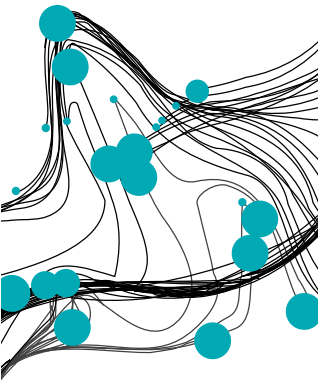
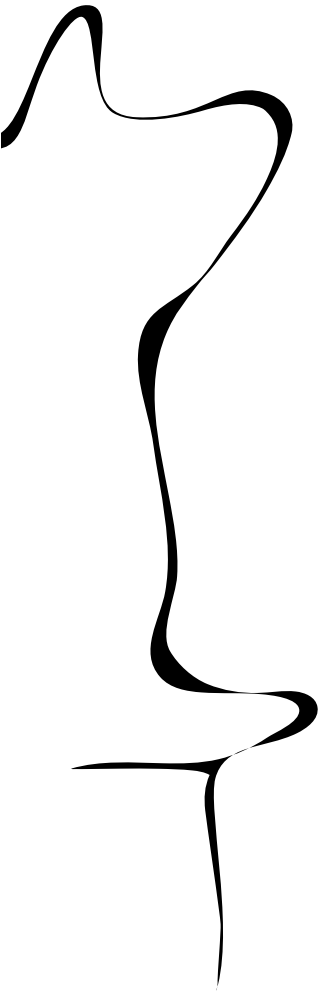


DMB

DATABASE MANAGEMENT
AND
BIOMETRICS

.32744



TOWARDS DEEP LEARNING FRAMEWORKS FOR THE ANALYSIS OF MAGNETIC FLUX LEAKAGE CAPTURES

Nicolas Stoian

MASTER'S GRADUATION THESIS
ASSIGNMENT

Committee:
Estefanía Talavera-Martínez
Nicola Strisciuglio
Le Viet Duc

December, 2022

2022DMB0007
Data Management and Biometrics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Towards deep learning frameworks for the analysis of magnetic flux leakage captures

Nicolas Stoian

December 18, 2022

***Abstract*—Non-destructive testing (NDT) is an industrial analysis technique used to evaluate the characteristics of a material, component or structure without causing any type of damage to it. In the petrochemical industry, one of the most important NDT methods is magnetic flux leakage (MFL), a method which uses the magnetic properties of pipes and storage tanks to detect defects in them.**

While the use of machine learning is not a new idea in the field of MFL inspection, newer techniques such as deep learning have only recently started being applied. This approach is supported by the massive amounts of data that is produced when MFL scans are performed. While the method being used is the same for pipelines and storage tanks, the resulting data is very different between the two mediums, which means that the machine learning methods used on one of them will not translate well to the other. At the moment, most of the research done into using machine learning technique focuses on pipelines alone, with little to no effort being put into combining MFL inspection for tank bottoms and deep learning.

Machine learning efforts are usually split into two parts in such contexts: a detection algorithm and a classification algorithm. ROSEN already has a detection algorithm and a deep learning model deployed as part of its MFL scanning tool, the TBIT, however the results obtained so far tend to differ by a large margin from scan to scan. In this case, the project has focused towards improving the second component, the classification model.

For the purposes of this project, ROSEN has provided 135000 entries split over 20 datasets captured from 10 different storage tanks scanned over a number of years. This thesis has a two-fold contribution: the first is to try to identify the potential causes behind the discrepancy in the results of the baseline deep learning classification model between the different datasets, and the second is to test multiple generalization techniques in order to decrease the generalization error of the model. This thesis also proposes two new novel generalization techniques. The first one is a new training schema which combines attention-based algorithms such as GRAD-CAM and HiRes-CAM and occlusion algorithms during the training phase of a deep learning model, which creates a new augmentation tool for datasets used in deep learning problems. The other is a novelty approach to loss functions, which attempts to improve already-existent loss functions by adding two new parameters to them, a softening margin and a stochastic variance parameter.

I. Introduction

Non-destructive testing (NDT) is the use of various analysis techniques in order to evaluate the properties of materials, components or systems of components in order to find anomalies, differences or discontinuities without harming the usability of the item being tested [1]. These types of tests are advantageous for ensuring the integrity and reliability of an item during and after fabrication or while in service.

Non-destructive testing (NDT) is a very broad field, which is subdivided into methods, each with a different scientific principle behind it, that are themselves subdivided into techniques, different implementations of the same principle. The most well-known and also the first example of NDT is visual testing (VT). Other popular methods are Magnetic Particle Testing (MT), Liquid Penetrant Testing (PT), Ultrasonic Testing (UT), and Electromagnetic Testing (ET) [2].

The method used in this project is Electromagnetic Testing, which is a measurement method for steel and other ferromagnetic materials. It uses electric currents and magnetic fields in order to create an electromagnetic response which can then be measured. If there are fluctuations in the measurements, it indicates that an anomaly has been detected. The two MFL techniques used in this project are Magnetic Flux Leakage detection (MFL) and Eddy Current testing (ECT).

In the petrochemical industry, NDT techniques are used to detect possible anomalies in the transport and storage infrastructure. Most commonly this type of analysis is done on oil pipelines using specifically designed equipment. Although not a recent development, the same techniques have also been applied to storage tank floors with a high degree of success [3]. However, there is still some delay in the development of the technologies between the two applications. In the last two decades, pipe inspection tools have started being improved with the addition of machine learning techniques to their process flows, which range from classical methods, such as SVMs [4] to various types of neural network architectures [5] [6]. For storage tank bottoms, this type of framework is still very limited in usage at the moment.

A. Problem

The usual way of approaching defect detection in magnetic flux leakage captures is to build a framework out of a detection algorithm and a classification algorithm. The detection algorithm is a technology that is already established within ROSEN, and as such it is out of the scope of this project. For the classification part, the company also already has a neural network model deployed, which classifies the possible anomalies identified by the detection algorithm into true positives (called ”true calls” for the purposes of this project) and false positives (also called ”false calls”). Its architecture can be seen in Figure 14. However, the problem with the current system is that the model has a high generalization error rate, as it can be seen in Figure 1. While for the validation phase, the model

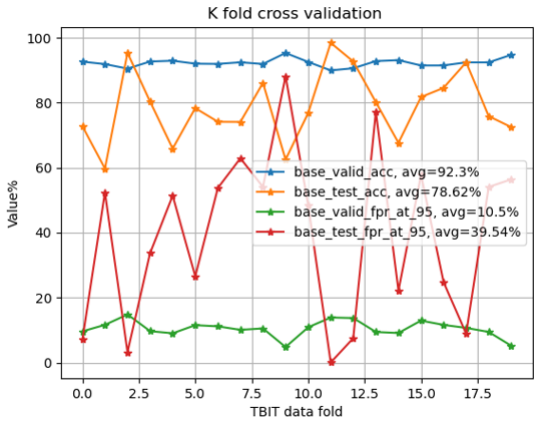


Figure 1: Performance of the baseline model during 20-fold cross validation. The blue line shows the accuracy of the model for each of the 20 validation datasets, the orange line shows the accuracy on each of the test datasets. The green line shows the FPR-95% recall rate for each of the validation datasets, while the red line shows the FPR-95% recall rate for each of the test datasets

is performing in a stable manner over all the datasets, the results when applied on the training set vary by a large margin.

B. Aim

The are two main goals for the thesis:

1) to identify the source of the discrepancy in the performance of the model across various datasets through root cause analysis, and
2) and to improve the existing model by testing a number of generalization techniques, some of which will be novel approaches

For the first goal, the research questions were the following:

What are the main causes behind the discrepancy in results between different datasets when the same model is used?

How can the discrepancies in properties between the datasets be minimized?

For the second goal, the research question is:

How can deep learning models trained on images obtained of MFL scans be improved in order to generalize better?

C. Method

For the first stated goal, the initial step is the identification of the issues with the baseline approach, this thesis used root cause analysis, a risk assessment method from the field of industrial process management [7], in order to identify the core issues that cause the discrepancies presented earlier. The second step is then to propose solutions that can fix them systematically, if such solutions are feasible within the context of the project. The chosen method for the first step was misclassification analysis, in which the instances that were labeled wrongly by the baseline model were checked for possible patterns. After that, a licensed operator re-checked a number of misclassified instance and then was asked to give an opinion on a number of factors regarding potential issues with the data.

The data that was used in the experiments was provided by ROSEN, and was gathered by multiple field operators over a number of years, using multiple MFL sensors on 10 storage tanks. For each of the tanks, the measurements were done using two different calibration standards, which resulted in 20 usable datasets. The resulting scans were then checked by a licensed operator, which meant that the data was of a high enough standard as to be used for supervised deep learning.

The goal is then to use the results obtained from it as a launching point for the second stated goal. The proposed framework is to first try to identify the issues that affected the results of the the baseline model, which will then guide the implementation for the second stated goal, where a variety of generalization methods for deep neural networks applied to the field of magnetic flux leakage detection are implemented and tested as an ablation study. A diagram of the areas of the network where generalization methods were applied can be seen in Figure 4. There were two main areas that were the focus of the research: the data and various ways to augment it, and improvements to the model in use.

In the first area, two types of techniques were tested: balancing techniques, in order to address the imbalances on the type and amount of data between the datasets used in this project, and data augmentation techniques, which increase the amount of available data by generating new data based on already-existing data.

For the second part, four types of methods were chosen. The first one was previously proposed by [8]. Attention based algorithms use various techniques to show how deep learning models make decisions. In the experiments, the algorithms were combined with occlusion algorithms. The idea in this case is that by combining the two algorithms, the decisions of the algorithm can be changed by first finding out what the algorithm sees as important in an image, and if the area of image the algorithm sees as important is problematic, it gets occluded and the algorithm is trained again, this time on the occluded image, in order to force it to switch its focus to another part of the image. For the visualization part, two algorithms were used: GRAD-CAM [9] and HiRes-CAM [10].

Stochastic regularization is another generalization method where multiple networks are simulated, each with its own unique architecture, by randomly dropping a percentage of the weights in a network during each training step. Two methods are used in this case: Dropout, which randomly drops entire neurons from the network [11], and dropConnect, which drops individual weights [12].

Soft labeling is a method through which the network can reduce overfitting by asking it to draw a decision boundary based on a label which has a probability label attached to it [13]. This way, the network can gain some leeway when drawing the boundaries between classes. This technique is tested by itself in this thesis, but it is also combined with loss functions in a novel manner.

D. Contributions of this work

This thesis has the following adds the contributions to existing research:

- on the one hand it provides an analysis of the issues that surround the application of deep learning techniques to classifying MFL scans obtained from tank bottoms,
- secondly, this thesis tested various generalization techniques for convolutional neural networks that train on images created by magnetic flux leakage sensors
- to that end, various generalization techniques are applied and discussed and novel improvements to those techniques were also proposed and tested
- the first one is a new training schema which uses saliency maps obtained from attention-based algorithms in order to change the areas of perceived importance for the network by occluding the areas of high importance with various types of noise
- the second one is a new type of loss function, which integrates a softening margin inside the function itself, instead of applying it to the labels of the data

E. Organization of the paper

Section II of this thesis describes the background knowledge necessary for understanding the technologies and the processes behind the data that is used in the project. Section III presents related research. Section IV describes the proposed methods used in the experiments. Section V describes the datasets used in the project. Section VI discusses the experiments run in relation to the first goal of the project, root cause analysis. Section VII presents the setup of the experiments, the metrics that were used to measure their results, and tested alternatives to the methods in section IV. Section VIII presents the results of the experiments. Section IX provides in-detail discussions for a number of experiments. Section X presents the final conclusions, and section XI presents directions for future research.

II. Background

A. Magnetic Flux Leakage

Magnetic flux leakage (MFL), also known as Transverse Field Inspection (TFI), is an NDT method that detects defects, such as corrosion. The field of application of these inspection techniques is in pipelines most commonly, but also in storage tanks. The method works by using a strong permanent magnet, which induces an elliptical magnetic field in the steel parts of the measured item. In the case of storage tanks it is used on the plates forming the floor of the tank. In the areas where the metal is damaged, a distortion of the base magnetic field appears, which is called a "leak". Typically, the larger the size of the defect, the bigger the leakage from the magnetic field will be.

MFL detection tools have two main components: a powerful permanent magnet combination which magnetizes the metal, and Hall-effect sensors between the poles of the magnet which measure the amount of magnetic flux leakage that comes from the to be inspected object. A schema of how MFL tools can be seen in Figure 2.

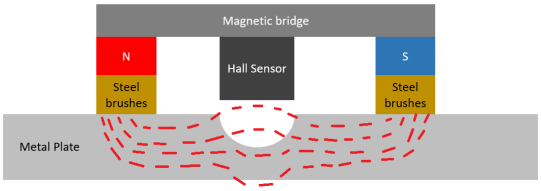


Figure 2: The principle of Magnetic Flux Leakage

The magnet in an MFL tool is designed to create a uniform signal throughout the metal and to be powerful enough to create a measurable amount of magnetic flux leakage. The leakage happens when there is enough magnetic flux to produce magnetic saturation, that is, when any further increase in the magnetization field applied to an object no longer produces any significant further increase in the magnetic flux density. The saturation point for any given object depends on the material it consists of and the thickness of the involved object [14]. The magnetization and permeability curves for sheet steel, the material typically used for tank bottoms, can be seen in Figure 3.

A second important factor is the permeability μ of the material to be inspected, which is the ability of magnetic flux to be induced into a material. Each material has its own permeability curve. Typically, at 0 magnetization, the curve starts linearly, but flattens quite significantly when saturation occurs. When the magnetization level is high, a metal loss defect (corrosion, cracks, holes, etc.) will create a reduction in permeability, which along with a local reduction in metal plate thickness, lowering the point of saturation even further, will make the magnetic flux to flow outside of the metal, which creates the magnetic flux leakage phenomenon.

The Hall effect is a method of measuring magnetic fields. It appears when electrical current flows through a metal plate, and then a magnet is placed perpendicular to the plate. The magnetic field then disturbs the flow of the current, by separating the positive and the negative current to opposite sides of the plate. This separation creates a small voltage difference between the sides of the plate which can then be measured. The appearance of this voltage difference is what constitutes the Hall Effect. The effect is measured in volts (V), but the effect is usually very weak, so in practice it is measured in μV.

B. Eddy Currents

One of the disadvantages of NDT techniques is that they are indirect measurements. That means that specific attributes of the defects or the relationships between them cannot be identified only through those techniques. In the field, this often results in the situation that, when using MFL on anomalies, several object parameters have an influence on the final measurements and some not at all, but they all show up in the measurement of an object [15]. One of the important parameters that suffers from

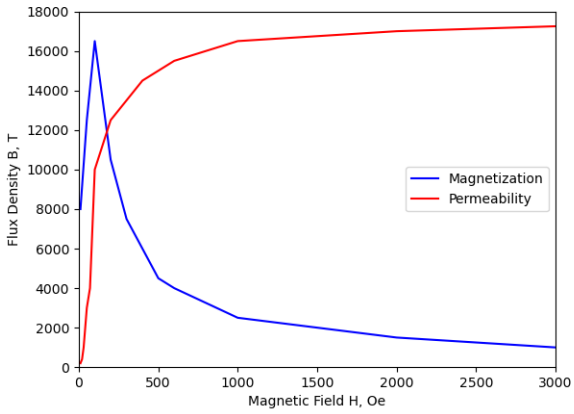


Figure 3: Magnetization Curves for sheet steel, the most common material for storage tanks

this issue is the localization in depth of an anomaly (i.e. is the anomaly on the inside side of the object being scanned, the outside, or inside the object itself in the case of stress fractures). In order to solve this problem, Eddy current sensors are used.

Eddy currents are circular electrical currents induced within a coil of conductor wire when the coil becomes excited with an alternating electrical current. This excitation creates an alternating magnetic field around the coil. When the coil gets close to a conductive material, in this case the steel plates of a tank bottom, eddy currents are induced in the material which, as per definition, rotate in the opposite direction to those in the coil. Variations in the electrical conductivity of the material being measured can then be detected by Eddy Current sensors. Furthermore, by looking at the phase and the amplitude, the distance to the defect can be calculated, which can help to understand whether the metal loss defect is on the inside or the outside of the object being scanned.

C. Signal Analysis

In this subsection the signal analysis of both magnetic flux leakage as well as the eddy current sensors is discussed. Within ROSEN , the combination of MFL and EC techniques is used for defect detection in two closely-related applications: in pipelines and in tank bottoms. The principles are the same for both, the only major difference is in the configuration of the equipment.

Generally, a defect identified by a MFL-EC detector combination has three main significant properties: width, length and metal loss percentage (ML %). There are many other measurements that can be obtained, such as closeness to other defects, orientation of defect in respect to the magnetic field, etc.

The metal loss, also known as the depth, represents the amount of lost material as a percentage of the total thickness of the plate, and is the factor most closely related to the amplitude of an MFL signal. It is measured as the percentage of the depth of the defect over the normal wall thickness. Thus, there is no universal measurement of depth, and its effects differ from case to case. This measurement is very important as higher metal depth requires more attention.

The width and the length of the defect also affect detection, defining here the width respectively the length as the defect's size orthogonal to respectively in line with the applied main magnetic flux. When talking about the width, the relationship between it and flux leakage is correlated, with less anomaly width resulting in less leakage. However, for length the opposite holds true, which is one of the biggest problems of the MFL technique. A lengthier defect spreads the flux leakage along its entire length, which leads to lower flux leakage values. This makes classification tasks problematic and requires additional processing in order to correct the effect.

The measurement tool used by ROSEN is the TBIT [16], a device which uses both Hall sensors and Eddy Current sensors. There are two types of detection related service types employed by ROSEN for the TBIT, one is called Standard and one called Ultra. Most of the differences between the two standards are outside the scope of this project. The only exception is the sensitivity factor: the Standard method does not capture anomalies that are under 20% metal loss, while the Ultra method allows for captures at 10% metal loss and above.

The device is equipped with multiple sensors, each with a different axial position in order to increase the chances of anomaly detection as described above. The ones used in this project are the horizontal hall sensor (abbreviated as PHH) and the vertical hall sensor (abbreviated as PVH). The horizontal sensor measures flux leakage as seen from the surface of the steel plate, while the vertical sensor measures the flux leakage as seen in the depth of the plate.

Plates are usually large in size and require multiple partly overlapping runs in order to be properly inspected. For that reason, the inspection of a plate is split into multiple sections, which are called stripes in this context.

Next up, an algorithm takes the sensor data and detects possible anomalies (peaks in the data above a certain threshold) caused by flux leakage by looking at the signal readings obtained from the sensors. Due to interactions between the factors discussed above and others, false positives and false negatives are a common problem in MFL detecting systems. There are also anomalies that do not threaten the integrity of the object in any way, but are still detected by the sensors, such as welds, fluctuations in the thickness of the metal or joints which connect metal plates.

The anomalies found by the detected algorithm are then sent to a trained specialist which labels them as either true positives or true negatives. True positives, more commonly known as true calls in these types of projects, are anomalies identified by the sensors which are threatening to the integrity of the object being scanned. The most common types of true positives are metal loss features (such as corrosion), dents, holes and gouges. False positives, also called false calls, are detections by the algorithm which come from either the non-threatening features described above or from natural random fluctuations in the magnetic fields that is being measured.

There are multiple ways to work with the captured anomalies for classification purposes: most research is

done on working directly with the signals captured from the anomalies and their data related properties, which can be analyzed using various signal-based techniques. However, this project normalizes the signals from the regions highlighted by the detection algorithm, re-scales the detected areas of interest to a standard size, and then uses the obtained values as pixel intensity values, turning them into images, where the values from each sensor constitute a channel and where the values from several channels of the same type form an image. Applying this technique allows the project to also use image analysis techniques in addition to more classical approaches.

III. Related Works

As discussed in the previous sections, there are two parts to the analysis of MFL signals: the detection algorithm and the classification algorithm. A large of the papers that apply machine learning techniques focus most of their research on the detection side [17] [18] [19] [6], while only employing more traditional classifiers for the second part. The other papers focus only on the classification part, working with already processed data and employing more complex machine learning methods.

On the classification side, three types of classifiers were popular: neural networks [5] [4] [6] [20], support vector machines (SVMs) [17] [18] and random forests [21] [19].

On the neural networks side, there were a multitude of approaches. [4] investigated a number of training functions, along with different network configurations, on noiseless and noisy data obtained from a number of pipelines using MFL signals. The optimum model found that for both the noiseless and the highest level of noise (10dB), a Polak-Ribière Conjugate Gradient had the best results, with an RMSE (Root Mean Score Error) of 1.39% at 0 noise and 9.47% at 10dB noise level.

An approach by [20] used an ELM (Extreme Learning Machines) Classifier, a technique where single/multiple layers of hidden nodes have fixed parameters and only the output weights are tuned during training. The technique was applied on MFL signal data in order to classify signals as benign or injurious, and had 95.63% average accuracy.

Another possible usage of machine learning in the domain of MFL is the use of semi-supervised learning. This type of research is important as MFL data is produced in large quantities, but correctly labeling findings requires human operators, which makes labeling a time-consuming procedure. [5] used a novel design for a Fuzzy Min–Max Neural Network (a type of semi-supervised neural network) on a variety of types of data, among which was also MFL data from pipeline scans. Their network obtained a performance of over 85% with only 40% of the training dataset being labeled.

Related to the issue of the large amount of data and neural networks is an approach based on big data techniques, as presented by [6], which used various machine learning and big data techniques in their paper and tried to detect defects in oil and gas pipelines and estimate the depth and size of the defects.

Their paper also presented a discussion on activation functions for neural networks, and the final results were compared with those presented by ROSEN. At a 10% error tolerance range, a dynamic FFNN (feed-forward neural network) with a hyperbolic tangent sigmoid activation function achieved an accuracy of 86%, compared to the 80% presented by ROSEN.

Regarding SVMs, [17] used SVR (SVMs for regression tasks) for the prediction of both metal loss (severe and non-severe) and malignancy (injurious and benign) in several pipes with various wall thicknesses. The experiment on an 8-inch pipeline had an SVR model with a Gaussian kernel as the best performing model, at 97.81% average performance, although an RLS model came close at 97.69% average performance. The experiment on a 10-inch pipeline had again the SVR model as the top performer at 97.09% average performance. Regarding the experiment on metal loss severity, the best performance was achieved by an SVR model again, but this time with a polynomial kernel, with 98.28% average performance.

A different experiment was done by [18], who used multiple SVMs in order to detect various types of defects and classify. The paper used a separate SVM model for each type of defect, and achieved an average accuracy of 96.6%, while the ROC curves show error rates of 5.3% for dents, 3.6% for gouges and 1.3% for lakes.

Lastly, Random Forests were used by [19] for the classification part of their project, and achieved results of 92% recall and 86% accuracy. Another approach was used by [21], who used TCA (transfer component analysis) combined with a random forest model to classify defects from a number of datasets, each obtained from a different pipeline. The motivation for this project was the fact that the data obtained from each individual pipeline has a different distribution, and TCA is used to reduce those differences. The final result was an average error of 15.38% for the random forest model, and 9.12% when TCA was also used.

As it can be seen, there is no research published at the moment regarding working with machine learning models on magnetic flux leakage data obtained from storage tank bottoms, as the research is focused on oil/gas pipelines. While the principles behind the technologies which are used are similar, they way they are applied in pipelines and tank bottoms is different, which means that the results obtained from one type of application cannot be compared with the results from the other directly.

There are two workflows present in the literature for approaching the issue of MFL anomaly detection using machine learning techniques. The first one is about designing an entire pipeline consisting of a detection algorithm and then adding a classifier after it, in which case most of the effort goes into the anomaly detector, which results in less false positives, which means that a less complex classifier can be used without issue. The second possibility is the use of pre-produced data, where the detection mechanism is outside of the scope of the project. In this case, the added complexity in the data coming from the independence of the two parts creates a requirement for more complex classifiers. All the mentioned papers that used the second data pipeline use some type of neural network.

Another important point is that most of the research is

either focused on distinguishing between multiple types of defects or tries to estimate the depth of the feature. There is some literature present that tries to distinguish between defects and benign captures, but a large portion of it does this step during the detection process, not during the classification as is the case for this project. Overall, the largest number of experiments are performed using various types of neural networks, although SVMs and Random Forests are popular alternatives.

IV. Proposed Method

This section defines the methods that the thesis used for its goal of improving the baseline model. Each subsection describes a different method, and the variations in their implementation. The areas where the methods are introduced into the architecture of the network that is used in the experiments can be seen in Fig. 4.

A. Network architecture

The detailed layer architecture of the neural network used in this paper can be seen in Figure 14 in Appendix A. It receives two types of inputs: one for the images, and one for metadata. On the image input, the network uses three groups of two convolutional layers and a pooling layer which halves the size of the input after each group and then sends it to the next group. The metadata consists of an array of 8 metrics such as the coordinates of a detection within a scan or its proximity to the edge of the metal plate it is part of. This data is then fed into a dense neural network with two fully-connected layers. The outputs of the two networks are then concatenated and fed into another dense layer, which then goes through batch normalization and dropout, if the latter operation is enabled for the experiment.

B. Data Augmentation

1) Mixed element data augmentation: Proposed in a paper by C. Summers and M.J. Dinneen [22], mixed element augmentations are non-linear augmentation techniques which also alter the labels of the images they augment. The proposed techniques are presented in Table I. In the table, all λ values are drawn from a $Beta(\alpha, \alpha)$ distribution with α = 1.

C. Class Activation Maps-based algorithms

High Resolution Class Activation Mapping, better known as HiRes-CAM [10], is a visual explanation technique for Convolutional Neural Network models. In this thesis, the algorithm is slightly modified in order to be used as a data augmentation tool.

The first step in the algorithm is computing the gradient of the output of the neural network with respect to the feature map activations A^k of a chosen convolutional layer, i.e. $\partial y^c / \partial A^k$. The resulting gradients are then averaged over the height and the width of the filter maps, which results in a single value α_k^c, which is the weight of the neuron on the overall activation, i.e.

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A^k}. \tag{1}$$

Lastly, a ReLU activation function is used on the weighted product of the activation feature maps A^k

$$L^c_{HiRes-CAM} = ReLU(\sum_k \frac{\partial y^c}{\partial A^k} \cdot A^k). \tag{2}$$

The resulting heatmap is then up-scaled to the size of the original input, which shows which parts of the original image were important for the classifier when computing the output.

After the heatmap is calculated, its maximum value point is multiplied by a parameter θ, which sets the threshold for occlusion. If a point in the heatmap has a higher value than the threshold, then the corresponding point in the original image gets replaced by a chosen value, which can be either be static (e.g. 0, 1, -1) or taken from any continuous distribution.

1) Training schemes for CAM-based algorithms: Two different approaches were considered for the application of the CAM-based algorithms:

Algorithm 1 CAM-based algorithm applied before the training step

```
Require: trainingSet, model, p, Θ
  for batch in trainingSet do
      convolutionOutput, gradients ← model
      r ← random(0, 1)
      if r < p then
          heatmap ← CAM(batch, lastConvLayer, gradients)
          batch ← occlusion(heatmap, Θ)
      end if
      train(model, batch)
  end for
```

Algorithm 1 was utilized previously by [8]. In this case, the CAM-based algorithm is applied before the training step for a batch. A p value is chosen, between 0 and 1, and then, during each step, a random value r is taken from the [0, 1] interval. If the r is smaller than the p value, then the batch is augmented with a CAM-based algorithm. The occluded area is determined by the Θ parameter. If a pixel in the heatmap has a value over 255 * Θ, the area corresponding to the pixel in the original image will be occluded. The model is then trained on the batch of 128 augmented images. For all the experiments performed with this algorithm, the chosen value for p is 0.25.

Algorithm 2 CAM-based algorithm applied after the training step

```
Require: trainingSet, model, p, Θ
  for batch in trainingSet do
      train(model, batch)
      misclassBatch ← batch[label ≠ prediction]
      heatmap ← CAM(misclassBatch, lastConvLayer, gradients)
      misclassBatch ← occlusion(heatmap, Θ)
      train(model, misclassBatch)
  end for
```

For the second algorithm is a proposed novel approach, the CAM-based algorithms are applied after the training

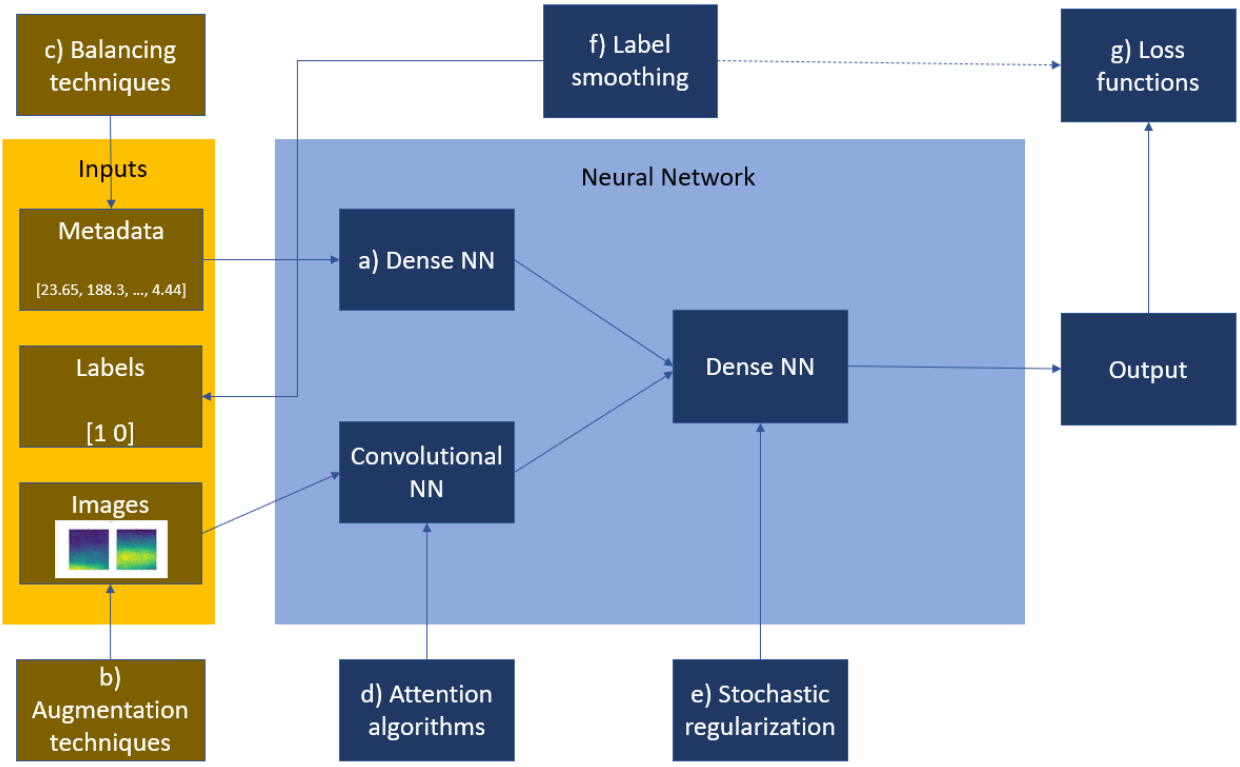


Figure 4: Schema of proposed areas for experiments. The boxes listed with letters are the areas where experiments are performed

step for a batch. First the model is trained on a batch, the misclassified instances are fed into the CAM-based algorithm, and then the model is trained again, this time on the misclassified and augmented instances from the initial batch.

The difference between the two algorithms is that in the first one, the CAM-based algorithm is computing the heatmap before the weights are updated from training on the batch, while the second algorithm uses the weights which were updated by training on the batch. This means that the second algorithm is guaranteed to work with convolutional layer weights which went through at least one training step, while the first one may create a saliency map which is computed with random weights during the first training steps, which can cause issues in the initial stages of training.

The second difference is that the second algorithm updates the weights twice in one step, while the first only updates them once.

D. DropConnect

DropConnect is also a stochastic regularization technique, similar to dropout, but which removes individual weights from the neurons at random, instead of entire neurons [12]. Overall, it is more flexible than dropout, because the adjustments the network has to make have the possibility to create a larger number configurations during each step when singular weights are omitted than when entire neurons are dropped out.

E. Label smoothing

Hard labeling is a binary method of labeling data: if the instance is a member of a class it receives a certain label, otherwise it is assigned the other possible label. One-hot encoding is such an application of the method, where the class the instance is a part is labeled with one, while all the others are labeled with zero. The problem with such a technique is that it forces the model to draw hard decision boundaries between concepts, which makes models prone to over-fitting. One technique for solving such a problem is label smoothing [23], which works by attaching a probability score ϵ to the label, thus leading to softer decision boundaries, and thus better regularization.

The formula used for soft labeling in this paper is:

$$L_s = [L \cdot (1 - \phi)] + (\phi / s), \tag{3}$$

where L_s is the softened label array, L is the original label array, ϕ is the softening factor and s is the length of the label array.

F. Loss Functions

1) Added margin L2-distance loss: Loss functions try to optimize the neural network such that the value of the loss over the training examples tends towards a value k, which is usually 0. However, trying to get as close as possible to that value can lead to over-fitting. By adding a margin m, the network will instead try to come close to $m + k$ rather than k, which should in theory reduce the overfitting. Of course, if the value of m is too large, the network will not train properly, due to the distance between the margin and the optimal value being too large

Table I: Mixed-element data augmentation techniques used in this thesis

Technique	Description	The formula for the augmented image	The formula for the augmented label
Vertical Concatenation	Takes the top λ rows from one image and concatenates them to the bottom $1-\lambda$ rows of the second image	$\tilde{x}(r,c) = \begin{cases} x_1(r,c), \text{if } r \leq \lambda H \\ x_1(r,c), \text{otherwise} \end{cases}$ where H is the height of the image, r is the row coordinate and c is the column coordinate of the pixel	$\tilde{y} = \lambda y_1 + (1-\lambda) y_2$
Horizontal Concatenation	Takes the top λ columns from one image and concatenates them to the bottom $1-\lambda$ columns of the second image	$\tilde{x}(r,c) = \begin{cases} x_1(r,c), \text{if } c \leq \lambda W \\ x_1(r,c), \text{otherwise} \end{cases}$ where W is the width of the image, r is the row coordinate and c is the column coordinate of the pixel	$\tilde{y} = \lambda y_1 + (1-\lambda) y_2$
Mixed Concatenation	Two boundaries determined by λ_1, λ_2, which splits the image into 4 areas, with the main diagonal corners belonging to image 1, and the secondary diagonal belonging to image 2	$\tilde{x}(r,c) = \begin{cases} x_1(r,c), \text{ if } (r < \lambda H \text{ and } c < \lambda W) \\ \text{ or } (r > \lambda H \text{ and } c > \lambda W) \\ x_2(r,c), \text{otherwise} \end{cases}$ where H is the height of the image, W is the width of the image, r is the row coordinate and c is the column coordinate of the pixel	$\tilde{y} = (\lambda_1\lambda_2 + (1-\lambda_1)(1-\lambda_2))y_1 + (\lambda_1(1-\lambda_2) + (1-\lambda_1)\lambda_2)y_2$
Random 2x2 concatenation	The image is split into 4 quarters, and then each part is taken from either image 1 or image 2, with the selection done using a Bernoulli distribution	$\tilde{x}(q) = \begin{cases} x_1(q), \text{ if } p = 1 \\ x_2(q), \text{ if } p = 2 \end{cases}$, for each section q of the image, where p is a value taken from a Binomial distribution with $Pr = 0.5$	$\tilde{y}$ = 1/4 * the number of selected quarters * x_1 + 1/4 * (1 - the number of selected quarters) * x_2
Random square	A square area within image x_1 is replaced by x_2	$\tilde{x}(r,c) = \begin{cases} x_1(r,c), \text{ if } (r > \lambda_2 \text{ and } c > \lambda_3) \text{ and} \\ (\text{r} < \lambda_2 + \lambda_1 \text{ and } c > \lambda_3) \text{ and} \\ (\text{r} < \lambda_2 + \lambda_1 \text{ and } c < \lambda_3 + \lambda_1) \text{ and} \\ (\text{r} > \lambda_2 \text{ and } c < \lambda_3 + \lambda_1) \\ \text{x}_2(r,c), \text{otherwise} \end{cases}$, where λ_1 is the length of the square, λ_2 is the start coordinate of the square on the x-axis, and λ_3 is the start coordinate of the square on the y-axis	$\tilde{y} = y_1(1-\lambda_1) + y_2\lambda_1$
Random Row Interval	A selected interval of rows from x_1 is replaced with rows from x_2	$\tilde{x}(r,c) = \begin{cases} x_2(r,c), \text{ if } r \geq \lambda_1 \text{ and } r \leq \lambda_1 + \lambda_2 \\ \text{x}_1(r,c), \text{otherwise} \end{cases}$, where λ_1 is the starting row for the interval, and λ_2 is the size of the interval	$\tilde{y} = y_1(1-\lambda_1) + y_2\lambda_1$
Random Column Interval	A selected interval of columns from x_1 is replaced with columns from x_2	$\tilde{x}(r,c) = \begin{cases} x_2(r,c), \text{ if } c \geq \lambda_1 \text{ and } c \leq \lambda_1 + \lambda_2 \\ \text{x}_1(r,c), \text{otherwise} \end{cases}$, where λ_1 is the starting row for the interval, and λ_2 is the size of the interval	$\tilde{y} = y_1(1-\lambda_1) + y_2\lambda_1$
Random Rows	A random number of rows from x_1 are replaced with rows from x_2	$\tilde{x} = \begin{cases} x_1(r,c) \text{ if } r \notin I \\ \text{x}_2(r,c), \text{otherwise} \end{cases}$, where I is the set of λ selected random rows	$\tilde{y} = y_1(1-\lambda_1) + y_2\lambda_2$
Random Columns	A random number of columns from x_1 are replaced with columns from x_2	$\tilde{x} = \begin{cases} x_1(r,c) \text{ if } c \notin I \\ \text{x}_2(r,c), \text{otherwise} \end{cases}$, where I is the set of λ selected random columns	$\tilde{y} = y_1(1-\lambda_1) + y_2\lambda_2$
Random Pixels	A random number of pixels from x_1 are replaced with columns from x_2	$\tilde{x} = \begin{cases} x_1(r,c) \text{ if } (r,c) \notin I \\ \text{x}_2(r,c), \text{otherwise} \end{cases}$, where I is the set of λ selected random pixel coordinates	$\tilde{y} = y_1(1-\lambda_1) + y_2\lambda_2$

for the decision boundary to be properly drawn. This approach is similar to soft labeling, however the margin is added to the loss function directly instead of applying it to the ground truth labels.

2) L2-distance based stochastic loss: Another proposed loss is the L2-distance based stochastic loss, which uses an additional parameter θ. This parameter is taken from a two-point distribution as follows:

$$P(X = x) = \begin{cases} 0.5, if x = k \\ 0.5, if x = -k \end{cases}, \tag{4}$$

where k is the variance scaling factor of the parameter, $k \in \mathbb{R}$. The expected value of the parameter is:

$$\begin{aligned} E[\theta] &= x_1 p_1 + x_2(1-p_1) \\ &= k \cdot 0.5 + (-k) \cdot 0.5 \\ &= 0, \end{aligned} \tag{5}$$

and its variance is:

$$\begin{aligned} V[\theta] &= p_1(1-p_1)(x_1 - x_2)^2 \\ &= 0.5 \cdot 0.5 \cdot (k - (-k))^2 \\ &= 0.25 \cdot 4k^2 \\ &= k^2 \end{aligned} \tag{6}$$

The expectation in this case is that the neural network will still minimize its gradients towards the same optimum as before, while the added variance will keep the network from drawing over-fitted boundaries.

3) Stochastic softmax cross entropy loss: This proposed loss function uses the same principle as the one discussed above, but uses the softmax cross-entropy loss function instead of the L2 distance as its basis.

V. Dataset

20 datasets were available for use in the project. Out of those, 18 contained real captures obtained through scans of various tank bottoms with TBIT devices, and

2 contained data captured in a testing environment using metal plates with holes of varying depth drilled in them. 10 of the datasets were created by scans using the Standard calibration method, while the other 10 used the Ultra method.

The entries in the datasets were part of 2 main classes: true calls and false calls, with 5 sub-classes for each, representing by the metal loss percentage of the captures (10-20%, 20-30%, 30-40%, 40-60% and 60-100% metal loss). The distribution of the data can be seen in Figure 5. Even though false calls do not present any actual metal loss, the detected depth is still shown for symmetry purposes with the true calls.

As it can be seen, there is a large imbalance between the number of captures in each class, for both the individual datasets and for the data as a whole. In order to solve the issue, all the datasets have been balanced using the following scheme:

- For each dataset, the data is split into 20 sub-categories: one for each of the two classes combined with the metal loss percentage, taken at intervals of 10% (e.g. 10-20%, 20-30%, etc.)
- For each dataset, the number of instances from the most populated sub-category is chosen
- All the other sub-categories from the dataset are stochastically up-sampled to the number of instances of the most populated one. If a sub-category contains 0 instances, then it remains 0 after up-sampling.

The distribution of the data after the procedure described above can be seen in Figure 6.

VI. Root Cause Analysis

This section discusses the techniques used for the first goal of the thesis, identifying the core issues behind the uneven performance of the baseline model over the 20 provided datasets. The first three sections deal with misclassification analysis and present initial observations. These findings were then discussed with operators which resulted in additional insights. The next part of the section deal with analysing the possibility of a suspected domain shift present in the datasets. The last section presents final observations and discusses how the results of these experiments can be used in the next set of experiments.

Solving the stated issue, the large generalization error of the classifier, requires a deeper look into the methodology that creates the datasets that just exploring the datasets themselves. As such, a framework that work looks into the methodology of an entire industrial process is required. Root cause analysis (RCA) is a risk assessment method used in industrial process engineering for identifying the root causes of failures or problems in a system [7] in order to prevent further re-occurrences. The basic steps of the method are:

- the identification of the problem,
- gathering data and evidence from previous failures,
- performing structured analysis in order to determine the root cause,
- proposing solutions and making recommendations,
- implementing the proposed solutions and recommendations,
- evaluating the proposed solutions.

Root Cause Analysis is the framework of choice in this context as it requires only the data obtained from the failure as input, and then through the analysis steps described above, creates the following outputs:

- documentation of the gathered evidence, of the analysis of the process and of the possible hypothesis tested,
- conclusions about the most likely causes for failure,
- implementations for possible solutions.

A. Visual Analysis

From an initial visual analysis, the following observations could be made:

- There was a significant amount of Poisson noise present in the patches. This type of noise appears due to the discrete nature of the magnetic fields that were measured in order to create the patches
- The true calls presented a very high intensity peak and a very low intensity background, which can be explained by the normalization technique which was used to create the patches
- If an anomaly was captured very close to the edge of a scan, the part that was extracted when building the patch which is outside of the scan will have very low intensity values, while the part that is on the scan has very high intensity
- There are occasional patches that suffer from generation glitches throughout the datasets, due to missing readings from some of the sensors or corrupted data, and there is a very specific type of glitch that only appears in the calibration plates datasets, due to tests on the equipment. Examples of such entries can be seen in Figure 13

B. Misclassification Analysis

Misclassification analysis is an error measurement technique used in root cause analysis. It works by identifying patterns in the misclassified instances and can reveal biases or configuration errors in classifiers and / or data quality issues. For this, three possible types of issues were initially identified:

- Instance mislabeled by operator error - Type 1 error
- Instance misclassified by the classifier - Type 2 error
- Instance inconclusive due to noise - Type 3 error

For type 1, the error came from the specialist that created the ground truth table. Human errors are always a possibility, but the analysis is about trying to find out the severity of the issue. The analysis of this type of error can indicate either biases, if one type of mislabeling happens repeatedly, data quality issues, or identification method issues, if the labeling method is proven to create a large number of mislabeled instances.

For type 2, there are multiple possible causes: data quality issues, classifier bias or configuration error. These issues can lead to two scenarios: non-differential misclassification, where the probability of error between the classes is equal, or differential, where the probability differs between the classes. The first one indicates the classifier

Dataset	(10, 20) TC	(20, 30) TC	(30, 40) TC	(40, 60) TC	(60, 100) TC	Total TC	(10, 20) FC	(20, 30) FC	(30, 40) FC	(40, 60) FC	(60, 100) FC	Total FC	Total
Standard 0	0	38	12	6	4	**60**	0	78	25	11	2	**116**	**176**
Standard 1	0	2	4	4	13	**23**	0	104	103	74	22	**303**	**326**
Standard calibration plates	0	19	34	67	157	**277**	0	20	5	12	5	**42**	**319**
Standard 3	0	97	34	24	36	**191**	0	44	25	14	6	**89**	**280**
Standard 4	0	88	33	8	7	**136**	0	100	53	29	2	**184**	**320**
Standard 5	0	19	14	4	1	**38**	0	90	95	73	27	**285**	**323**
Standard 6	0	8	5	4	3	**20**	0	145	90	45	23	**303**	**323**
Standard 7	0	4	7	13	14	**38**	0	52	56	105	70	**283**	**321**
Standard 8	0	9	1	0	0	**10**	0	232	62	16	1	**311**	**321**
Standard 9	0	3	1	8	15	**27**	0	46	43	92	113	**294**	**321**
Ultra 0	1856	262	65	29	7	**2219**	2093	95	29	13	2	**2232**	**4451**
Ultra 1	0	4	0	0	0	**4**	0	3135	1501	1898	729	**7263**	**7267**
Ultra calibration plates	625	588	530	726	1620	**4089**	6924	2366	1260	626	368	**11544**	**15633**
Ultra 3	675	268	105	69	138	**1255**	2231	630	220	250	300	**3631**	**4886**
Ultra 4	4282	473	83	28	9	**4875**	6028	306	66	35	23	**6458**	**11333**
Ultra 5	954	37	6	41	2	**1040**	8522	1821	325	989	482	**12139**	**13179**
Ultra 6	6	2	0	0	0	**8**	10531	4302	1582	822	60	**17297**	**17305**
Ultra 7	1419	897	283	242	112	**2953**	7766	2854	1308	860	143	**12931**	**15884**
Ultra 8	3082	773	36	13	0	**3904**	26663	2096	413	96	10	**29278**	**33182**
Ultra 9	1197	366	136	548	185	**2432**	4699	1439	770	159	12	**7079**	**9511**
Total	**14096**	**3957**	**1389**	**1834**	**2323**	**23599**	**75457**	**19955**	**8031**	**6219**	**2400**	**112062**	**135661**

Figure 5: The distribution of the entries in the initial 20 unbalanced datasets

Dataset	(10, 20) TC	(20, 30) TC	(30, 40) TC	(40, 60) TC	(60, 100) TC	Total TC	(10, 20) FC	(20, 30) FC	(30, 40) FC	(40, 60) FC	(60, 100) FC	Total FC	Total
Standard 0	0	79	77	154	156	**466**	0	78	78	156	156	**468**	**934**
Standard 1	0	104	104	209	418	**835**	0	104	103	202	415	**824**	**1659**
Standard calibration plates	0	40	42	81	162	**325**	0	40	40	79	120	**279**	**604**
Standard 3	0	97	98	194	390	**779**	0	99	98	193	292	**682**	**1461**
Standard 4	0	100	100	200	400	**800**	0	100	101	201	100	**502**	**1302**
Standard 5	0	95	94	190	95	**474**	0	97	95	188	380	**760**	**1234**
Standard 6	0	145	145	291	290	**871**	0	145	136	285	582	**1148**	**2019**
Standard 7	0	56	56	111	224	**447**	0	56	56	113	220	**445**	**892**
Standard 8	0	233	232	0	0	**465**	0	232	233	466	232	**1163**	**1628**
Standard 9	0	47	47	93	188	**375**	0	48	49	95	188	**380**	**755**
Ultra 0	2082	2092	2093	4186	6279	**16732**	2093	2092	2091	4187	2092	**12555**	**29287**
Ultra 1	0	3135	0	0	0	**3135**	0	3135	3122	6286	12549	**25092**	**28227**
Ultra calibration plates	6917	6928	6919	13832	27681	**62277**	6924	6934	6923	13849	27694	**62324**	**124601**
Ultra 3	2241	2231	2230	4466	8925	**20093**	2231	2229	2233	4465	8922	**20080**	**40173**
Ultra 4	5990	6012	6031	12055	24111	**54199**	6028	6005	6030	12056	24110	**54229**	**108428**
Ultra 5	8528	8522	8520	17043	17044	**59657**	8522	8504	8521	17033	34098	**76678**	**136335**
Ultra 6	10531	10532	0	0	0	**21063**	10531	10560	10519	21041	31594	**84245**	**105308**
Ultra 7	7757	7790	7760	15521	31062	**69890**	7766	7783	7766	15529	31060	**69904**	**139794**
Ultra 8	26692	26656	26667	53327	0	**133342**	26663	26676	26645	53329	53327	**186640**	**319982**
Ultra 9	4696	4698	4709	9389	18792	**42284**	4699	4685	4703	9400	18798	**42285**	**84569**
Total	**75434**	**79592**	**65924**	**131342**	**136217**	**488509**	**75457**	**79602**	**79542**	**159153**	**246929**	**640683**	**1129192**

Figure 6: The distribution of the entries in the datasets after stochastic balancing

is not fit for the given problem, while the second can have multiple explanations, all dependent on the scenario in which they appear.

In the case of type 3, due to various issues, such as sensor faulty calibration or problems caused by the environment, the noise level present in the data is too strong for the operator to clearly identify possible issues. If so, then the classifier also cannot be expected to classify the instance properly.

Types 1 and 3 indicate issues in the quality of the ground truth. If such issues are present and significant in the data, possible avenues for addressing the issue include increasing the robustness of the classifier by modifying its configuration or by creating simulated data that can be used in place of the original captures. There already exists a tool within ROSEN that can generate simulated data, but using techniques such as Generative Adversarial Networks (GAN) is also possible.

Since the quantity of data is large, having all entries re-checked by an operator is unfeasible. Therefore, an experiment was set up, where 10 random misclassified instances were taken from each test set, for a total of 200 entries. Since the experiment uses 'leave-one-out' k-fold cross-validation, each test set is equivalent to one dataset, and as such any notable differences between the batches of misclassified may indicate a domain shift in the data.

C. Initial observations

D. Operator investigation

E. Domain shift analysis

As discussed above, one of the sources of data, the calibration plates used to test the sensors, had 6 misclassified instances of true calls with high percentage of metal loss in the Standard calibration and 2 in the Ultra calibration, which constitute together 66% of all such entries. Such behaviour from the model might indicate that a domain shift is present in those datasets.

Domain shift is the phenomenon that appears when the distributions of the testing and training datasets are different. Important for the model, when cross-validation is performed, and the issue of domain shift is present, the cross-validation will be biased towards the domain of the training data, which will result in a weaker performance when the model is deployed on previous unseen data [24].

In order to confirm such an issue, PCA (principal component analysis) [25] has been used to reduce the data to 2 or 3 dimensions which could be plotted in order to obtain visual confirmation of the phenomenon. In this case, principal component analysis (PCA) was chosen

because the technique preserves the global structure of the data, which is required here in order to properly visualize the structural difference between the distributions of the datasets. There were other possible techniques, such as t-SNE or other related manifold techniques, however those do not preserve the global structure of the data.

The PCA implementation was used on a sample of 150 entries taken randomly from each dataset. The results can be seen in Figure 16. For 2 dimensions, the remaining variance was 26%, and for 3 dimensions 38%, both of which are low values, but which is to be expected when the dimensionality reduction is drastic (from 2048 to 2 or 3 dimensions). In the figure, each color represents a different dataset, the circles represent false calls and the triangles represent true calls.

As it was difficult to visualize each individual dataset, another one-vs-rest visualization was created for the calibration plates - standard settings dataset, which can be seen in Figure 17.

As it can be seen from Figure 17, the true positive entries from the calibration plates – standard settings dataset tend to cluster at the extremes of the true positive clusters. As discussed before, this dataset contains a large number of entries with high metal loss, which when combined with these visual observations shows that true calls with high metal loss have a slightly different distribution from lower loss calls. As such, when this dataset is removed from the training dataset, the distribution domain for training cannot extend to this specific type of high metal loss true calls.

Figure 20 shows the distribution of the two classes throughout the datasets (false calls in green, true calls in red). There are two clusters of true calls on the extreme left and right of the distribution. As shown with the calibration plates standard dataset above, the further away from the origin of the plane the points are situated, the higher the metal loss. This was to be expected, the normalization of the signal when transformed into an image results in stronger contrasts than usual between the intensity values of the pixels for high metal loss true calls, as the amplitudes of the signals in the affected area are also higher than for lower metal loss anomalies.

F. Final observations

There is a domain shift issue present in a number of datasets, and it is most pronounced in the calibration plates - standard dataset. This shift can be seen from Fig. 17, where the instances from dataset are presented in coral pink, to other datasets, such as the ones from Fig. 18 or Fig. 19. While the latter two datasets present a random distribution of their instances, it is quite clear that the calibration plates datasets has its instances concentrated at the very edges of the distribution, which indicates some degree of domain shift.

Another points that supports this hypothesis is the fact that when the dataset is used as for testing in the k-fold cross-validation, the classifier loses the ability to properly classify true calls with high metal loss. The calibration plates dataset contains most of the high metal loss detections, as it can be seen in Fig. 5. In such a case, it is expected that the model would not be able to extend its domain to such instances, since they are not in training set. This is consistent with the results obtained by the baseline model, giving further proof to the hypothesis.

Initial analysis pointed towards the metal loss being the main reason behind the issue. However, after further discussions with operators, the more plausible explanation was that the difference could actually came from a difference in shape of the anomalies, as drilled holes create a different distortion in magnetic fields than corrosion. Such a hypothesis would be consistent with the results of other studies which looked at defect geometries, such as [26] [18]. Nevertheless, a more in-depth analysis should be done in the future regarding the geometry of different types of defects in the context of image analysis.

Another problematic issue is high intra-class variation and low inter-class variation between the true and the false calls in the overall distribution of the dataset, as shown by the PCA in Figure 20. One possibility for addressing the issue, additional pre-processing techniques could be applied on the images in order to increase the distinction between the classes.

There is a possible data quality issue stemming from the fact that the sensors that measure magnetic flux leakage can gather more information in the central section of a scan than on the exteriors. These distortions lead to false calls being virtually indistinguishable from true calls at the edges of the scans. This issue does not affect true calls in a meaningful way because the distortion would make the instance seem to have more metal loss than normal. As said, this does not present an issue for false calls in a binary classification setting, but it is expected to cause issues in a multi-class classification setting, where the intensity of the metal loss is important.

There is also a label quality issue which comes from the fact that different parts of the process of identifying anomalies have different methodologies. Due to this, datasets such as the ones used in this project will have an inherent level of label noise in them. Fortunately, the level of noise is not high enough to interfere with the training process of the model used in the other experiments performed in this paper.

Going into the next section, three of the identified issues are pertinent for experiments on generalization techniques:

1) Label noise is an inherent feature of the methodology that creates the datasets, and as such a solution to it is out of the scope of the project. There is however the possibility of using techniques that have integrated mechanisms against label noise
2) High intra-class and low inter-class variance means that feature identification and extraction need to be given significant attention
3) Some datasets present domain shift, which is part of the reason behind the discrepancy between the results on the 20 datasets. Multiple methods can be used for extending the domain space of the model, such as data augmentation or adding new data

VII. Experiments

This section presents the experiments performed in this thesis and presents their results. The section first

lists the methods used in the experiments of this thesis, then presents the theory part behind alternative methods tested along with those presented in section IV - proposed method. After that, the setup for the experiments and the metrics on which the experiments are measured are listed. Lastly, the results are presented along with a short discussion on them. The experiments where the results are marked with an "X" could not be completed, but the discussion around why they failed to yield results is still important.

A. Exploratory analysis

The following areas have been tested in the experiments of this paper, as described in Fig. 4:

a) Different network architectures, one with image-only input, one with image and metadata input
b) The images coming from the MFL scans that constitute the input, on which augmentation techniques has been applied
c) Dataset balancing techniques applied to the datasets, namely balancing the datasets by class, balancing them around both class and dataset size, and lastly comparing them to the results obtained on the original unbalanced datasets
d) Attention-based that have been applied to the final convolutional layer of the network, namely GRAD-CAM [9] and HiRes-CAM [10]. The first experiments were performed using GRAD-CAM, in order to find the ideal configuration for further experiments. Those experiments were concerned with the type of occlusion to be used and the classes to which the input should apply. After that, experiments were performed with for both algorithms, to find out which training schema and which occlusion threshold bring better results
e) Stochastic regularization techniques that have been applied to the second to last dense layer of the network (the last one being the output layer)
f) Label smoothing, where multiple values for the softening factor were tested
g) Various loss functions, starting with the Softmax and Sigmoid loss functions. After that, experiments were performed with the L2 loss function, first by itself, then followed by experiments where a margin parameter was added to create an effect similar to label softening. Lastly, an parameter that added extra variance to the function was added. The same experiments were then performed using instead the Softmax Loss function.

B. Background of exploratory analysis

1) Network architectures: An alternative to the baseline network can be seen in Fig. 15, which only uses the images as input. Since the network does not use metadata input, there is no bias towards false detections at the borders of the scans, unlike the previous architecture.

2) Class Activation Maps-based algorithms: Gradient Weighted Class Activation Mapping, or GRAD-CAM [10], is an alternative to HiRes-CAM, which proposes that instead of directly multiplying the gradients of the output with respect to the selected convolutional layer, the gradients of the activation maps should instead be averaged, which results in a single value, which is then multiplied with the filter maps. The formula for the heatmap then becomes

$$L^c_{HiRes-CAM} = ReLU(\sum_k \frac{\partial y^c}{\partial A^k} \cdot A^k). \quad (7)$$

3) Dropout: An alternative to dropConnect is dropout, which is a regularization method where a number of neurons chosen at random in a neural network are not considered during a particular forward or backward pass [11]. The technique helps reduce the generalization error of a network by reducing the possibility of co-adaptation on training data in the layers of the network, a phenomena in which the neurons develop excessive co-dependencies with each other, which leads to overfitting.

4) Data augmentation:

Basic data augmentation Basic data augmentation refers to a number of non-linear image transformation techniques, which generate new images based on the original ones. The list of techniques used in this paper can be seen in Table II.

Table II: Basic data augmentation techniques

Method	Description
Rotation	Rotate the image at a chosen angle α
Shifting	Each pixel of the image is moved to a new location along an axis
Zooming	Expand the pixels from a certain part of the image
Flipping	Flip the image along an axis

Linear MixUp Linear MixUp [27] is a linear data augmentation technique that augments the data by combining two images into one using the formula

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j, \quad (8)$$

where x_i and x_j are the raw input vectors and λ is a value sampled from a $Beta(\alpha, \alpha)$ distribution, with α = 1.

The label of the new image is

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j. \quad (9)$$

Overall, the technique is expected to counteract the memorization problem of neural networks by providing convex inputs and labels, which encourages robustness for the network.

5) Loss functions: Table III presents all the loss functions used in experiments in this thesis.

Softmax cross entropy loss The cross entropy is a metric which measures the difference between the probability distribution predicted by a machine learning model and the actual distribution of the data. The loss function associated with it tries to optimize the model such that the value of the metric gets close to a given margin m, which is usually 0, which means that the 2 probability distributions are identical. In front of the cross entropy loss is the softmax activation function, which calculates the probability of each target class over all the possible classes, which means that all the probabilities will be shifted to the range [0, 1], and the sum of all probabilities will be equal to 1. The two functions are used together, first the outputs of

Table III: Proposed loss functions and their formulas

Loss function	Formula
Softmax Cross Entropy Loss	$L(y, P(y\|x)) = -logP(y\|x)$, where $P(y\|x) = \frac{e^{f_y(x)}}{\sum_j e^{f_j(x)}}$, $f_y(x)$ and $f_j(x)$ are the outputs corresponding to the classes y and j
Sigmoid Cross Entropy Loss	$L(y, P(y\|x)) = \begin{cases} -log(\sigma(s_i)), \text{ if } y = 1, \\ -log(1-\sigma(s_i)), \text{ if } y \neq 1 \end{cases}$, where $\sigma(s_i) = \frac{1}{1+e^{-f(x)}}$, and i is the output corresponding to the given class
L2 Loss	$L(y, \hat{y}) = \sqrt{(output_{x1} - label_{x1})^2 + (output_{x2} - label_{x2})^2}$, where $x1$ and $x2$ are the two dimensions of the one-hot encoded label, y is the actual label, and $\hat{y}$ is the probability prediction of of the model
Margin-Based L2 Loss	$L(y, \hat{y}) = (d - m)^2$, where d is the L2 distance as described above, and m is the margin parameter
Stochastic L2 Loss	$L(y, \hat{y}) = (d - m + \theta)^2$, where d is the L2 distance as described above, m is the margin parameter, and θ is taken from a Bernoulli distribution with $Pr(X = 1) = \frac{1}{2}$ and then multiplied by a factor of k
Stochastic Softmax Cross Entropy Loss	$L(y, P(y\|x)) = (-logP(y\|x) - m + \theta)^2$, where $P(y\|x)$ is the same as described for the Softmax Cross Entropy, m is the margin and θ is taken from a two-point distribution with $Pr(X = k) = \frac{1}{2}$, where k is the variance scaling factor, with $k \in \mathbb{R}$
Batch Stochastic Softmax Cross Entropy Loss	$L(y, P(y\|x)) = ((\frac{1}{N}\sum_{i=1}^{N} -logP(y_i\|x)) - m + \theta)^2$, the same method as for the Stochastic Softmax Cross Entropy Loss, but the m and θ parameters are added at the batch level, instead of for every sample

the model are normalized via the softmax, then the loss function helps optimize the model.

Sigmoid cross entropy loss In the case of binary classification, an alternative to the softmax function is the sigmoid activation function, which also normalizes the outputs of the network to the [0, 1] range, but uses a different method for doing so, as it can be seen in Table III.

L2-distance loss function The L2-loss function, also known as the squared error loss function is the squared difference between the prediction output of a network and the actual label of the data. The function provides an alternative way of calculating the error rate of classification. Instead of using probabilities like the previous loss functions, this function uses the squared distance between the output of the network and the label of the given instance.

$$L2 - loss = (y_{actual} - y_{predicted})^2 \tag{10}$$

The function is widely used in various machine learning tasks due to its sensitivity to outliers. This can also be a disadvantage however, as the resulting predictions of the function will be heavily skewed by outliers due to the squaring factor.

When averaged over a batch, the function is called the Mean Squared Error function, where:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{actual} - y_{predicted})^2 \tag{11}$$

C. Validation metrics

1) Accuracy: The accuracy is the metric that measures the percentage of correct predictions of a model among all the entries. Its formula is

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}, \tag{12}$$

where TP is the true positive rate, TN is the true negative rate, FP is the false positive rate, and FN is the false negative rate.

2) ROC-AUC curve: The Area under the Receiver Operating Characteristic curve (ROC-AUC) is a measure that shows how good a model is at distinguishing between classes by showing the relationship between the TPR and the FPR. By its definition, the larger the area under the curve, the better the results.

3) False positive rate at k recall (FPR-k TPR): The false positive rate at k true positive rate (also known as the false positive rate at k recall) is the probability that a negative (out-of-distribution) example is misclassified as positive (in-distribution) when the true positive rate is k. In this paper, the threshold k=95% is used in all the experiments, which means a high degree of sensitivity is expected from any model being tested.

D. Experimental setup

The experiments described in this thesis were all implemented using Python 3.8 and Tensorflow 2.6, and were performed using a machine running Linux, with 64GB RAM and accelerated using a NVIDIA RTX 3090, with 26GB VRAM.

Training was performed using batches of 128 entries. A variant of early stopping was implemented by testing the model on the validation dataset every 1000 batches. If the performance of the model was better than the current best, the model would be saved to an external file. Due to the fact that data balancing techniques were used, the number of epochs was kept low, at 3 epochs per training fold. For each fold, the training would be done in 3 runs, each one with a different random initialization. At the end of the training phase, meaning at the end of all the training epochs for all runs for each cross-validation fold, the best performing model would be restored and tested on the test dataset. This way, if the model were to overfit and lose performance, the over-fitting would not affect the final result. Thus, the results shown in the tables of Section VI show the results obtained using the best parameter configuration on the validation dataset over 3 runs, and then evaluated on the test dataset.

For all experiments the hyper-parameters of the network were kept the same, with any changes being mentioned in the results section. The architecture of the network on which all the experiments were conducted, except those

on alternative architectures, can be seen in Figure 14. All the weights of the network are randomly initialized. All the dense layers of the network use the ReLU activation function, except for the last one, which uses the Softmax activation function. The network learns using stochastic gradient descent, with a learning rate 0.005 and no decay. The usual number of training epochs is 3 per fold. Beta regularization is applied by default to all weights after a training step. Dropout/DropConnect is set to 0 for all experiments, except for those on stochastic regularization. All the experiments used the Softmax loss function except for the ones on loss functions.

VIII. Results

A. Network Architecture

The two architectures described before were tested on the same setup, the results can be seen in Table IV. Even though the image-only input network has a slightly better results regarding false positives, as it can be seen from the AUC-ROC, the accuracy and the FPR at 95% TPR results are significantly worse than the two-input architecture.

Table IV: Comparison between results of architectures, no changes from the experimental setup presented in section VII

Architecture	FPR at 95% TPR	AUC-ROC	Acc.
Image-input only	44.47	89.57	77.42
Image + meta-data input	39.54	89.35	78.62

B. Dataset balancing techniques

In order to understand the effect balancing techniques had on the training of the model, three experiments were performed. In the first one the baselines model, as seen in Figure 14 was trained on the original datasets, as seen in Figure 5. In this experiment, since the entries do not repeat in any way, a higher number of epochs was required

In the second one, the same model was trained on the stochastically balanced datasets, the ones seen in Figure 6, in order to see if balancing between classes could help the model.

In the third one, the balancing was first done as in the previous experiment, and then the datasets were stochaistically balanced again, this time with respect to their number of instances. As the difference between the sizes of different datasets is significant, this experiment was done in order to check whether the model could suffer from overfitting to the datasets with larger numbers of instances.

In all the other experiments presented in this thesis, the model was trained on the dataset which was balanced around the two classes, and the result obtained from this experiment when using only the two-input architecture presented in Figure 14 serves as the baseline result.

The results of the experiments on balancing techniques can be found in Table V. The best results were achieved in the experiment where the balancing was done only around the two classes. In the case of the unbalanced datasets, as expected the imbalance between the two classes affected the performance of the model, as the accuracy had marginally lower results, but the AUROC suffered significantly more.

Table V: Comparison between dataset balancing techniques, the experiment on unbalanced data uses 25 epochs of training instead of the standard 3, for the other techniques there are no changes from the experimental setup presented in section VII

Technique	FPR at 95% TPR	AUC-ROC	Acc.
Unbalanced datasets	48.46	82.77	77.60
Balanced classes	39.54	89.35	78.62
Balanced classes and dataset sizes	44.01	86.54	78.36

C. Data augmentation

Three experiments were performed using data augmentation, each using a different set of techniques. The first one used the techniques used in Table II, the second one uses all the non-linear techniques from Table VI together, and the third one uses only linear mixup. The results of the three experiments can be seen in Figure VI. The strongest results were obtained by the techniques presented by [22], however, none of the presented techniques improved over the provided baseline configuration.

Table VI: Data Augmentation Techniques, no changes from the experimental setup presented in section VII

Technique	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	39.54	89.35	78.62
Classical	41.27	88.12	77.56
Mixed-Element	40.57	88.53	77.15
Linear Mixup	44.95	87.99	77.28

D. Stochastic Regularization

For both techniques, three experiments were performed, with 20%, 30% and 40% of the weights/neurons being dropped. The results can be seen in Table VII.

Between the two techniques, the difference was notable. The experiments that used DropConnect were more successful than the baseline, while the results for dropout were significantly lower on all metrics. Additionally, in the case of dropout, values higher than 20% caused the network does not train to be unable to reach convergence.

Table VII: Stochastic Regularization, in this set of experiments the Dropout/DropConnect layer is activated and given the values seen in the table

Technique	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	39.54	89.35	78.62
Dropout - 20%	48.82	86.57	77.81
Dropout - 30%	X	X	X
DropConnect - 20%	40.10	89.58	78.70
DropConnect - 30%	39.90	89.44	79.13
DropConnect - 40%	42.28	89.98	77.71

E. Soft labeling

Soft labeling was used in multiple experiments with different values for the softening factor in the range 0.2 to 0.4 in order to find the optimal value. The results can be seen in Table VIII. In all the experiments involving soft-labeling, the accuracy increased significantly when compared to the baseline, but the AUC-ROC decreased, which indicates that soft labeling improved the overall

model, but the number of false positives increased, which in this case meant more false calls identified as true calls.

Table VIII: Table of results for soft labeling, no changes from the experimental setup presented in section VII

Softening factor	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	39.54	89.35	78.62
0.2	41.34	88.22	79.59
0.3	41.35	88.05	79.13
0.4	37.97	88.67	79.21

F. Attention-based algorithms

1) Occlusion types: Multiple experiments were performed in order to determine the optimal way to occlude the high attention areas. All the experiments performed for this purpose used Algorithm 1 and an occlusion threshold of 0.8. Three values were chosen: -1, 0 and 1. The results for these experiments can be seen in Table IX. However, none of the values proved to improve on the baseline.

Table IX: Types of occlusion in GRAD-CAM (θ = 0.8), algorithm 1 integrated in the last convolutional layer of the network, a learning rate of 0.0025 was used, training used 5 epochs per cross-validation fold

Technique	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	39.54	89.35	78.62
-1 occlusion	44.33	82.77	77.07
1 occlusion	57.11	78.52	74.03
0 occlusion	60.39	74.97	71.04
GN occlusion	40.61	89.55	79.44

The solution for this issue was to have an occlusion mechanism that takes values from a Gaussian distribution with the mean and standard deviation equal with that of the pixel value distribution of the image to be occluded. The experiment which used this type of occlusion mask had a higher accuracy and a lower FPR at 95% recall compared to the baseline, but had a weaker result on the AUC-ROC metric. Nevertheless, this type of occlusion was chosen for usage in all the other experiments regarding attention-based algorithms.

All the experiments in this subsection used training algorithm 1, and an occlusion threshold of 0.8.

2) Training algorithms: Another question besides the problem of the type of occlusion was whether the network would benefit more from training on the entire batch, or on just one of the classes. The results of the three experiments can be seen in Table X. The best results were achieved when only the false calls were augmented. In the other two experiments, the network could not reach convergence, therefore the results are weaker than the baseline.

Table X: Classes used in occlusion, GRAD-CAM - algorithm 1 integrated in the last convolutional layer of the network, a learning rate of 0.0025 was used, training used 5 epochs per cross-validation fold

Technique	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	39.54	89.35	78.62
False calls only	40.61	89.55	79.44
True calls only	47.09	83.70	77.71
Both classes	44.33	82.77	77.07

After the optimal occlusion pattern and training schema were found, the two attention-based algorithms, GRAD-CAM [9] and HiRes-CAM [10], were tested using the two proposed training algorithms: 1 and 2. For this, values for the occlusion threshold ranging from 0.7 to 0.85 were used with both GRAD-CAM and HiRes-CAM. The results for Algorithm 1 and GRAD-CAM can be seen in Table XII, while the results for HiRes-CAM can be seen in Table XI. The results results for Algorithm 2 and GRAD-CAM can be seen in Table XIII, while the results for HiRes-CAM can be seen in Table XIV.

Table XI: HiRes-CAM - Thresholds θ for false positives only, algorithm 1 integrated in the last convolutional layer of the network, a learning rate of 0.0025 was used, training used 5 epochs per cross-validation fold

Technique	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	39.54	89.35	78.62
θ = 0.7	38.89	88.95	78.81
θ = 0.75	37.36	88.93	79.12
θ = 0.8	36.00	89.35	79.52
θ = 0.85	37.67	89.00	78.32

Table XII: GRAD-CAM - Thresholds θ for false positives only, algorithm 1 integrated in the last convolutional layer of the network, a learning rate of 0.0025 was used, training used 5 epochs per cross-validation fold

Technique	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	39.54	89.35	78.62
θ = 0.7	40.06	87.94	77.80
θ = 0.75	40.57	88.45	79.20
θ = 0.8	40.61	89.55	79.44
θ = 0.85	41.40	88.37	78.04

Using the training algorithm 1 with HiRes-CAM, the optimum value was at an occlusion threshold of 0.8, where all the metrics were higher or equal to the baseline. Also, in this case, all the values had a better FPR at 95% recall and almost all of the had a better accuracy than the baseline.

The same training algorithm combined with GRAD-CAM offered overall worse results, but still above the baseline. Again, the occlusion threshold of 0.8 had the best results.

Table XIII: Results for GRAD-CAM applied only on misclassified false positives, algorithm 2 integrated in the last convolutional layer of the network, a learning rate of 0.0025 was used, training used 5 epochs per cross-validation fold

Technique	FPR at 95% TPR	AUC-ROC	Accuracy
Baseline	39.54	89.35	78.62
θ = 0.7	39.68	89.76	78.42
θ = 0.75	35.15	90.92	78.42
θ = 0.8	34.48	90.76	79.56
θ = 0.85	40.78	89.70	79.19

Training algorithm 2 had a similar pattern of results, with the optimum occlusion threshold again at 0.8, for both GRAD-CAM and HiRes-CAM, and with the latter outperforming the first one.

The algorithm 2 outperformed algorithm 1 by a significant margin on all metrics. Additionally, HiRes-CAM

Table XIV: Results for HiRes-CAM applied only on misclassified false positives, algorithm 2 integrated in the last convolutional layer of the network, a learning rate of 0.0025 was used, training used 5 epochs per cross-validation fold

Technique	FPR at 95% TPR	AUC-ROC	Accuracy
Baseline	39.54	89.35	78.62
θ = 0.7	38.31	90.28	80.26
θ = 0.75	36.90	89.87	79.45
θ = 0.8	32.37	91.57	79.76
θ = 0.85	36.50	90.21	80.62
Stoch. θ	37.33	89.75	79.57

and an occlusion threshold of 0.8 obtained the best result out of all the experiments on the with a result of 32.37% FPR-95% TPR and an AUROC of 91.57%. The same experiment configuration, but with a threshold of 0.85 had the highest registered accuracy out of all the experiments at 80.62%.

G. Loss functions

The first two experiments on loss functions were performed using the softmax and the sigmoid cross entropy loss functions. The results can be seen in Table XV. Here, the softmax cross entropy had better results on all metrics, due to the fact that the softmax function assumes mutual exclusivity between the probabilities of the outputs of the network, which is the case in this project.

Table XV: Table of results for the Sigmoid loss function, no changes from the experimental setup presented in section VII

Margin m	FPR at 95% TPR	AUC-ROC	Acc.
Softmax	39.54	89.35	78.62
Sigmoid	41.10	89.22	78.25

After that, an experiment was performed using the L2 loss function. The result can be seen in Table XVI. For the margin added L2-loss function, multiple experiments were conducted in order to find the best optimum value for the margin value m, ranging from 0.1 to 0.5, which can be seen in Table XVII. Looking at the FPR-95% recall, all the chosen values improved the model, with an optimum value at 0.25. Looking at the AUC-ROC, the best result was achieved at a margin of 0.1, which was also the only one to have a higher value than the baseline. Regarding the accuracy, all the tested values had better results than the baseline. Overall, notably, there is a clear drop in results for margin values over 0.3 in all metrics, which indicates that if the margin is too large, the network will not be able to train properly.

Table XVI: Table of results for the L2 loss function, no changes from the experimental setup presented in section VII

Margin m	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	39.54	89.35	78.62
L2	55.23	84.74	78.38

The most promising margin values were chosen from the previous set of experiments were tested using the stochastic L2-loss function. The value 0 was chosen to see

Table XVII: Table of results for the values of the margin added to the L2 loss function, no changes from the experimental setup presented in section VII

Margin m	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	39.54	89.35	78.62
0.1	36.86	89.52	79.61
0.2	35.08	88.39	79.49
0.25	33.71	88.88	79.40
0.3	36.80	88.96	79.61
0.4	37.73	87.17	78.77
0.5	37.37	87.75	78.98

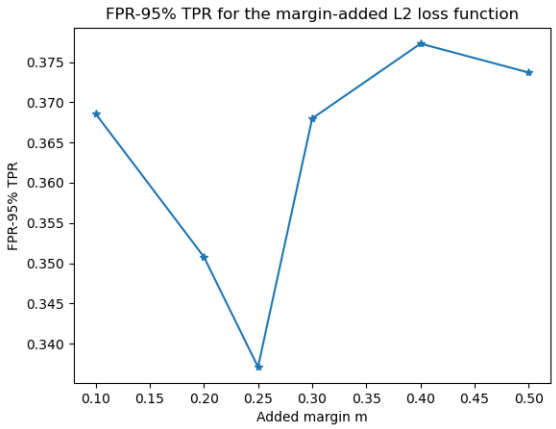


Figure 7: FPR-95% TPR for the values given to the margin added to the L2 loss function presenting the trend towards the optimum value of 0.2

the effect of using the added variance without interference from the added margin. The 0.1 had the highest accuracy and AUC-ROC value, and the 0.25 had the lowest FPR at 95% recall, while the 0.5 was chosen just as a large value comparison. Each of the values was tested with an added variance of 0.25 and 0.5. The results can be seen in table XVIII.

Table XVIII: Table of results for the combinations of margin and variance values added to the L2 loss function, no changes from the experimental setup presented in section VII

Margin m	Variance θ	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	-	39.54	89.35	78.62
0	$\pm$ 0.25	46.96	87.37	79.09
0	$\pm$ 0.5	41.23	88.34	78.00
0	$\pm$ 1	X	X	X
0.1	$\pm$ 0.25	43.60	87.58	78.14
0.1	$\pm$ 0.5	42.64	86.99	79.26
0.1	$\pm$ 1	X	X	X
0.25	$\pm$ 0.25	38.72	87.42	77.87
0.25	$\pm$ 0.5	36.10	89.10	79.33
0.25	$\pm$ 1	X	X	X
0.5	$\pm$ 0.25	41.40	88.73	78.04
0.5	$\pm$ 0.5	42.68	87.41	78.01
0.5	$\pm$ 1	X	X	X

Unfortunately, the added variance did not improve any of the previously obtained results. All of the results were overall worse than the baseline, with the exception of the results obtained at an added margin of 0.25. The results, however, were weaker than those with just the margin.

The experiments done on the L2-loss based function

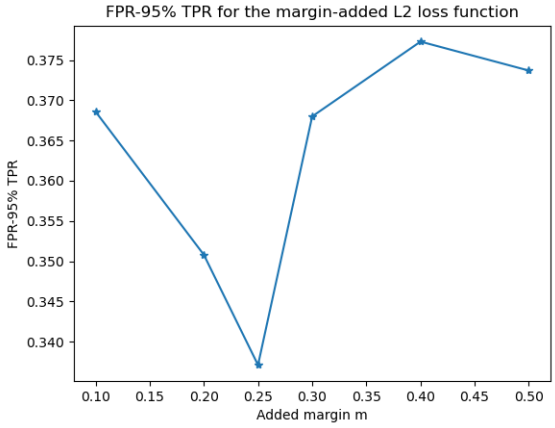


Figure 8: FPR-95% TPR for the margin values added to the Softmax Loss function presenting the trend towards the optimum value of 0.25

were repeated using the Softmax function, with the results shown in Table XIX. Overall, the behaviour of the network was similar to the previous experiments. There is an optimum point for the the margin at 0.2 when looking at the FPR at 95% recall metric and the AUC-ROC, and the values of the margins follow a curve on this metric. The accuracy also followed a similar pattern, with the highest value at a margin of 0.4, although the margin of 0.3 had a similar result as well.

Table XIX: Table of results for the margin values added to the Softmax Loss function, no changes from the experimental setup presented in section VII

Margin m	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	39.54	89.35	78.62
0.1	40.45	89.27	77.88
0.15	37.14	89.43	78.62
0.2	34.40	90.03	78.69
0.25	35.74	88.90	78.84
0.3	36.01	89.25	79.00
0.4	37.89	89.15	79.01
0.5	39.26	88.36	77.71

Table XX: Table of results for the combinations of margin and variance values added to the Softmax loss function, no changes from the experimental setup presented in section VII

Margin m	Variance θ	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	-	39.54	89.35	78.62
0	$\pm$ 0.25	35.18	90.29	78.95
0	$\pm$ 0.5	39.17	89.44	77.88
0	$\pm$ 1	X	X	X
0.1	$\pm$ 0.25	41.48	88.94	77.22
0.1	$\pm$ 0.5	41.35	88.04	80.05
0.1	$\pm$ 1	X	X	X
0.2	$\pm$ 0.25	36.68	89.46	78.59
0.2	$\pm$ 0.5	39.46	89.78	79.63
0.2	$\pm$ 1	X	X	X
0.25	$\pm$ 0.25	39.06	89.31	78.71
0.25	$\pm$ 0.5	36.31	89.78	79.34
0.25	$\pm$ 1	X	X	X
0.5	$\pm$ 0.25	39.71	88.44	78.36
0.5	$\pm$ 0.5	41.29	88.35	77.30
0.5	$\pm$ 1	X	X	X

When the additional variance is added to the Softmax loss function, the results improved overall regarding the accuracy but decreased on the FPR at 95% TPR, with one notable exception. At a margin of 0 and variance of 0.25, the model had the highest AUC-ROC result out of all the tested loss function configurations, along with better results on all the metrics when compared to the baseline. Overall, in this case, the addition of the θ parameter seems to benefit the network, but the difference is not significant.

Table XXI: Table of results for the combination of margin and variance values added batch stochastic Softmax cross entropy loss function, no changes from the experimental setup presented in section VII

Margin m	Variance θ	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	-	39.54	89.35	78.62
0.15	0	43.88	88.76	78.30
0.20	0	40.39	89.51	78.62
0.25	0	46.07	87.99	76.77
0.15	0.15	X	X	X
0.20	0.15	X	X	X
0.25	0.15	X	X	X

The last loss function that was tested was the batch stochastic Softmax loss function. The results for this function can be seen in Table XXI. Again, a value of 0.2 for the margin had a positive effect on the network. Only three values are present in table, as the other values would cause the network to crash during training. As the margin and the variance would be added at the batch level, instead of individually, values for the margin larger than 0.2 destabilized the network. Also, any value used for θ other than 0 had the same effect.

H. Combined Approaches

The three most successful approaches overall were the use of attention-based algorithms, the loss functions and the dropConnect stochastic regularization. The final round of experiments focused on combining the approaches in order to see if further improvements were possible.

The first round focused on combining HiRes-CAM / GRAD-CAM with the proposed loss functions. Unfortunately, the two approaches could not be combined, as using them together would cause the code of the network to crash during training, due the values of the loss function causing overflow.

The second round tried to combine HiRes-CAM / GRAD-CAM with DropConnect. For this, the same parameters were chosen as the most successful experiment on attention-based algorithms, and multiple values for DropConnect were used. The results can be seen in Table XXII. Unfortunately, the combinations did not improve the results obtained by the individual experiments.

Table XXII: Table of results for HiRes-CAM with an occlusion threshold of 0.8 and using Algorithm 2 combined with DropConnect, learning rate 0.25

Technique	FPR at 95% TPR	AUC-ROC	Acc.
Baseline	39.54	89.35	78.62
DropConnect - 20%	38.10	90.45	78.88
DropConnect - 30%	37.19	89.73	79.17
DropConnect - 40%	35.18	90.29	78.95

IX. Discussion

This section is dedicated to discussing observations made during or after performing the experiments described in the previous section, for some of the methods which required a more detailed discussion.

A. Data Augmentation

All the tested methods had worse results than the baseline. This result is due to the fact that the domains of the augmented data and the original data occupy different areas in the domain space, with the issue mostly relating to the physical properties of magnetic flux leakage.

The issue with data augmentation in the context of MFL captures come from the the fact that it is not possible to obtain real captures that have the same properties as the augmented samples. Rotating an anomaly by α degrees will not result in a capture that is also rotated by α degrees. Instead, the rotation will cause the magnetic fields to flow in a different manner around the anomaly, which will modify the way the information is received in the PHH and PVH channels, and thus create a completely different image from the initial capture. Flipping also suffers from the same type of issue, as an image flipped along one of its axes would only be possible if the direction of the magnetic field was inverted compared to the original direction used by the measurement tool.

Shifting suffers from a different issue than the other techniques. It is most useful when the areas of importance are in different locations in the images in the dataset. However, in this project, the detection algorithm used before the classification algorithm creates bounding boxes which centre around the possible anomalies, thus the important areas will be approximately in the same area most of the time.

Augmentation using the non-linear mixed-element techniques described in Table VI had a marginally better result on the AUC-ROC and on the FPR - 95% recall, but had an overall lower accuracy. While the results indicate better generalization than classical augmentation, the augmented data domain the techniques create is still too different from domain of the original images to improve generalization, and thus the results are worse than the baseline.

B. Stochastic Regularization

For all the values chosen for DropConnect, the AUROC results were higher than the baseline, which showed a decrease in the number of false positives, translating directly to a decrease in the number of misclassified true calls. Regarding the accuracy, the results were again as expected. At 20% and 30% weight drop values, the technique increased the accuracy, while the higher values started decreasing it, due to the network losing too many weights to be able to learn enough features.

Overall, DropConnect proved to only be partly successful. The results indicate that when the technique is used, the AUROC values improve, while the accuracy suffers, which means that the model becomes better at correctly predicting true calls while losing on the ability to correctly predict for false calls when DropConnect is used. This is consistent with the other findings of the paper, as false calls present a high degree of intra-class variation, and with less learning parameters the model leads to the model predicting more instances to be true calls.

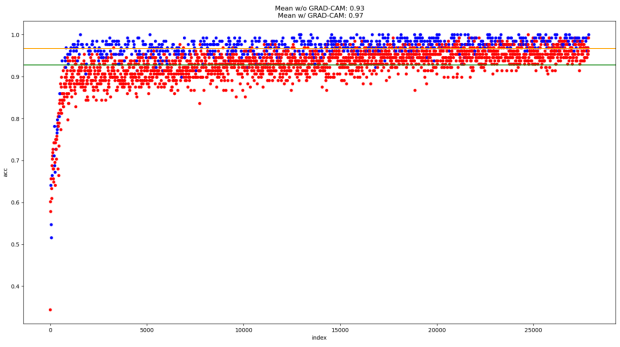


Figure 9: Accuracy on training batches for an entire run (augmented batches in red, non-augmented batches in blue, the yellow line represents the mean of the altered patches, the green line represents the mean of the unaltered patches)

Dropout proved to be rigid of a technique for the network. Even at the lower end value of 20% neuron dropout, the model already decreased significantly in its performance, and at over 30% the network did not converge.

C. Class Activation Maps-based Algorithms

1) Occlusion types: All the experiments that used a set value during the occlusion steps had weaker results than the baseline. The issue with using fixed values when occluding an image is that they alter the distribution of the original images. This causes overfitting to the altered patches, which in turn causes the accuracy of the model on the normal patches during the training phase to decrease. This behaviour can be seen in Fig. 9, where the average accuracy on the GRAD-CAM augmented batches is higher than the average accuracy on the non-augmented batches.

2) Visual Analysis of CAM algorithms: After looking through a number of augmented patches, some of which can be seen in Figure 21, the conclusions were as follows:

- Most patches containing true calls only have one important element in the capture. Occluding the element as the network finds it important leaves nothing but noise inside the image, which will not lead to correct predictions
- False calls can have a multitude of reasons for being detected: they can be true calls which were disabled for various reasons or they are noise picked up by the MFL sensors, in which case attention-based occlusion will most probably help, or they can be benign features such as metal plate welds, in which case attention-based occlusion might not be the ideal solution
- Overall, problematic areas for network in the case of false calls can be corrected, but attention-based occlusion is not a good answer for problematic areas in true calls, as they risk leaving nothing but noise in the image

3) Training instability: An interesting behaviour happened while experimenting with Algorithm 2. While the performance of the model increased when that training algorithm was used, the model became unstable, as a number of runs would not result in convergence for the network. An example of such behaviour can be seen in Figure 10.

A possible explanation for the phenomenon had to do with the domain spaces of the augmented patches. Since the occluded patches are part of a different domain than the original domain, the network tries to draw a class boundary that separates the classes by looking at both domains at the same time. If the domains of the two types of patches differ too much, the resulting total domain could be too far from the domain of the original patches, which would cause the algorithm to treat the original patches as outliers, and cause over-fitting towards the modified patches. Such a behaviour would be in line with the fact that in the runs where the network did not achieve convergence, the loss values for the unmodified training batches are consistently high, which indicates that the network might be over-fitting to the augmented patches.

Another argument towards the over-fitting theory is that, as it can be seen from Figure 10, the behaviour of the runs is the same for the first few hundred batches, after which it diverges. This indicates that while the gradient descent is performed in the same manner throughout all the runs, different initializations can cause the model to over-fit towards the augmented patches. In such case, it would mean that further experimentation with initialization techniques would be a possibility for solving the issue.

4) Fixed vs stochastic occlusion threshold: A separate issue was whether a fixed occlusion threshold would be the best choice for the model. As the results showed, the results keep improving until reaching the optimum occlusion threshold of 0.8. Such behaviour can be explained as follows: in theory, if the threshold is too low, such a large part of the images would be occluded that the resulting domain of the augmentations would be too different from the original domain to help the model generalize better. If the threshold is too high however, the two domain would simply overlap, which would again not help the model. However, between the two there should be an optimum point where the domain of the augmented and the original patches overlap each other in a way that would be useful for the model. The idea is then that a stochastic occlusion threshold, where each batch would have its misclassified instances occluded with a different threshold between 0.7 and 0.9, the resulting domain would occupy as much space as possible, and help the model generalize better. As such, an experiment using HiRes-CAM, training algorithm 2 and a stochastic occlusion threshold was set up. The result of this experiment can be seen in Table XIV. The idea proved unsuccessful however, as the result was overall worse than the experiments which used a fixed threshold.

D. Loss functions

1) L2-loss: The L2 loss had the weaker results than the baseline for all metrics, as it can be seen in Table XVI. As discussed before, the function works best when presented with infrequent large outliers. The poor performance indicates two possible causes: either the datasets contain a large number of frequent large outliers, or frequent small errors are present. This, combined with the analysis from the earlier experiments on domain shift shows that the dataset contained a large number of small outliers. This could also be seen through the difference created by adding even a small margin of 0.1, which was impressive, as even such a small margin lead to a decrease of almost 20% in the FPR-95% TPR rate. Even though the other metrics did not improve significantly, the FPR-95% TPR is a very sensitive metric, and its improvement shows that there is a large number of small outliers present in the dataset.

2) Added variance parameter θ: When discussing the θ parameter, a specific pattern emerged. When experiments using the θ parameter with the values -1, 1 were done, the network could no longer train properly, as the values computed by the loss function would spiral out of control during training. If the error on a given batch was already high, probably due to outliers, the added variance would cause the network to adjust the gradient descent by a lot in a single training step, which would cause the next batch to seem full of outliers. As such, the next batch would cause the network to adjust even more, and within a few steps the loss would reach values out of the storage capacity of the variable (in this case, 2^{31}). Therefore, the added variance through the θ parameter can only be as large as the maximum difference between the output of the network and the label, which in this case is 1.

X. Conclusions

The nature of the images used in this project created a number of unique challenges. The fact that the captures usually contain only a single feature meant that modifying the attention mechanism of the network via occlusion was only possible in certain cases, otherwise the image would only consist of noise. Another issue was the fact that the capture process would make most types of data augmentation create images which would be impossible to reproduce in a real setting. In conclusion, even though the signals coming from MFL sensors can be used as images, doing so creates images with a very specific topology and set of features.

Three of the issues identified through root cause analysis affected the generalization performance of the algorithm: the presence of label noise, high intra-class and low inter-class variance and lastly, domain data shift for some datasets. In this case, attention-based algorithms helped alleviate this issue by occluding the main features of the images which were mislabelled. The same approach also partially helped with the variance issue, although a larger part of the work was dedicated to solving the issue by using the proposed loss functions.

Overall, multiple approaches wielded positive results in terms of improving the generalization of the network. Attention-based algorithms proved the most effective, where a network using the algorithm 2 and HiRes-CAM, along with an occlusion threshold of 0.8 had an accuracy of 79.76, and an AUROC of 91.57, while the same

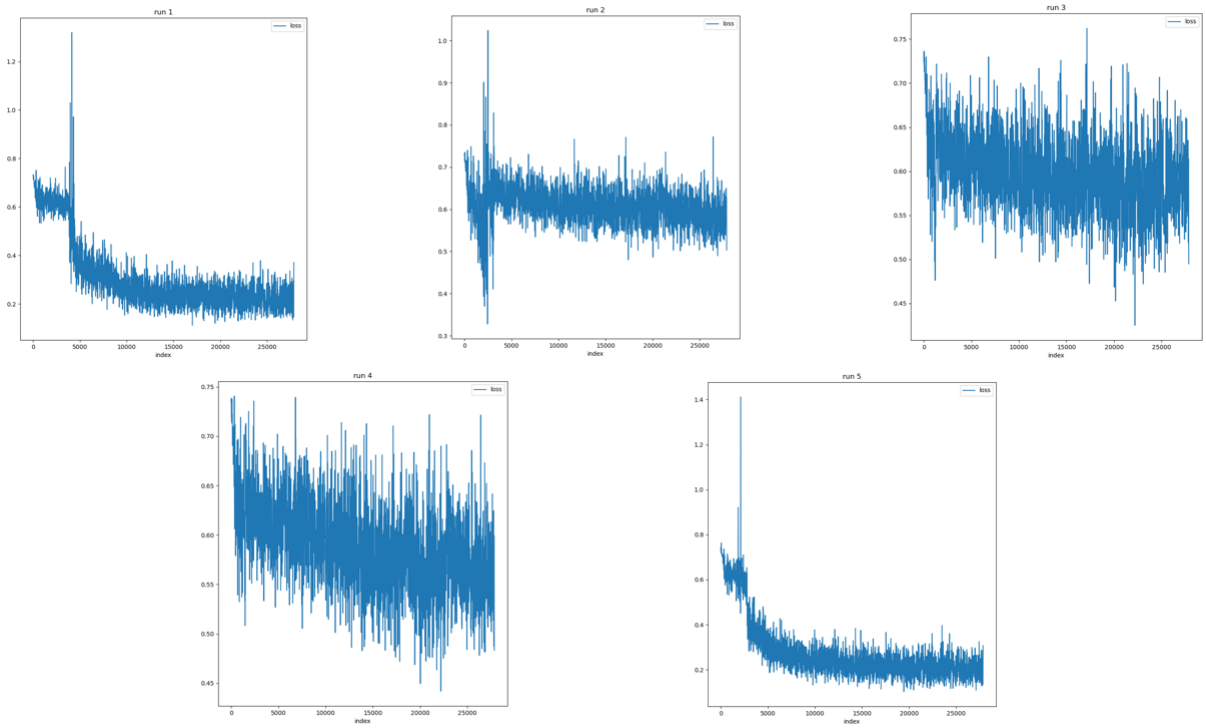


Figure 10: Loss values for training batches over five runs when Algorithm 2 was used. Runs 1 and 5 resulted in the model training successfully, runs 2, 3 and 4 resulted in the model not training properly, with the loss function not being able to optimize the weights during the process

algorithm configuration which used a threshold of 0.85 had the highest registered accuracy at 80.62.

The use of the novelty loss functions also yielded positive results, when compared to their already-existent counterparts. Adding a softening margin to the Softmax cross entropy and to the L2-loss function improved the results of the model on the accuracy and FPR-95% recall, although a constant decrease could be seen on the AUROC metric. Overall, an amount of softening around 0.2 to 0.25 offered the best results on the FPR-95% recall, while a softening factor of 0.3 offered the highest accuracy for both types of loss functions. Unfortunately, adding variance via the θ parameter to the loss functions worsened the results on all metrics.

Unfortunately, while the presented approaches obtained positive results, combining the more successful ones did not yield compounded improvements. Some of the approaches interfered with each other, which resulted in worse overall results, while others caused problems during the training phase, which made obtaining results not possible.

In the end, ROSEN decided to adopt the above-mentioned HiRes-CAM configuration into its project. Some examples of patches that were misclassified by the initial entry, but were correctly classified by the chosen HiRes-CAM augmented model can be seen in Figure 22. The most common types of improvements came from two types of patches: entries generated by sensors with high levels of noise and entries which were mislabeled by the operators, as described in the experiment on mislabeling.

XI. Future Work

The attention-based algorithm 2 proposed in this paper obtained the best results on all metrics for the experiments in this paper. There are however issues that still need to be solved regarding the algorithm. The most important issue is the initialization of the weights of the network. All the experiments using the algorithm used random initialization for the weights of the model. This resulted in the network not reaching convergence in 3 out of 5 runs on average for every fold. As such, future experiments should focus on the initialization techniques for the deep neural networks, such as He or Xavier normalization.

Not enough attention could be given to how combinations of the presented approaches could work together and whether they could improve the model further. Although combinations of the proposed loss functions and attention-based algorithms and combinations of stochastic regression and attention-based algorithms were tested, none of them proved successful in improving over their results when taken individually, with the former combination causing the network to crash during training. However, with the large number of possible combinations of configurations, there is potential in increasing the results obtained in this paper.

References

[1] L. Cartz, *Nondestructive Testing: Radiography, Ultrasonics, Liquid Penetrant, Magnetic Particle, Eddy Current*. ASM International, 1995.

[2] Flyability, "The Last Guide to NDT (Non-Destructive Testing) You'll Ever Need!," tech. rep.

[3] D. M. Amos, "Magnetic flux leakage as applied to aboveground storage tank flat bottom tank floor inspections," *Materials Evaluation*, vol. 54, 1 1996.

[4] H. Aldosari, R. Elfouly, and R. Ammar, "Optimal Artificial Neural Network Model for Prediction of Oil and Gas Pipelines Defect Length," *Proceedings - 2020 International Conference on Computational Science and Computational Intelligence, CSCI 2020*, pp. 1457–1462, 12 2020.

[5] J. Liu, Y. Ma, F. Qu, and D. Zang, "Semi-supervised Fuzzy Min–Max Neural Network for Data Classification," *Neural Processing Letters*, vol. 51, pp. 1445–1464, 4 2020.

[6] A. Mohamed, M. S. Hamdi, and S. Tahar, "A Machine Learning Approach for Big Data in Oil and Gas Pipelines," *Proceedings - 2015 International Conference on Future Internet of Things and Cloud, FiCloud 2015 and 2015 International Conference on Open and Big Data, OBD 2015*, pp. 585–590, 10 2015.

[7] "ISO.IEC 31010:2009 – Risk management – Risk assessment techniques," standard, International Organization for Standardization, Geneva, CH, june 2009.

[8] D. Morales, E. Talavera, and B. Remeseiro, "Playing to distraction: towards a robust training of CNN classifiers through visual explanation techniques," *Neural Computing and Applications*, vol. 33, pp. 16937–16949, 12 2020.

[9] A. D. R. V. D. P. D. B. Ramprasaath R. Selvaraju, Michael Cogswell, "Grad-cam: Visual explanations from deep networks via gradient-based localization," 2016.

[10] R. L. Draelos and L. Carin, "Use HiResCAM instead of Grad-CAM for faithful explanations of convolutional neural networks," 11 2020.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[12] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of Neural Networks using DropConnect," *Proceedings of the 30th International Conference on Machine Learning*, pp. 1058–1066, 5 2013.

[13] Q. Nguyen, H. Valizadegan, and M. Hauskrecht, "Learning classification models with soft-label information," *Journal of the American Medical Informatics Association*, vol. 21, pp. 501–508, 5 2014.

[14] G. Dobmann, G. Walle, and P. Höller, "Magnetic leakage flux testing with probes: physical principles and restrictions for application," *NDT International*, vol. 20, no. 2, pp. 101–104, 1987.

[15] A. S. for Nondestructive Testing, *Nondestructive Testing Handbook, Third Edition*, vol. 5.

[16] R. Group, "Tbit - tank bottom inspection tool by rosen,"

[17] A. Khodayari-Rostamabad, J. P. Reilly, N. K. Nikolova, J. R. Hare, and S. Pasha, "Machine learning techniques for the analysis of magnetic flux leakage images in pipeline inspection," *IEEE Transactions on Magnetics*, vol. 45, pp. 3073–3084, 9 2009.

[18] Y. Lijian, L. Gang, Z. Guoguang, and G. Songwei, "Oil-gas pipeline magnetic flux leakage testing defect reconstruction based on Support Vector Machine," *2009 2nd International Conference on Intelligent Computing Technology and Automation, ICICTA 2009*, vol. 2, pp. 395–398, 2009.

[19] J. Liu, M. Fu, F. Liu, J. Feng, and K. Cui, "Window feature-based two-stage defect identification using magnetic flux leakage measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, pp. 12–23, 1 2018.

[20] J. Liu, M. Fu, Z. Wu, and H. Su, "An ELM-based classifier about MFL inspection of pipeline," *Proceedings of the 28th Chinese Control and Decision Conference, CCDC 2016*, pp. 1952–1955, 8 2016.

[21] Y. Ren, J. Liu, X. Yu, and H. Zhu, "A general inversion method based on magnetic flux leakage inspection," *Proceedings of 2019 IEEE 8th Data Driven Control and Learning Systems Conference, DDCLS 2019*, pp. 483–487, 5 2019.

[22] C. Summers and M. J. Dinneen, "Improved Mixed-Example Data Augmentation," tech. rep.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 2818–2826, 12 2016.

[24] A. J. Storkey, "When Training and Test Sets are Different: Characterising Learning Transfer 1 Overview," 2013.

[25] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[26] N. R. Pearson, M. A. Boat, R. H. Priewald, M. J. Pate, and J. S. D. Mason, "A study of MFL signals from a spectrum of defect geometries," 2012.

[27] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: BEYOND EMPIRICAL RISK MINIMIZATION,"

APPENDIX

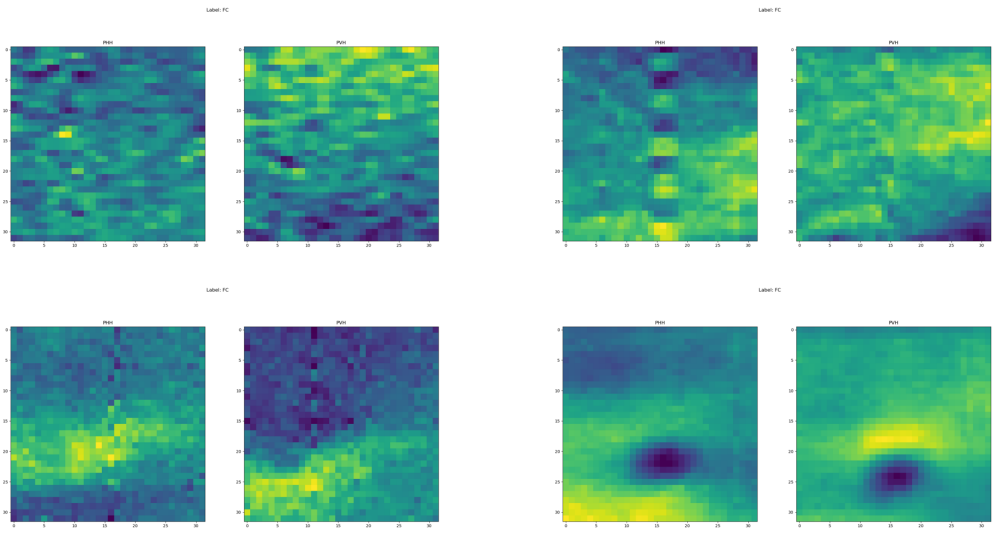


Figure 11: Examples of entries which are false positives - called ”false calls” within the context of the project - of the detection algorithm

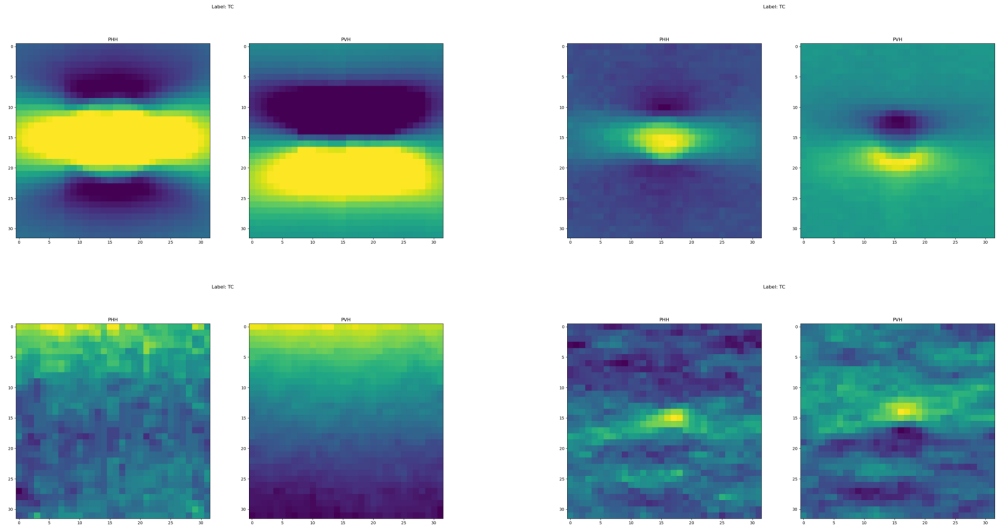


Figure 12: Examples of entries which are true positives - called ”true calls” within the context of the project - of the detection algorithm

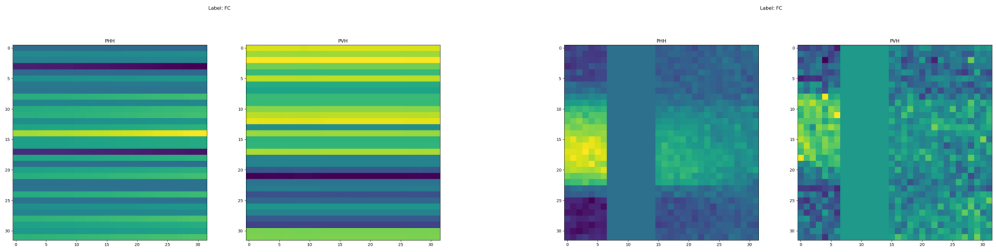


Figure 13: Examples of anomalous entries present due to various sensor issues

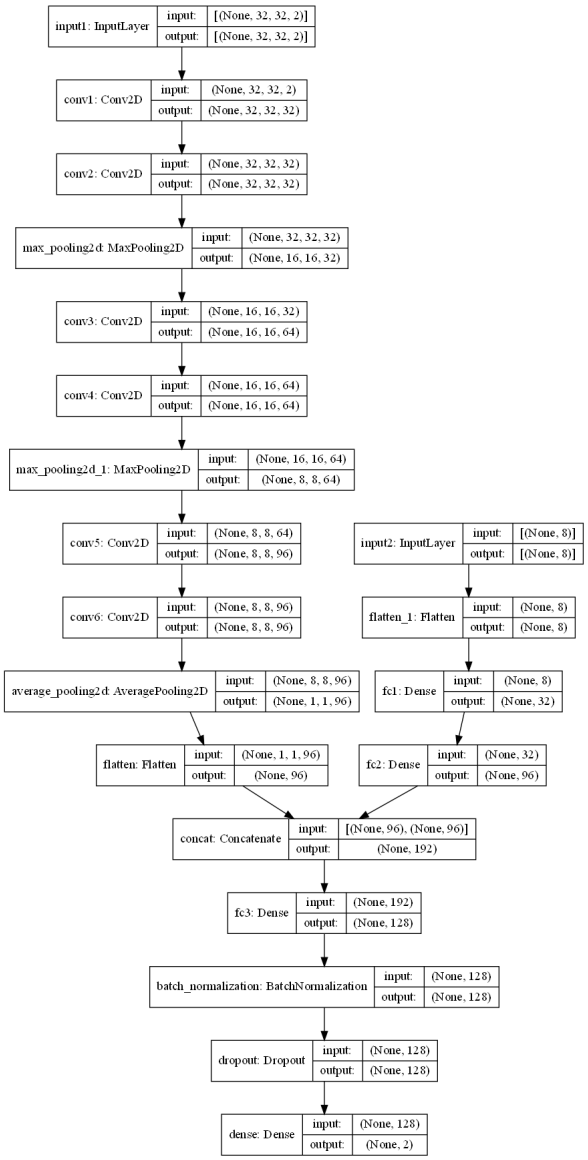


Figure 14: The baseline architecture of the neural network utilized in this project

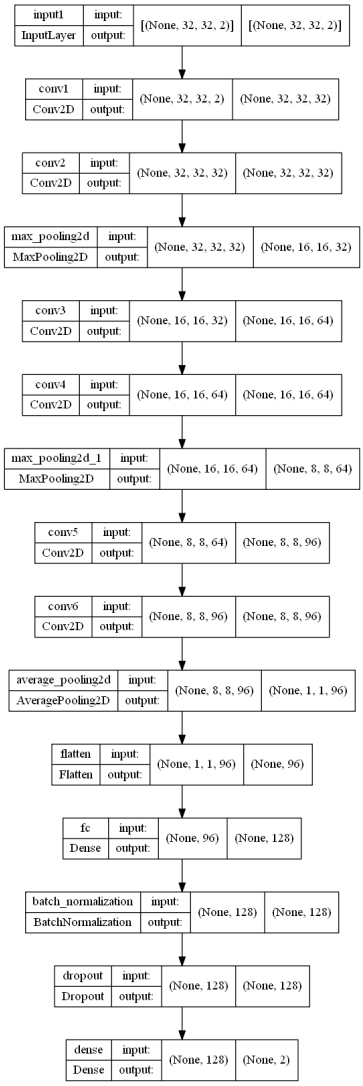


Figure 15: The alternative architecture proposed for the neural network utilized in this project

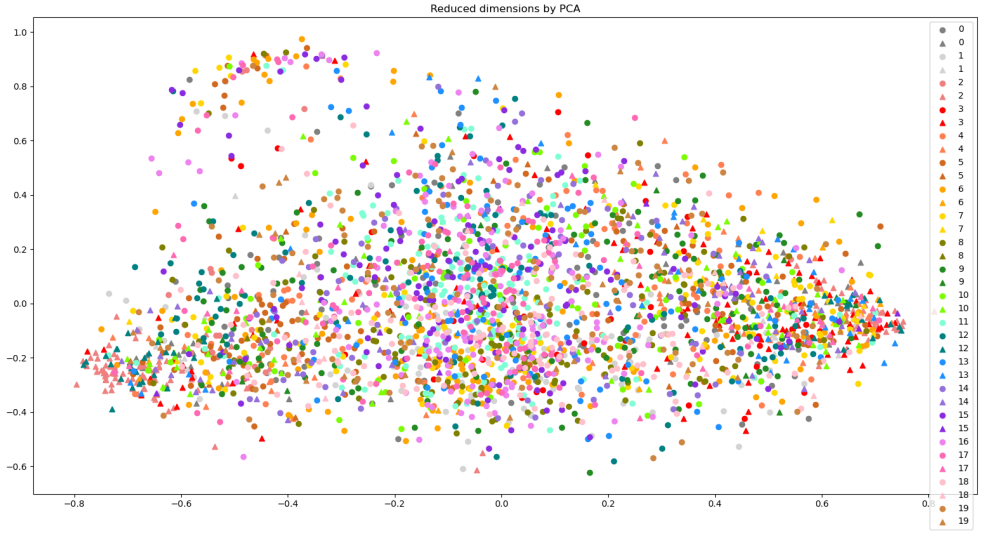


Figure 16: PCA result for all 20 datasets, each color represents a different dataset, the triangles represent true positive instances, the circles false positive instances)

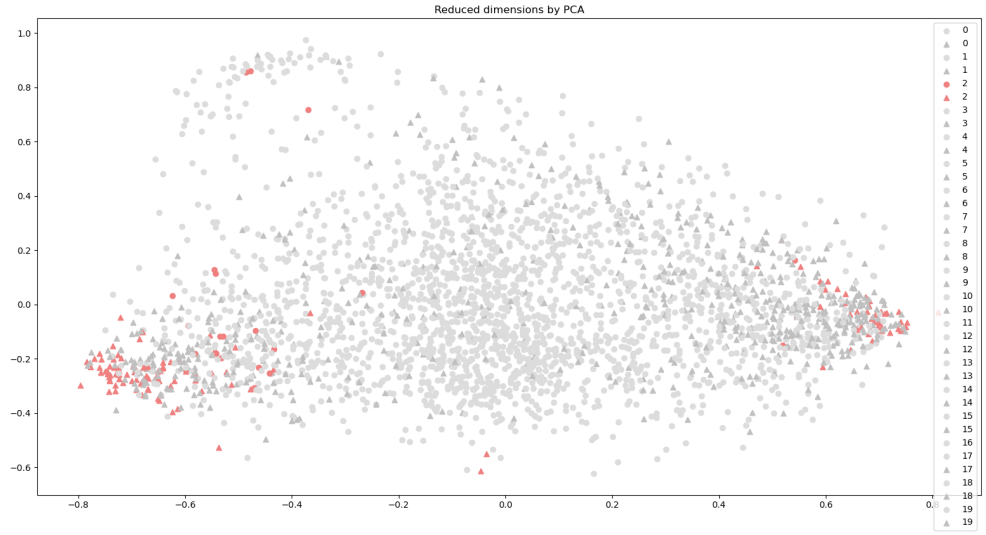


Figure 17: PCA result for the calibration plates dataset - standard calibration, the dataset is represented by the coral pink instances, the other datasets are represented by the gray instances

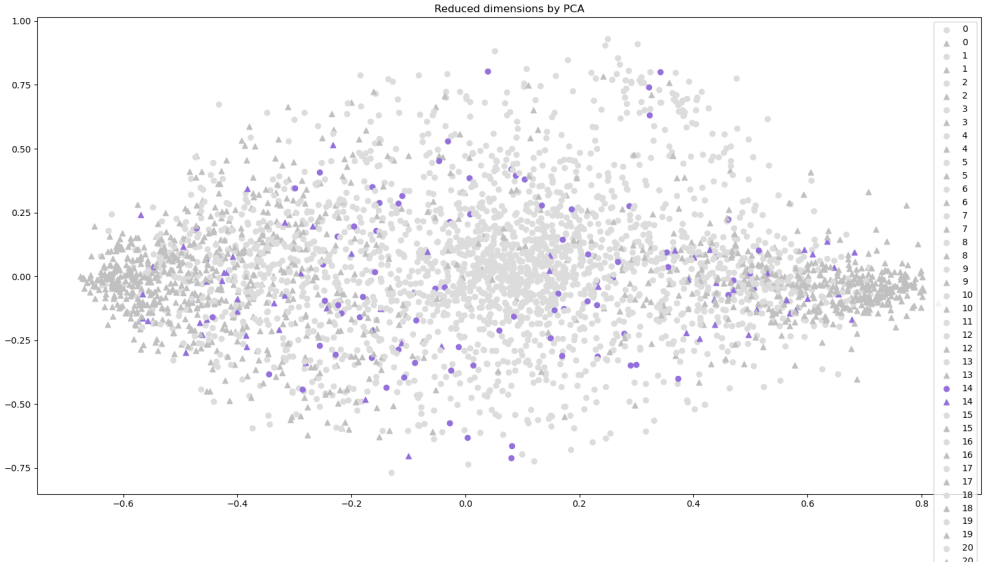


Figure 18: PCA result for dataset number 4 - ultra calibration, the dataset is represented by the purple instances, the other datasets are represented by the gray instances

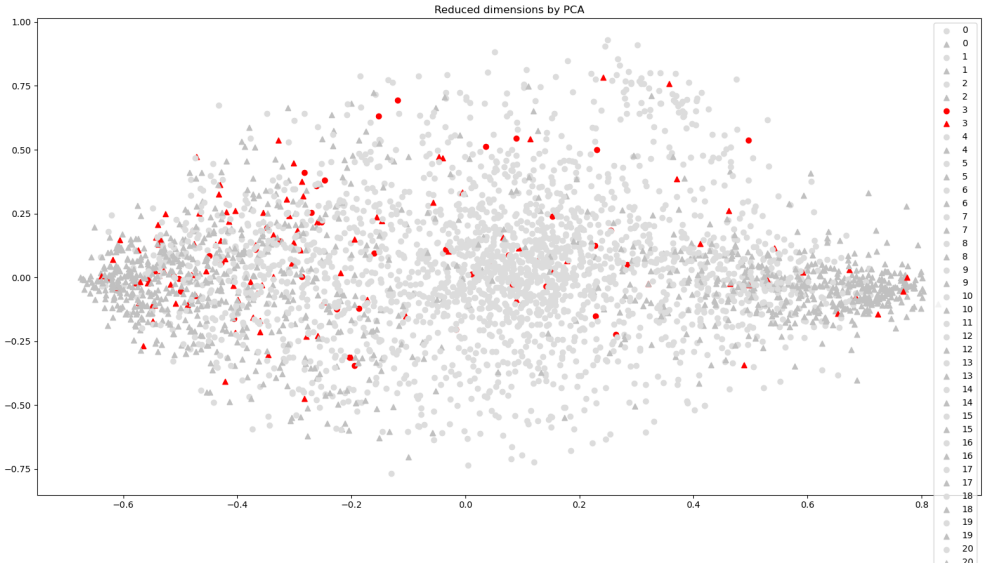


Figure 19: PCA result for dataset number 3 - standard calibration, the dataset is represented by the purple instances, the other datasets are represented by the gray instances

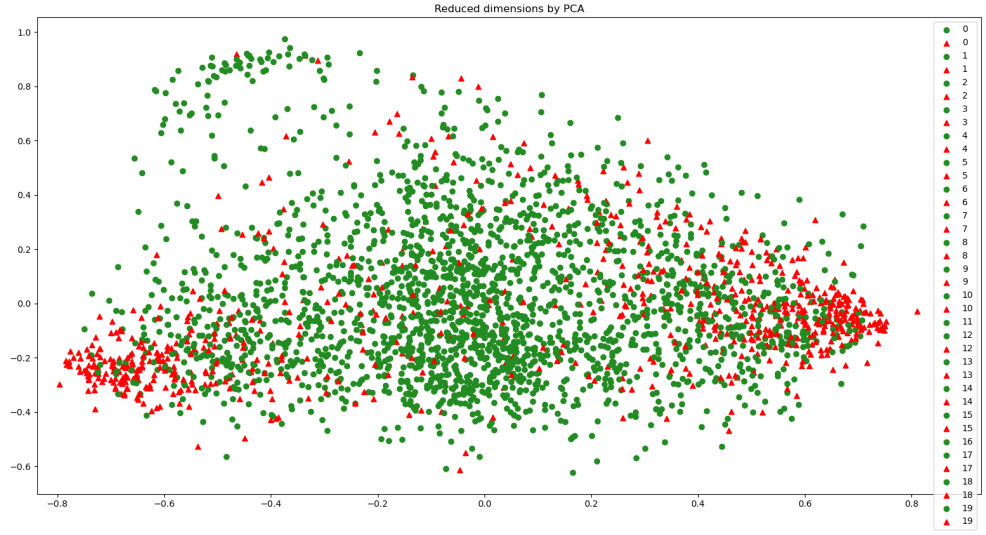


Figure 20: PCA result with class distributions highlighted, the true calls are represented by the red triangles, the false calls by the green circles

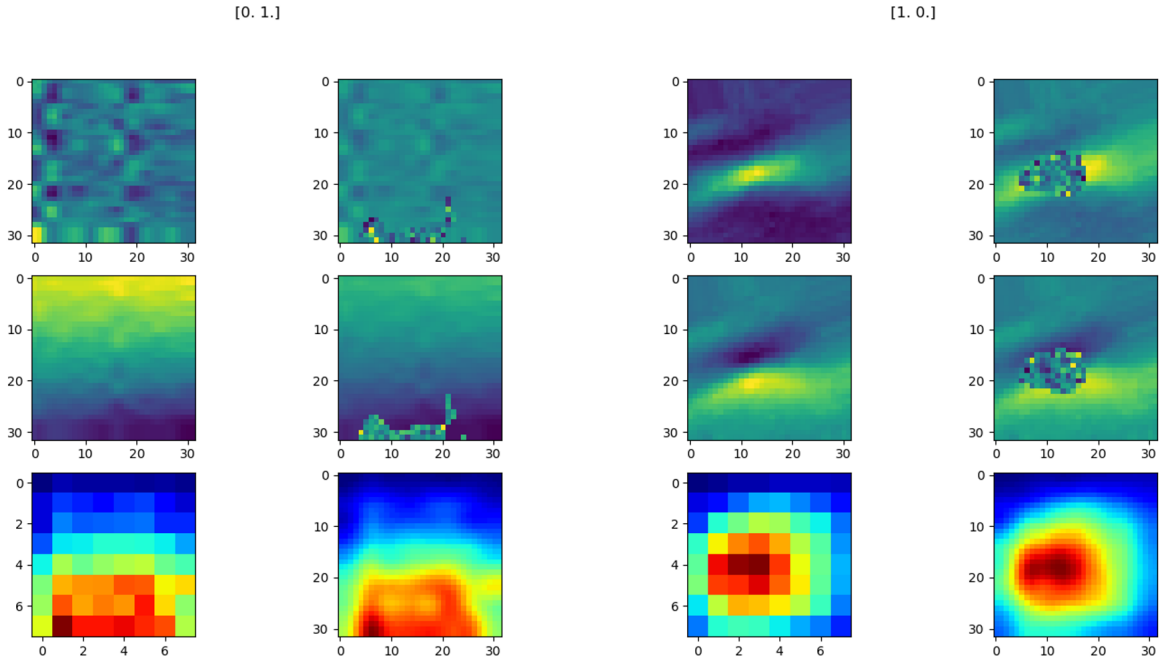


Figure 21: Examples of patches augmented using GRAD-CAM, the first one shows a false positive being modified, the second one shows a true positive being modified

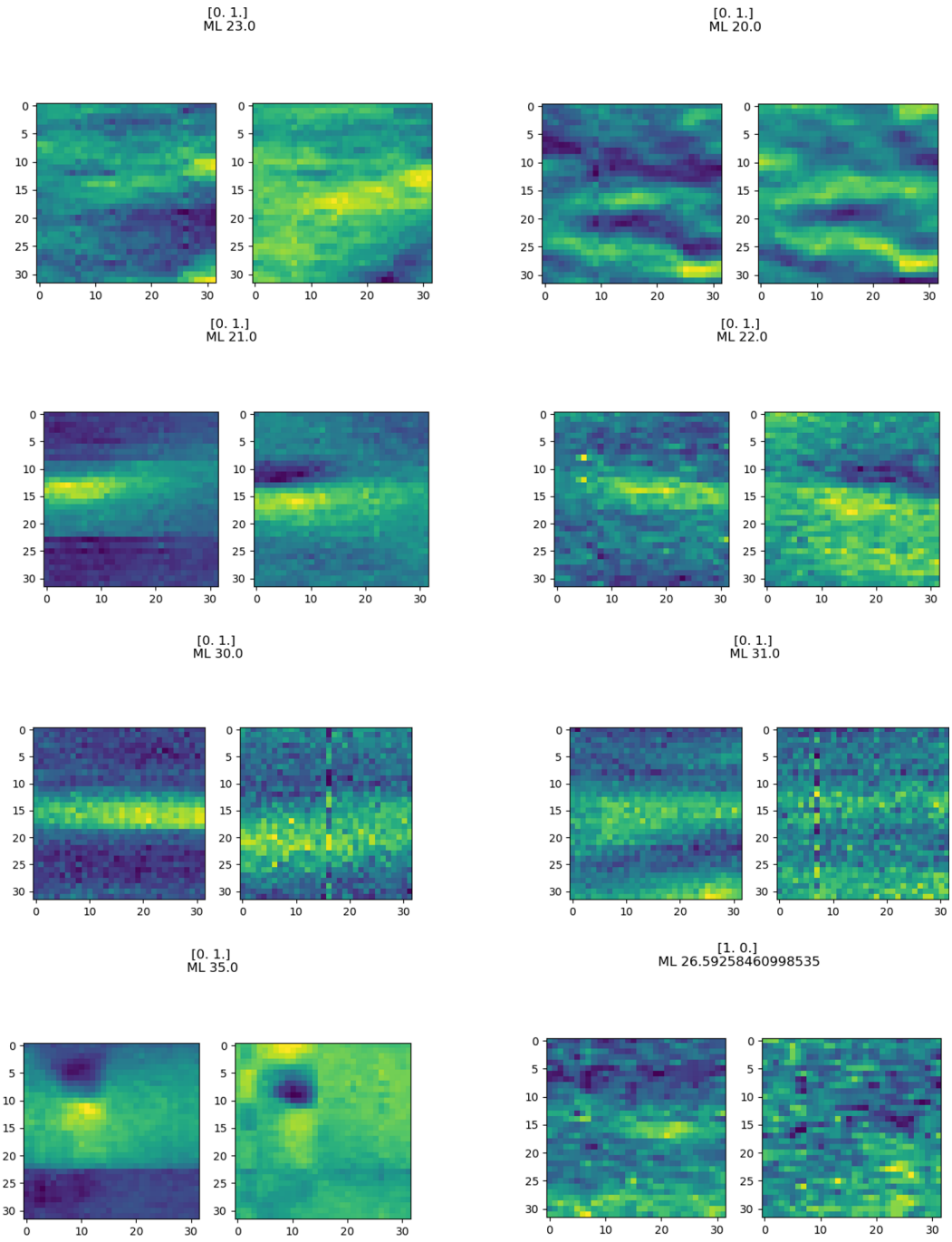


Figure 22: Examples of entries which were mislabelled by the baseline model, but were correctly classified by the improved model