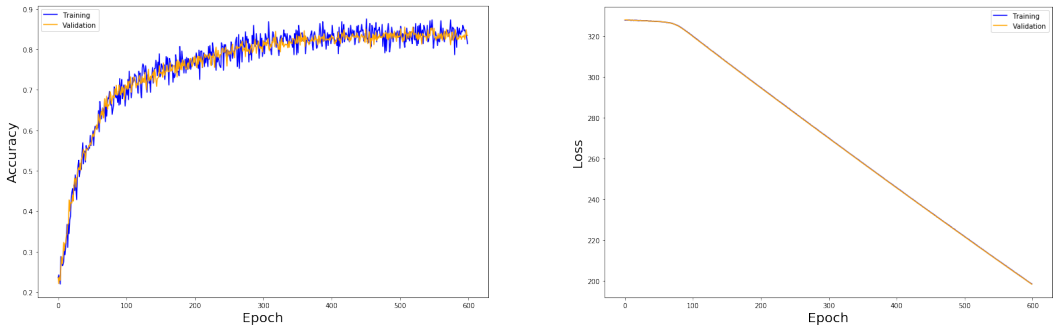

Figure 56: Train and Validation Curves FCNN1D - Accuracy (left) and Loss (right)

2 Appendix 2: Full Graphical Plots of Example Samples - Flipout CNN1D

2.1 Confident Prediction Example

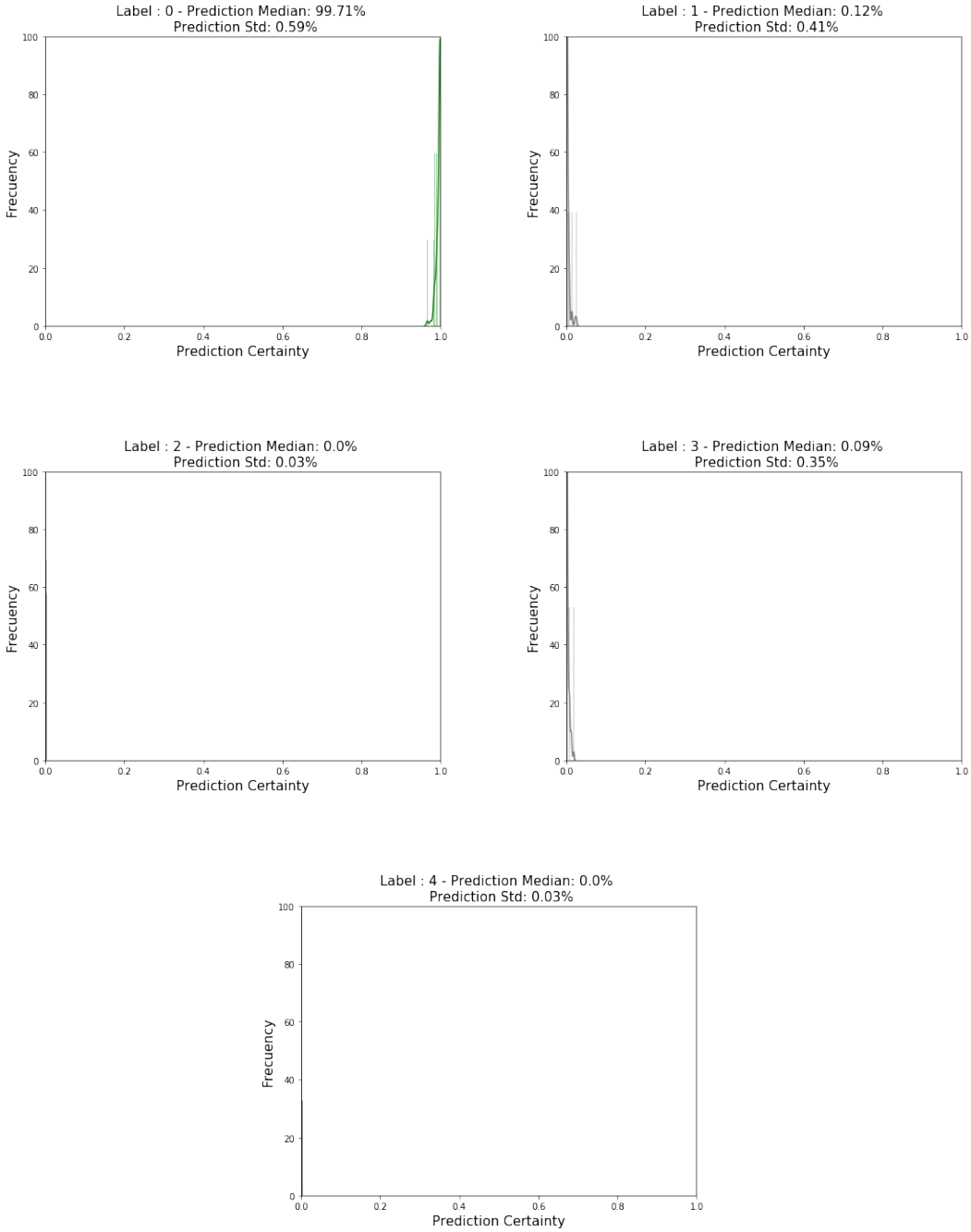

Figure 57: Confident Prediction - Full Graphical Class Prediction Plot

2.2 Wrong Under-confident Prediction Example - Median Rejected

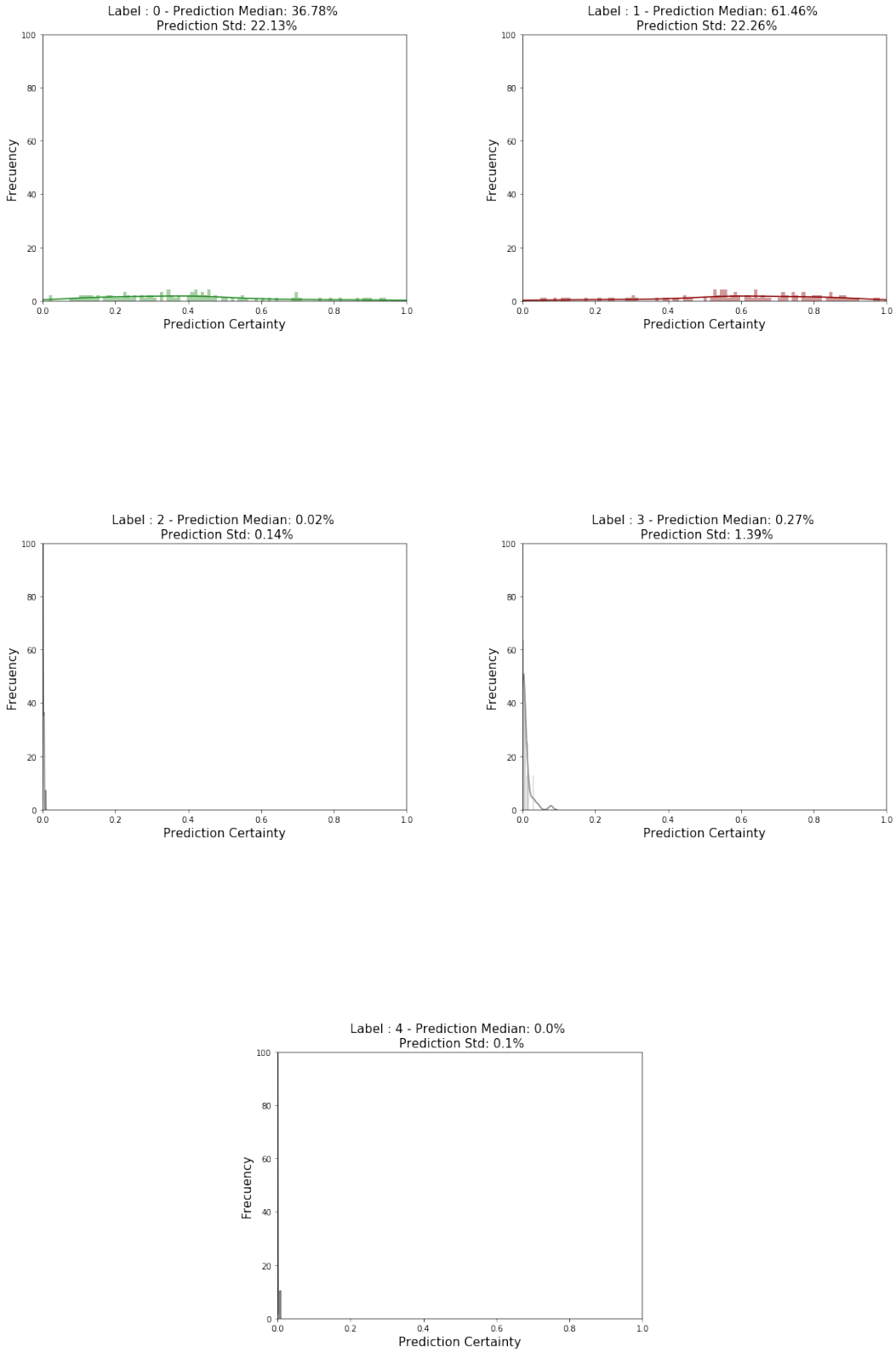

Figure 58: Wrong Under-confident Prediction - Full Graphical Class Prediction Plot

2.3 Multiple Fault or Transition Fault State Identification

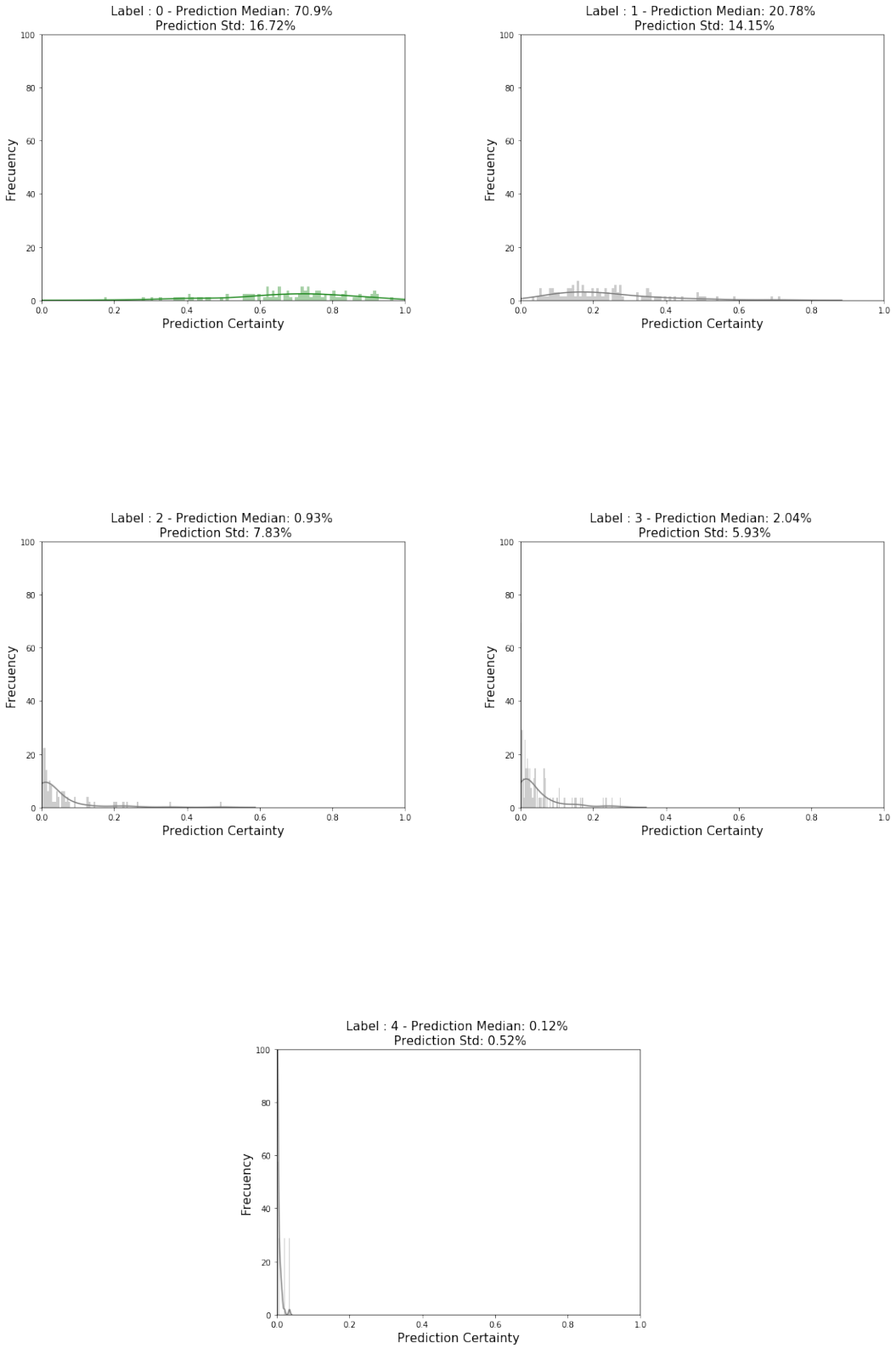

Figure 59: Multiple Fault/Transition Fault State - Full Graphical Class Prediction Plot

2.4 Accepted Simulated OOD

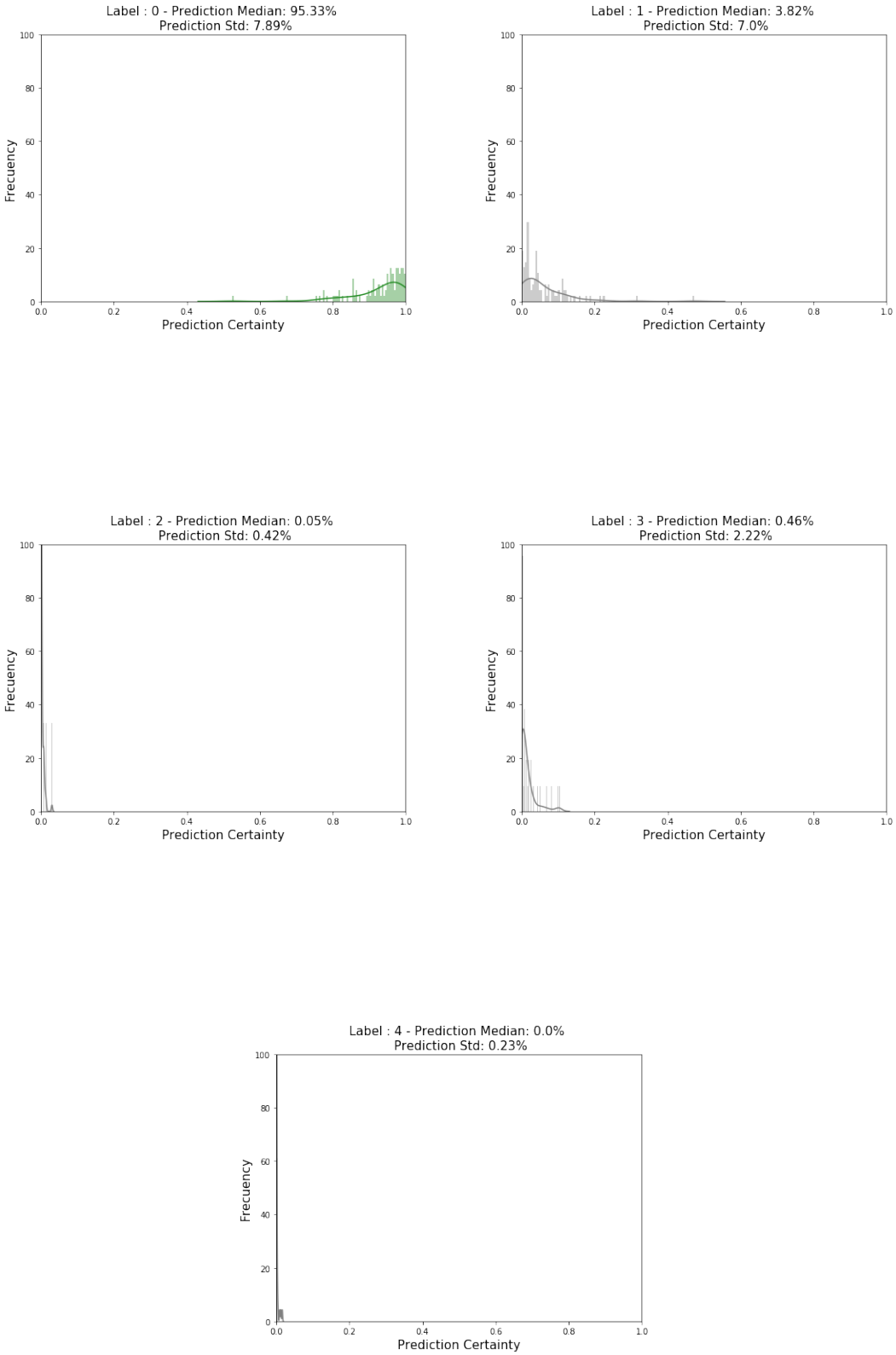

Figure 60: Accepted Simulated OOD - Full Graphical Class Prediction Plot

2.5 Rejected Simulated OOD - STD Rejected (Over-confident)

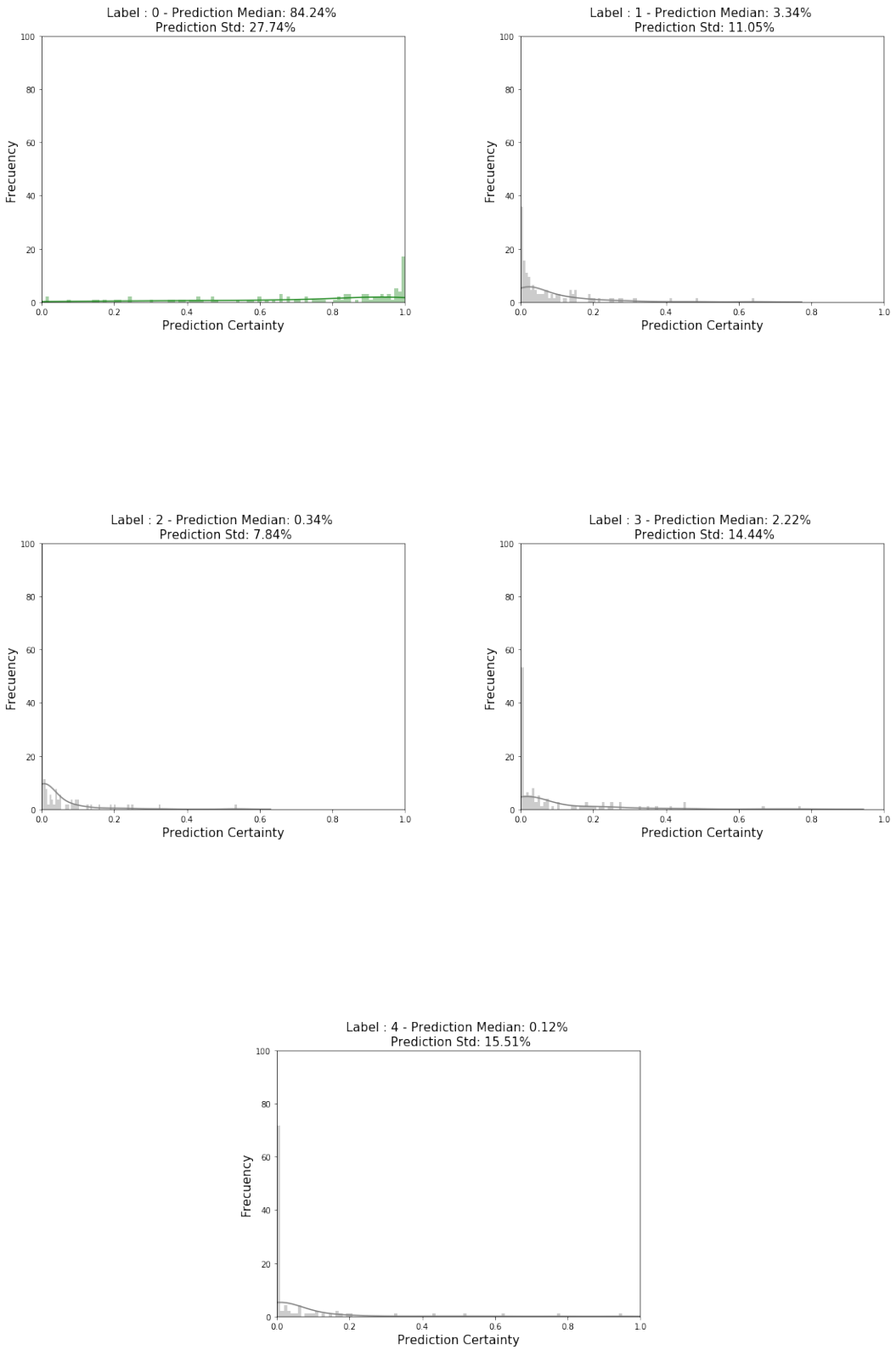

Figure 61: Rejected Simulated OOD STD - Full Graphical Class Prediction Plot

2.6 Rejected Simulated OOD - Median Rejected (Over-confident)

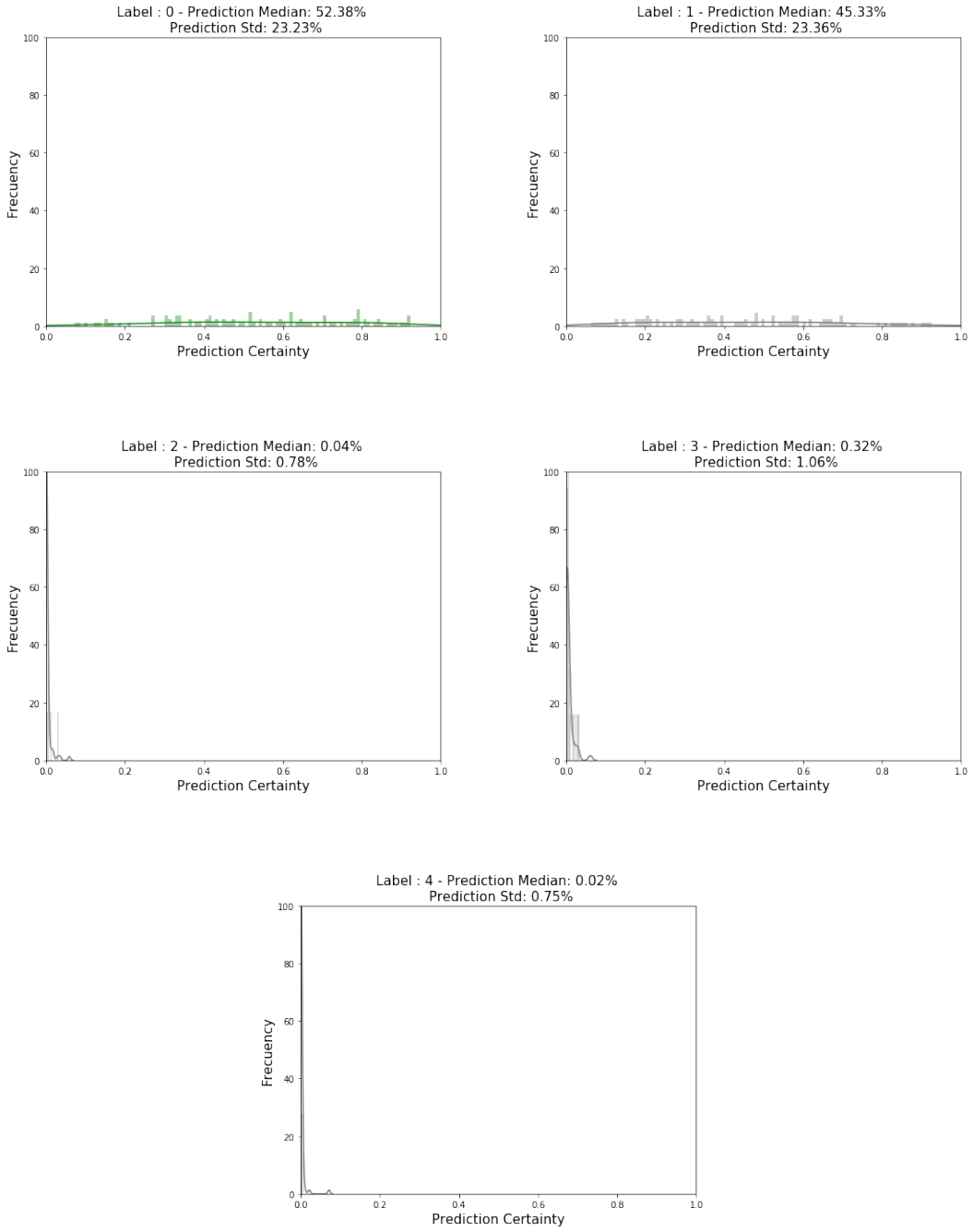

Figure 62: Rejected Simulated OOD Median - Full Graphical Class Prediction Plot