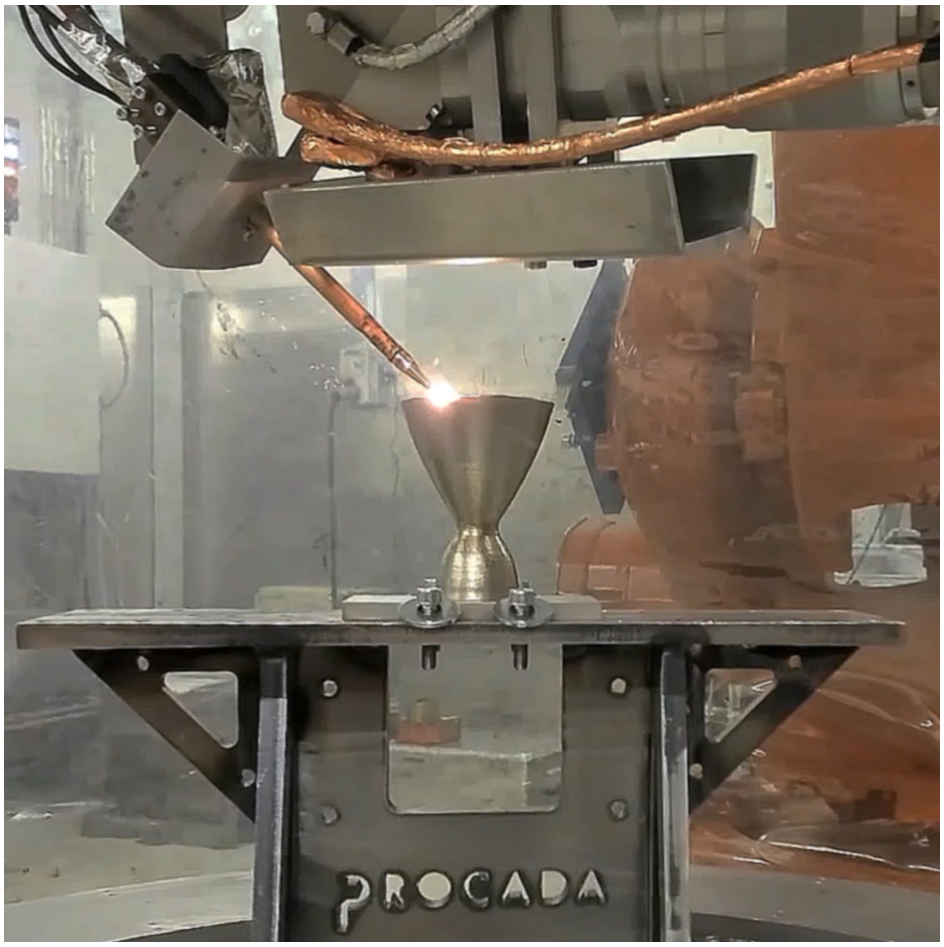
CHALMERS
UNIVERSITY OF TECHNOLOGY



Data-Driven Estimation of Droplet Formation for Conductance Control in Wire-Based Metal Additive Manufacturing

Degree project report in physics

HARALD DE BLANCHE

DEPARTMENT OF PHYSICS AND ASTRONOMY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Degree project report 2026

Data-Driven Estimation of Droplet Formation for Conductance Control in Wire-Based Metal Additive Manufacturing

Harald de Blanche



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics and Astronomy
Chalmers University of Technology
Gothenburg, Sweden 2026

Data-Driven Estimation of Droplet Formation for Conductance Control in Wire-Based Metal Additive Manufacturing
Harald de Blanche

© HARALD DE BLANCHE, 2026.

Supervisor: Petter Hagqvist, Procada
Examiner: Anders Hellman, Physics

Degree project report 2026
Department of Physics and Astronomy
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Photograph of Procada's robot building a nozzle prototype.

Gothenburg, Sweden 2026

Abstract

This work studies a wire-based metal additive manufacturing process where the distance between the wire and the part being built is controlled indirectly through electrical conductance. The measured conductance is compared to a setpoint, and the wire is raised or lowered depending on whether the conductance is too high or too low. At present, this setpoint is adjusted manually by an operator monitoring the process. The aim of this work is to investigate whether the conductance setpoint can instead be determined automatically, using models that estimate the probability of droplet formation from measured process data. Several approaches are evaluated, including linear models based on PCA as well as convolutional neural networks. The linear models show some correlation with droplet formation but are generally noisy. The convolutional neural network models show promising preliminary results, although further work is required before it can be used as the basis for a control algorithm. In addition, a simple control algorithm based only on observed drops, rather than predictions of future drops, was evaluated. The algorithm was tested in only one experiment, so the results should be considered preliminary. Nevertheless, the results indicate that the algorithm was able to control the process with a similar performance to that of a human operator.

Keywords: Additive manufacturing, laser metal deposition with wire, conductance control, droplet formation, process monitoring, convolutional neural networks, machine learning, real-time control.

Acknowledgements

This project was made possible, and considerably more enjoyable, thanks to the good company of the employees at Procada. In particular, I would like to thank David Johansson, who was a great help in setting up and performing the experiments in this project. I would also like to thank Petter Hagqvist, CTO at Procada and my supervisor, who contributed a great deal of his experience, helped me understand the different challenges encountered during the project, and suggested a wide range of inventive solutions. Finally, I would like to thank my examiner, Anders Hellman, Professor of Chemical Physics at Chalmers, for his guidance and feedback throughout the project.

Harald de Blanche, Gothenburg, September 2026

List of Acronyms

Below is a list of acronyms used throughout this thesis, listed in alphabetical order:

AM	Additive Manufacturing
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
LMD-w	Laser Metal Deposition with Wire
PCA	Principal Component Analysis
PLS	Partial Least Squares

Nomenclature

Below is the nomenclature for the parameters and variables most central to this thesis.

Variables

G	Conductance between the wire and the surface
G_0	Conductance setpoint used by the control system
G_{lim}	Estimated conductance limit associated with drop formation
d	Distance between the nozzle and the surface
R	Electrical resistance between the nozzle and the surface
v	Wire feed rate
w	Input window length
d_{far}	Minimum distance from a detected drop for a far-from-drop window
d_{short}	Distance between the end of a close-to-drop window and the following drop
dt_{drop}	Maximum time between consecutive drops for treating them as part of the same drop sequence
s_{fk}	Size of the first convolutional kernel
μ	Estimated mean value of the conductance limit
P	Variance of the estimated conductance limit
σ	Standard deviation for the conductance limit for modelling drops
z_k	Normalized distance between the estimated conductance limit and the setpoint
$\alpha(z_k)$	Correction term used in the context-aware filter
k	Steepness parameter for the algorithms, ($k_{\text{drop/no drop}}$ used for intuitive algorithm)
$t_{0,\text{drop}}$	Time parameter of the damping function following a drop
$t_{0,\text{no drop}}$	Time parameter of the damping function when no drop is observed
t_{ramp}	Rise time of the ramp used for ramp-based outputs
y_{ramp}	Ramp-based target describing proximity to a drop
y_{density}	Density-based target describing the local frequency of drops
y_{log}	Log-transformed and normalized density-based target

Contents

List of Acronyms **ix**

Nomenclature **xi**

List of Figures **xv**

List of Tables **xvii**

1 Introduction **1**
1.1 Background . . . 1
1.2 Problem description . . . 1
1.3 Purpose . . . 2
1.4 Goals . . . 3
1.5 Limitations . . . 3

2 Theory **5**
2.1 The conductance setpoint . . . 5
2.2 Dataset reduction . . . 5
2.3 Convolutional neural networks . . . 7

3 Methods **9**
3.1 Datasets . . . 9
3.1.1 Slow dataset . . . 11
3.1.2 Fast dataset . . . 11
3.1.3 Image data . . . 12
3.2 Targets . . . 12
3.3 Models . . . 13
3.3.1 Linear model . . . 14
3.3.2 Linear classifier . . . 15
3.3.3 CNN model . . . 16
3.3.4 Model evaluation . . . 18
3.4 Control algorithms . . . 18
3.4.1 Simulations . . . 19
3.4.2 Algorithm design . . . 19
3.4.3 Experiment . . . 22
3.5 Visualisations . . . 22
3.5.1 PCA and PLS plots . . . 22

3.5.2 FFT spectrograms . . . 22
3.5.3 Image analysis . . . 23

4 Results and discussion **25**
4.1 Visualisations . . . 25
4.1.1 PCA and PLS plots . . . 25
4.1.2 FFT spectrogram . . . 27
4.1.3 Image analysis . . . 28
4.2 Linear model . . . 28
4.2.1 R^2 score . . . 28
4.2.2 Ordered accuracy . . . 29
4.2.3 Summary . . . 29
4.3 Linear classifier . . . 33
4.4 CNN . . . 34
4.5 Algorithm performance . . . 37
4.5.1 Simulation . . . 37
4.5.2 Real process . . . 42

5 Conclusion and outlook **45**
5.1 Future work . . . 46

Bibliography **47**

List of Figures

1.1 The left image shows the machine working at a proper distance from the surface and the wire melted in the melt pool. The middle shows the machine working too close to the surface resulting in stubbing. The right image shows the machine working too far from the surface resulting in a weak link, and eventually drops. 2

2.1 Schematic of the general structure of a CNN. The structure shown consists of multiple sets of convolution and pooling layers, with the convolution layers consisting of multiple kernels. The convolution and pooling layers could be repeated to create a deeper network. 8

3.1 Figure showing the mean time between data points for each file in the slow dataset. For each file, the shortest and longest 1% of the intervals were removed to eliminate outliers. The figure shows that the mean sampling interval ranges from 0.10 to 0.13 s. 10

3.2 Figure showing the time between data points for a typical file in the slow dataset. The figure shows that the sampling rate varies within each file, ranging from 0.1 to 0.13 s with some longer outliers. 10

3.3 Figure showing the pattern of time between data points in the fast dataset. The times range from 0.000248 to 0.000252, roughly corresponding to 4 kHz. This pattern was repeated throughout each file in the fast dataset with some small variations. 12

3.4 Figure showing an example from the image dataset. The brightest part of the image is the weld pool. The red part to the left of the melt pool is the formed bead. 13

4.1 All data points projected on the first two principal components. To the left data points far from drops are projected, in the centre data points shortly before drop and to the right data points shortly after drop. 25

4.2 All data points projected on the two first latent variables constructed using the same data. To the left data points far from drops are projected, in the centre data points close before drop and to the right data points close after drop. 26

4.3 Data points from a set of test files projected on the two first latent variables constructed using the data from different files. To the left data points far from drops are projected, in the centre data points close before drop and to the right data points close after drop. 27
4.4 Comparison of the conductance signal and a large range of the frequency spectrum in a region between two drops. 27
4.5 Comparison of the conductance signal and the lower part of the frequency spectrum in a region between two drops. 28
4.6 R^2 scores for all input methods and both output targets. 30
4.7 Ordered accuracy scores for all input methods and both output targets. 31
4.8 Accuracy and sensitivity for single- and three-layer CNN as a function of window, w and size of the first kernel. The blank cell in the bottom right of each plot is caused by the fact that the size of the kernel cannot be larger than the window size. 35
4.9 Accuracy and sensitivity for two single-layer CNNs using a window size of 100 and 2000 as a function of d_{short} and d_{far} 36
4.10 Simulation results for the intuitive algorithm using different values of $t_{0,\text{no drop}}$ and $k_{\text{no drop}}$. Each case is compared with the reference parameters $t_{0,\text{no drop}} = 500$ ms and $k_{\text{no drop}} = 0.0014$. 38
4.11 Damping coefficient as a function of time since the most recent drop for different values of $k_{\text{no drop}}$ compared to $k_{\text{no drop}} = 0.0014$. 39
4.12 Behaviour of the algorithm for different values of $k_{\text{no drop}}$ compared to $k_{\text{no drop}} = 0.0014$. 39
4.13 Reaction to drop in threshold for different values of $k_{\text{no drop}}$ compared to $k_{\text{no drop}} = 0.0014$. 39
4.14 Simulated setpoint for time intervals close to 70 ms when no buffer is applied. 41
4.15 Simulated setpoint for time intervals close to 70 ms when a buffer is applied. 41
4.16 Comparison of the simulated setpoint with and without the buffer for a time interval of 500 ms. A value of one corresponds to the buffer being applied. 42
4.17 Illustration of the experiment performed to evaluate the algorithms. The vertical grey lines indicate drops and the dashed lines indicate different sections of the experiment. The conductance varies close to 15 S both when controlled by an operator and the time interval algorithm. Shortly after the intuitive algorithm was activated the experiment crashed. 42
4.18 Plot of the area of the melt pool over time. The dashed vertical line indicates when the time interval algorithm was activated. Shortly before the algorithm was activated the melt pool decreases in size and the area is more stable. 43
4.19 Illustration of the drop positions within each layer during the experiment. The marker indicates when the time interval algorithm was activated. The x-axis represents the normalized position within each layer, from the beginning to the end of the layer. 43

List of Tables

3.1 Overview of the datasets used in the project. 9
3.2 Parameters used for the damping coefficient. 21

4.1 Mean and standard deviation of the order accuracy across all test files for the different inputs and outputs. 29
4.2 Classification results for each file used as test data. For each file the sensitivity, precision, false positive rate, balanced accuracy and class balance are presented. 33
4.3 Sensitivity, precision, false positive rate, balanced accuracy and class balance for the different CNN configurations. All values are given in percent. 36
4.4 Simulated drop frequency for the intuitive algorithm. 37
4.5 Simulated drop frequency for the time interval algorithm without a buffer. 41
4.6 Simulated drop frequency for the time interval algorithm with a buffer. 42
4.7 Drop statistics for the different process periods. 44

1
Introduction

1.1 Background

This project is based on addressing a problem within additive manufacturing (AM) and was carried out in collaboration with Procada, a company operating within this field. Procada is a relatively new company aiming to sell and service machines for laser metal wire deposition additive manufacturing (LMD-w AM). The company is currently working to demonstrate the efficiency and stability of its machines by producing different prototypes. However, LMD-w AM also has the potential to be used for manufacturing finished components in applications where material properties and process stability are critical, such as aerospace, while using a relatively low amount of material [1]. This project aims to investigate the possibility to further automate their process, removing, or at least reducing the need for supervision of the process. Automating the process has a number of potential advantages. It could lead to a more stable process and consistent build quality. However the main advantage would be a substantially increased industrial applicability [2].

1.2 Problem description

LMD-w is based on feeding wire into a laser which melts the wire, forming a melt pool on a substrate. As the melt pool solidifies, a bead is formed. This bead makes up the first layer of the component. Then the process is repeated in layers, adding beads to the previous layer and thereby building the component layer by layer [3]. One important factor for achieving good material properties is the distance, d, between the nozzle and the surface [4],[5]. In this report, the term surface refers to either the substrate or the previously deposited layer. The optimal scenario is that the wire melts inside the melt pool, but before reaching the surface, as is seen to the left in Figure 1.1[5], [6]. However, if the machine is working too close to the surface, the wire will not melt in time and we get what is known as stubbing, seen in the centre of Figure 1.1. Here unmelted wire will scrape along the surface and new material will be added on top, leading to insufficient fusion between the layers, meaning worse material properties and can lead to cracks forming in the finished product [7]. Lastly, if the machine works too far from the surface we get a weak link between the wire and the melt pool, seen to the right in Figure 1.1. If the distance is increased further, drop formation may occur. In this state, the wire melts outside the melt pool and material is deposited onto the surface in the form of drops. The process can often be recovered before full drop formation occurs. Therefore, a short

disconnection does not necessarily reduce the material properties, but it can lead to an uneven surface[8]. Building on an uneven surface is more difficult than building on a smooth surface, which increases the risk of stubbing or drop formation in the following layer[4]. In this report, this state will be referred to as a drop state, even if full drop formation does not always occur.

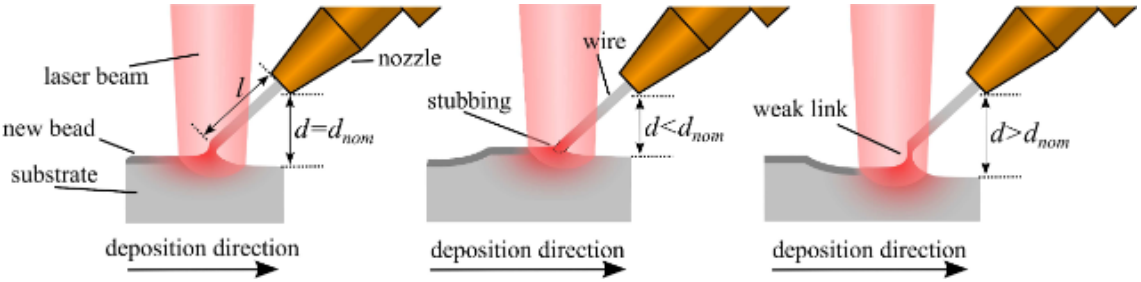


Figure 1.1: The left image shows the machine working at a proper distance from the surface and the wire melted in the melt pool. The middle shows the machine working too close to the surface resulting in stubbing. The right image shows the machine working too far from the surface resulting in a weak link, and eventually drops. Adapted from [5] with permission.

Given the importance of the distance for the process it may be assumed that the machine is programmed to aim for a certain distance. At Procada this is not the case. Instead the machine aims for a certain conductance between the nozzle and the substrate [9]. It has been shown that the distance can be estimated by looking at the wire feed rate and the resistance, resulting in a similar error margin to a laser scanning system when positioning accuracy is taken into account [5]. Further the wire feed rate is mostly constant throughout a run meaning that it is enough to look at the resistance. Finally, Procada has found that looking at the conductance, instead of the resistance, results in better control of the distance. When the wire is touching the surface, the conductance will be very high and when the wire leaves the melt pool it will be zero. Procada programs the machine to aim for a conductance setpoint corresponding to a state in-between, making sure the wire melts inside the melt pool. Process parameters that might affect the conductance setpoint could change with operating conditions or over time [10]. Therefore the conductance setpoint currently requires manual supervision and adjustment, which limits the level of automation in the process [2]. This project investigates whether data from previous process runs can be used to estimate the process state and support real-time adjustment of the conductance setpoint.

1.3 Purpose

The purpose of this project was to investigate whether data from previous process runs could be used to support real-time adjustment of the conductance setpoint. The aim was to reduce the need for manual supervision by evaluating a model-based control algorithm for maintaining the process within a stable operating window.

1.4 Goals

The project had two main goals. The first goal was to develop and evaluate a model capable of assessing whether the current conductance setpoint was too high or too low, based on how close the process was to a drop state. The second goal was to design and evaluate an algorithm that used the model output to adjust the setpoint in real time, with the aim of keeping the process within a suitable operating window between stubbing and drop formation.

1.5 Limitations

Procada is a relatively small company. Although access to previous data and the possibility of conducting new experimental runs were available, paid customer projects were prioritized, which limited the ability to perform experiments within the project. Since Procada is an evolving company, data from previous runs has not always been saved in the same way, and new parameters have been added over time. Consequently, there is a trade-off between the number of parameters included in the analysis and the number of process logs available for consistent comparison.
Finally, due to a lack of time and access to computational power, methods that were computationally demanding were not investigated in as much detail as desired. This mainly concerns the different CNN models that were evaluated, the image-based data was only investigated for a small set of data due to similar issues.

2
Theory

2.1 The conductance setpoint

The control system for Procada's machine is based on a conductance setpoint. As briefly explained in the background, controlling the conductance can help maintain the process within the operating window between stubbing and drop. It has been shown that the distance, d, can be estimated based on a linear model of the resistance, R, between the nozzle and the surface, and the wire feed rate, v [5]. The distance is estimated with the following relation:

$$d = \theta_0 + R\theta_1 + v\theta_2 + \sqrt{Rv}\theta_3. \tag{2.1}$$

If we assume a constant wire feed rate, we can estimate d as a linear model based on R and $\sqrt{R}$. However, this linear model would only be stable for distances close to the nominal distance. As the process approaches a drop state, the resistance tends toward infinity, and the linear model is no longer valid. Since the machine operates in a wide range of distances in real conditions, occasionally reaching both stubbing and drop, this model is not useful for real process applications. Instead Procada assumes that the distance varies linearly with the conductance, G, for the entire melt pool. Within the scope of this project, no literature source supporting this relationship was found. However, that is the linear assumption used by Procada based on their experience [9]. In contrast to the resistance model, the distance increases as the conductance decreases. This can be interpreted as stubbing corresponding to a high conductance and drop to the conductance approaching zero. Since conductance is bounded between zero and a maximum value determined by the process conditions, Procada approximates the intermediate region as linear. Since stubbing is generally more harmful to the process than short disconnections associated with a drop state, the desired operating point is not necessarily centred between the two failure modes. Instead, the setpoint is selected closer to the drop boundary in order to reduce the risk of stubbing.

2.2 Dataset reduction

Initially a solution including a model as small as possible was investigated. A smaller model has two main advantages that were considered in this project. First, it is easier to interpret, meaning one can understand what the model is looking at in order to make its prediction. Second, a smaller model has a smaller risk of overfitting, which

was relevant since the quality of the data was uncertain. The different models and datasets that were used as inputs are described in detail in section 3.1 and 3.3. However reducing the size of the model was only considered for the slow dataset, a dataset containing all the information the machine knows about the process, updated at a low frequency. In order to reduce the size of a linear model, the dataset that the model is applied to needs to be reduced. This was done using principal component analysis (PCA) and partial least squares (PLS).

PCA

For this project PCA was used to reduce the dataset. PCA is based on creating principal components, orthogonal linear combinations of the original dataset. These principal components are designed to maximize variance, meaning that as much variance as possible from the original dataset is transferred to the reduced one [11]. An arbitrary dataset X of size $n_p \times n_s$ consisting of n_p parameters and n_s samples could be reduced to a smaller dataset X_{PCA} of size $n_{PCA} \times n_s$, where n_{PCA} is the number of principal components used. The principal components are defined as the eigenvectors of the covariance matrix from the original dataset X [11]. The eigenvalues of these eigenvectors are equivalent to the variance of the corresponding principal components. The first principal component is the eigenvector with the highest eigenvalue, and so on. The reduced dataset can be obtained either by selecting a fixed number of principal components or by selecting the smallest number of components that explain a chosen fraction of the total variance.

PLS

PLS is a method that, similarly to PCA, reduces the dataset by constructing latent variables as linear combinations of the original variables. Unlike PCA, where the directions are chosen to maximize the variance in the input data, PLS constructs latent variables that maximize the covariance between the input data and a target [12].
For an input matrix X, the latent variables can be written as

$$t = Xw, \tag{2.2}$$

where w defines a direction in the input space. If Y contains several variables, a corresponding latent variable in the target space can be written as

$$u = Yc. \tag{2.3}$$

The directions w and c are chosen such that the covariance between t and u is maximized. New latent variables are then constructed successively from the remaining information in the data. The successive directions in the input space, w_i, are constructed to be orthogonal, whereas the Y-score vectors u_i are not necessarily orthogonal [13].

2.3 Convolutional neural networks

A more complex model was investigated as well, for this purpose a one-dimensional convolutional neural network (1D-CNN) was used. A CNN is a kind of artificial neural network that was originally and most commonly used to handle 2D inputs such as images [14]. This is because a 2D CNN with many layers, and a lot of neurons, can learn to detect complex patterns in images, for example, given a massive amount of training data [14]. More recently, 1D CNNs, CNNs that operate on a one dimensional signal, for example the conductance as a function of time, have been tested for different applications. 1D CNNs have a few advantages compared to 2D CNNs which make them suitable for the application in this project. Shallow 1D CNNs, meaning few layers and neurons, have been shown to be able to complete challenging tasks given 1D signals as input, whereas 2D CNNs generally require deeper networks, more layers and neurons [14]. This is relevant since a CNN with only one layer is easier to understand and may require less training data [14]. It is however worth noting that a shallow 1D CNN can still be a lot more complex than a linear model. 1D CNNs also have a relatively low computational cost and are therefore well suited for real-time applications [14]. Since the goal was to find a combination of model and algorithm capable of working in real time, low computational cost is a requirement.

Structure of a 1D CNN

A one-dimensional convolutional neural network (1D CNN) is commonly used for sequential data, such as time series or signals. The network typically consists of one or more convolutional layers, often followed by pooling layers and fully connected layers, as illustrated in Figure 2.1. The purpose of the convolutional layers is to extract local patterns from the input signal, while the fully connected layers use the extracted features to produce the final output.
In a 1D CNN, a convolutional filter, also referred to as a kernel, is a vector of trainable weights. The length of this vector is called the kernel size and is smaller than the input signal. The filter is moved along the input signal, and at each position a weighted sum is calculated. For an input signal X, a kernel k of size q, the convolution output can be written as

$$y_{i,j} = f\left(\sum_{m=0}^{q-1} X_{i+m} k_{m,j} + b_j\right), \tag{2.4}$$

where $y_{i,j}$ is the output value at position i for filter j, $k_{m,j}$ is the kernel weight at position m for filter j, b_j is the corresponding bias term, and $f(\cdot)$ is the nonlinear activation function. In this work, the ReLU activation function was used. Each filter produces one feature map, meaning that several filters can be used to extract different types of patterns from the same input signal. For layers with multiple input feature maps, each filter contains one kernel for each input feature map, and the resulting convolutions are summed to produce one output feature map per filter.

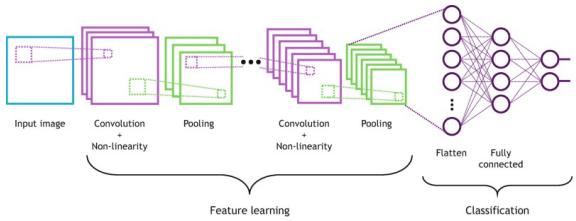


Figure 2.1: Schematic of the general structure of a CNN. The structure shown consists of multiple sets of convolution and pooling layers, with the convolution layers consisting of multiple kernels. The convolution and pooling layers could be repeated to create a deeper network. Image from [15], licensed under CC BY 4.0.

After the convolutional layer, a pooling layer can be applied to reduce the size of the feature maps. In max pooling, the feature map is divided into smaller regions according to a chosen pooling size, and only the largest value in each region is kept. For example, with a pooling size of 3, each group of three values is replaced by its maximum value. This reduces the dimensionality of the data while preserving the most prominent features.
If several convolutional layers are used, the output from one convolutional or pooling layer is used as the input to the next convolutional layer. In this way, deeper layers can learn more abstract features based on the patterns extracted by earlier layers. Finally, the resulting feature maps are flattened and passed to one or more fully connected layers, which combine the extracted features and produce the final prediction or classification.

3
Methods

3.1 Datasets

In this section the different datasets used in this project are discussed. Here, a dataset refers to the data used as input to the model. This could include both parameters decided by the operator that are sent to the robot, such as conductance setpoint, as well as parameters measured by the robot such as the actual measured conductance. There are three different datasets that were used, a slow and a fast dataset as well as image data. The main difference is what the datasets contain and at what frequency the data is saved. The datasets are summarized in Table 3.1. The first dataset will be referred to as the slow dataset, since it is saved at a significantly lower frequency. The slow dataset contains all the parameters that were measured for each run. The second dataset is referred to as the fast dataset, and is saved at higher frequency. In this dataset only one or two parameters are saved, conductance or potential and current. Lastly, images of the weld pool for each run were analysed to form the third dataset, these images were analysed to extract 4 features from each image, as described in further detail in section 3.1.3.

Table 3.1: Overview of the datasets used in the project.

Dataset	Number of parameters	Frequency	Total time	Total drops
Slow dataset	40	$\sim$ 8Hz	29 h	200 000
Fast dataset	1 (2)	4 kHz	29 h	200 000
Image data	4	40 Hz	5 h	40 000

The slow and fast dataset contain data from the same files. A file refers to all the data from one process. Note that if an object is constructed and the process is paused half way through, this would result in two different files for different halves of the same object. These files were selected so that the parameters in the slow dataset contained the same parameters and the fast dataset had the same median frequency. Procada does not only build different prototypes, but also perform experiments with the machine. The files were sorted and those that did not correspond to experiments were removed. This resulted in a set of 38 files. The image-based data was computationally expensive to process. Therefore, not all 38 files were analysed. Instead data were extracted from the 11 most recent files.

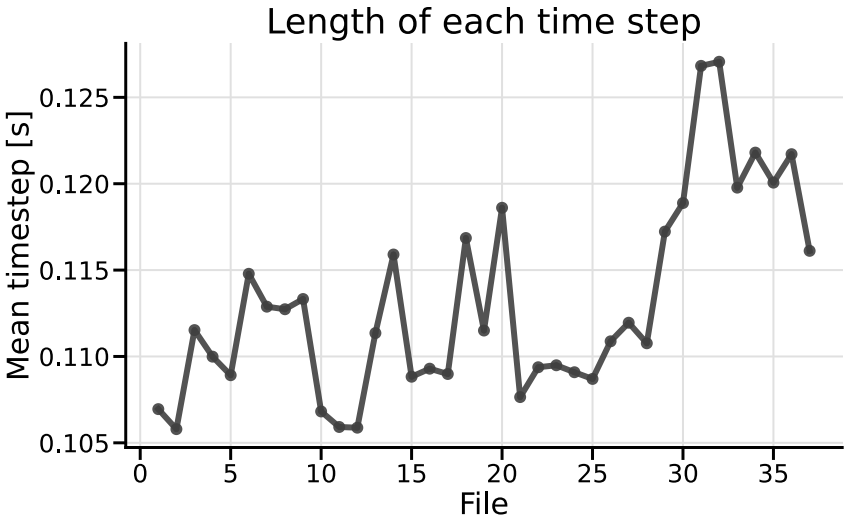


Figure 3.1: Figure showing the mean time between data points for each file in the slow dataset. For each file, the shortest and longest 1% of the intervals were removed to eliminate outliers. The figure shows that the mean sampling interval ranges from 0.10 to 0.13 s.

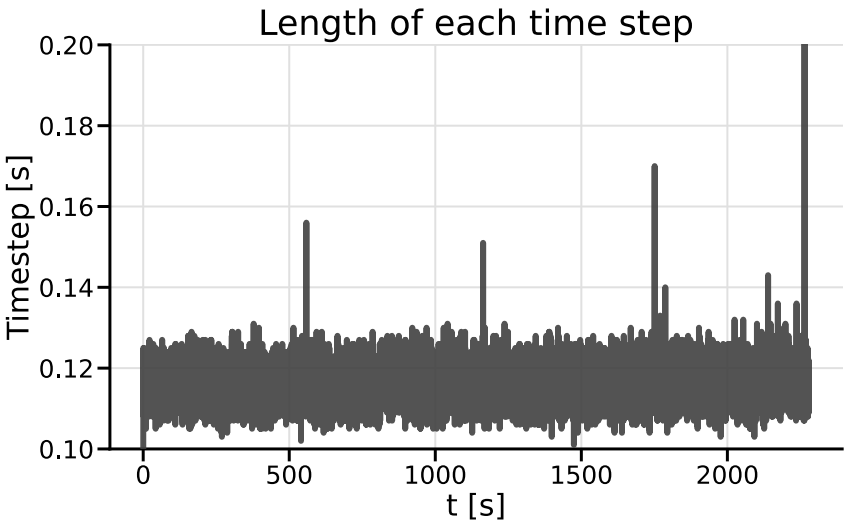


Figure 3.2: Figure showing the time between data points for a typical file in the slow dataset. The figure shows that the sampling rate varies within each file, ranging from 0.1 to 0.13 s with some longer outliers.

3.1.1 Slow dataset

The slow dataset was used to find a model that could determine if the combination of parameters together with the current setpoint results in a process with a high probability of drop or not. The slow dataset has a lower frequency than the other datasets. Another drawback of the slow dataset is that it has an inconsistent frequency. Figure 3.1 shows the mean frequency for each file in the slow dataset and Figure 3.2 shows time between samples for a typical file in the dataset. From these figures it can be seen that the mean sampling frequency varies between files, and within files as well. This means that detecting patterns in the data is harder, since the frequency is inconstant which will partly or fully distort the pattern. Therefore this dataset was used in combination with a linear model to find correlation between parameters and drop probability, and not for recognising signal patterns corresponding to drop using a CNN.

3.1.2 Fast dataset

The fast dataset was used to find patterns related to drops using a CNN. In contrast to the slow dataset, the fast dataset has a high and relatively constant frequency. It is worth noting that the files were chosen so that the fast dataset had the same median frequency. But the constant frequency also depends on how the different datasets are saved. For the slow dataset, each parameter is saved once per iteration of the loop in the machine code, the code that controls the robot. Whereas the fast dataset is written constantly on a ring buffer and then once per iteration all new values on the ring buffer are saved. This means that if the computer runs faster or slower or some part of the loop takes longer than usual, the frequency of the slow data is affected accordingly whereas the frequency of the fast dataset remains constant. The frequency is not perfectly constant, the time between samples varies between three values in a repeated pattern, one iteration of the pattern is shown in Figure 3.3. The pattern was consistent within and between files, with some exceptions. The origin of the pattern or the exceptions was not investigated. Due to the constant frequency and the dataset only containing one parameter, the fast dataset was used to recognise patterns related to drops instead of correlation between parameters and probability of drop.

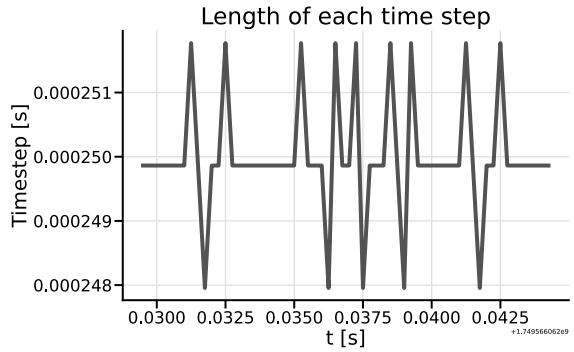


Figure 3.3: Figure showing the pattern of time between data points in the fast dataset. The times range from 0.000248 to 0.000252, roughly corresponding to 4 kHz. This pattern was repeated throughout each file in the fast dataset with some small variations.

3.1.3 Image data

The runs are filmed using a high-resolution camera, focusing mainly on the melt pool. The camera is positioned so that the melt pool appears in approximately the same location in each frame. Since the melt pool is hot, it appears bright in the image. The image is converted to grayscale, where the melt pool appears as a bright region. Contours are then identified using a threshold value between black and white to extract geometrical properties of the melt pool [6]. This threshold can be calibrated to make the detected melt pool larger or smaller. The largest contour area is then selected and assumed to correspond to the melt pool. From this contour, a number of parameters were extracted. These parameters were extracted using a tool developed by Procada. The analysed parameters were the area of the melt pool, as well as its length, width and convexity. The length was defined as the longest straight line that could be drawn within the melt pool, while the width was defined as the longest line that could be drawn in the direction orthogonal to the length. The convexity was chosen to describe the shape and is defined as the area of the melt pool divided by the area of the smallest convex shape enclosing the melt pool. Figure 3.4 shows an example image from the dataset.

Extracting these parameters from the videos was computationally expensive, and therefore only done for a small number of videos in order to evaluate the method. Further, the robot did not have access to the video data in real time, meaning that at the time of the project there was no way of actually implementing a model based on this data in real time. However Procada felt optimistic about the possibility to implement such a model in the future if the model showed promise. This dataset was only used as input for linear models.

3.2 Targets

In order to create an output for the different models which they should aim to predict, some parameter or event had to be chosen as a target to base these outputs

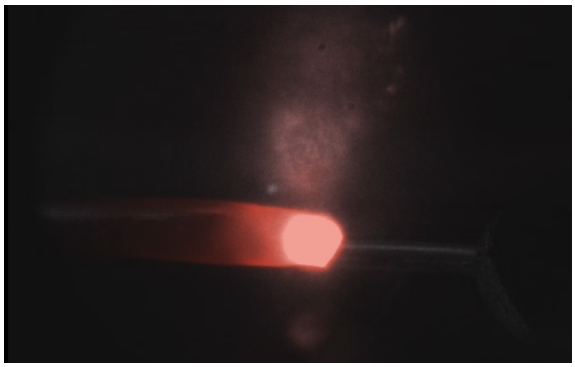


Figure 3.4: Figure showing an example from the image dataset. The brightest part of the image is the weld pool. The red part to the left of the melt pool is the formed bead.

on. Three main events were considered as possible bases for the output: changes in setpoint, drops, and stubbing. If the setpoint was lowered, the setpoint in a window before the change was considered too high, and if it was raised it would be considered too low. However this approach was too inconsistent. First, the setpoint was changed by a human operator, based on observed process behaviour and operator judgement, but not with a fixed amount. Further the setpoint was changed infrequently, meaning that the number of events per file was low. Instead a more concrete and frequent event was sought. The two main events that were considered were drops and stubbing. Drops can be detected by looking at the conductance from the fast dataset. If the value drops below a certain threshold or even reaches zero it is most likely due to a drop [8]. The other considered event was stubbing. In some cases the forces on the wire are measured. By looking at these forces it is possible to detect when the wire is no longer melted in the melt pool and instead scraping along the substrate, stubbing. Out of these two, drops were chosen as a target for multiple reasons. First, the forces are not always measured, meaning that only some data contain information regarding whether the process is stubbing or not. Secondly, even if it were possible to save the force data in the fast dataset in future runs, in the past it has only been saved in the slow dataset, meaning that some occasions of stubbing might have been missed if they occurred between two samples. Finally, occasional drops are more acceptable, and have less negative effects on the properties of the final product [8]. This means that building a combination of model and algorithm that works close to drop is safer than one that works close to stubbing. A measured conductance below 0.5 S was considered a drop. Since in the analysed data drops typically lasted approximately 10 ms, which is long compared with the sampling interval of the fast dataset, almost all drops were expected to be detected.

3.3 Models

This section describes the different models used in the project. Three main model types were evaluated. First, linear models were applied to the slow dataset in order

to investigate correlations between process parameters and drop probability. Second, a linear classifier was applied to image-based data to identify visual patterns related to drop formation. Finally, a 1D CNN was applied to the conductance signal in order to identify patterns associated with drops.
The purpose of the linear classifier and the 1D CNN was to identify process states that were close to drop, but where full drop formation had not yet occurred. These states, referred to as almost-drops, could then be used as a basis for the control algorithm.

3.3.1 Linear model

Most of the project was focused on different variants of linear models. In this case linear models refer to models where each value of the input is multiplied by its own weight. The output of the model is simply the sum of all values in the input multiplied with its weights. The weights were fitted by minimizing the mean squared error (MSE) on the training data. The output y given inputs X was calculated as:

$$y = X\theta + \theta_0, \tag{3.1}$$

where θ is the vector of fitted weights and θ_0 is an added constant.

Inputs

The inputs for the linear models were based on the slow dataset. This dataset was normalised so that the training set had 0 mean and unit variance. The test data was normalized using the same scaler, meaning that the mean of the training set was subtracted and it was divided by the standard deviation of the training set. It is worth noting that the time was included in the slow dataset. However, the time variable was not normalized. Instead, it was set to zero at the start of each dataset.

Single time step models

Single time-step models refer to models that use only one sample of each parameter for each output. If the dataset contains n parameters, each input is a vector of length n, containing the values of all parameters at a given time step. The number of weights in the model is therefore n. This approach was applied to the full slow dataset and to a reduced dataset consisting of principal components.

Multiple time step models

Multiple time-step models use a window of several samples for each parameter to produce each output. Each input can therefore be represented as a matrix of size $n \times w$, where n is the number of parameters and w is the window length. Similar to the single time-step models, this was tested both for the full dataset and for a reduced dataset consisting of principal components.
Since the slow dataset varies slowly over time, using every sample in a window may not provide much additional information. Therefore, an additional input representation was considered, where summary metrics were calculated for each window

instead of using the raw samples directly. Several metrics were evaluated, but only the mean value and standard deviation were used in the final models.

Outputs

Two different continuous output definitions were used and evaluated. The first output was based on a ramp function describing the proximity to a drop. Regions far from a drop were assigned a value of zero, while regions close to or within a drop sequence were assigned a value close to one.
If the time between two consecutive drops was shorter than dt_{drop}, the output was kept at one between the drops. This was done to treat closely spaced drops as part of the same drop sequence. If the time between drops was longer than dt_{drop}, the output decreased to zero, or to the value corresponding to the ramp leading up to the next drop. Between the non-drop region and the next drop, a linear ramp was used.
The output y_n for sample n was calculated from the time of the previous drop, t_{last}, the time of the next drop, t_{next}, and the time of the current sample, t_n, according to

$$y_{\text{ramp},n} = \begin{cases} 1, & \text{if } t_{\text{next}} - t_{\text{last}} < dt_{\text{drop}}, \\ \max\left(0, 1 - \dfrac{t_{\text{next}} - t_n}{t_{\text{ramp}}}\right), & \text{otherwise.} \end{cases} \tag{3.2}$$

For this output different values of t_{ramp} and dt_{drop} were explored, the results in section 4.2 are shown for $t_{\text{ramp}} = 20$ s, and $dt_{\text{drop}} = 1$ s.
The second output was based on the local density, or frequency, of drops. This was computed by placing a Gaussian distribution at each detected drop and summing the contribution from all drops at every sample. This gives an output that increases in regions where drops occur frequently, while isolated drops have a smaller and more localised influence.
For a set of detected drop times $\{t_i\}_{i=1}^{N_{\text{drop}}}$, the density-based output was defined as

$$y_{\text{density},n} = \sum_{i=1}^{N_{\text{drop}}} \exp\left(-\frac{(t_n - t_i)^2}{2w_{\textbf{Gauss}}^2}\right), \tag{3.3}$$

where t_n is the time of sample n, t_i is the time of drop i, N_{drop} is the total number of detected drops, and $w_{\textbf{Gauss}}$ determines the width of each Gaussian contribution. For consistency, the density-based output was also log-transformed and normalized to values between zero and one. This was done according to

$$y_{\text{log},n} = \frac{\log\left(1 + y_{\text{density},n}\right)}{\max_n\left[\log\left(1 + y_{\text{density},n}\right)\right]}. \tag{3.4}$$

3.3.2 Linear classifier

The linear classifier was similar to the linear models described previously, but instead of predicting a continuous output, it predicted the probability of two classes, drop or no drop. This was done using logistic regression, where a linear combination of

the input variables was transformed into a probability using the sigmoid function. The probability of a drop, $p(y = 1 \mid X)$, given input X, was calculated as

$$p(y = 1 \mid X) = \frac{1}{1 + \exp\left(-(X\theta + \theta_0)\right)}, \tag{3.5}$$

where θ is the vector of fitted weights and θ_0 is an added constant.

Inputs

The inputs to the linear classifier were based on image data. Only a window-based approach was considered for this model. For each image in the window, the melt pool was analysed and four metrics were calculated: area, length, width, and convexity. The input to the model consisted of the mean value and standard deviation of each metric within the window, resulting in a total of eight input parameters for each output. Different windows were considered. The results in section 4.3 are presented for a model using a window size of 5.

Outputs

The output was binary. If at least one drop occurred within the window considered for the input, the corresponding output was set to one. Otherwise, the output was set to zero.

3.3.3 CNN model

The conductance signal is one of the signals used by Procada to assess the state of the process. Therefore, a 1D CNN was evaluated as a model for identifying patterns in the conductance signal related to drop. Similar to the linear classifier, the aim was to identify process states close to drop formation, including states where full drop formation had not yet occurred.
Initially, the frequency spectrum of the conductance signal was investigated using a 1D CNN architecture inspired by [16]. The objective was to evaluate whether frequency-domain information could be used to distinguish between stable process conditions and states close to drop formation. A similar 1D CNN was also trained directly on the raw conductance signal in the time domain. Since the time-domain model performed better, and due to time constraints, this approach was prioritised for further evaluation.

Input

The input to the CNN consisted of windows of length w from the raw conductance signal. Two types of windows were used for training: windows far from drops and windows shortly before drops.
Windows far from drops were selected such that the distance to the closest detected drop was at least d_{far} samples in both directions. Since the fast dataset was sampled at a constant frequency, these distances were measured in samples rather than time.

Windows shortly before drop formation were selected before detected drops. The distance between the end of the window and the following drop was defined as d_{short} samples. Only windows not containing any drops and where the distance from the previous drop to the start of the window was larger than d_{far} were used. These windows were expected to contain patterns related to the process approaching a drop state. No overlapping windows were used.

Only windows before drops were used for the drop class. Windows after drops were excluded, since these may contain effects caused by corrective actions from the control system or disturbances resulting from the drop itself. Such effects would not necessarily be present in almost-drop states, where full drop formation does not occur.

Output classes

The CNN was trained as a binary classifier. The outputs for windows selected far from drop were set to zero, while windows selected shortly before drop formation were set to one.

Architecture and hyperparameters

Four main parameters were evaluated for the CNN: the window length w, the size of the first convolutional kernel s_{fk}, the distance defining far-from-drop windows d_{far}, and the distance defining shortly-before-drop windows d_{short}.

The parameters w and s_{fk} affect the architecture of the CNN. The window length w determines the input size and thereby also affects the size of the flattened representation and the fully connected layers. The size of the first kernel, s_{fk}, determines the size of the patterns that can be captured by the first convolutional layer.

In contrast, d_{far} and d_{short} do not change the architecture of the CNN. Instead, they determine how the training windows are selected and labelled. The parameter d_{far} controls how far a stable window must be from the nearest detected drop, while d_{short} controls how close a close-to-drop window is allowed to be to the following drop.

The architecture was first evaluated using a grid search over w and s_{fk}. Based on this evaluation, a smaller number of promising architectures were selected. For these architectures, the influence of d_{far} and d_{short} was evaluated separately.

A 1D CNN with three convolutional layers and one fully connected layer based on the work of [16] was used. This CNN was compared to a similar CNN constructed with only one convolutional layer and one fully connected layer.

For the model, 32, 16 and 8 filters were used for the first, second and third layer respectively for the 3 layer CNN. For the one layer CNN 32 filters were used. For the 3 layer CNN a pooling size of 4 was used for the first layer and 2 for the second layer.

ReLU activation, the Adam optimizer, cross-entropy loss, a learning rate of 0.001, and a batch size of 32 were used for both CNNs.

3.3.4 Model evaluation

In order to compare models with each other or compare different hyperparameters for the same model, each model with its hyper parameters had to be evaluated. For the linear classifier and the CNNs this was done by looking at the number of correctly classified inputs, known as the accuracy. However since the training and validation, data contains more zeros, no drop, than ones, drops, by a factor of ten, a model that predicted every input as a zero would get an accuracy of 91 %. This means that accuracy might be a misleading metric [17]. Instead, balanced accuracy was evaluated. It was calculated as the mean of the accuracies for the positive and negative samples and provides a more informative measure for imbalanced classification problems [18]. Furthermore, the fraction of detected drops, referred to as sensitivity, the fraction of predicted drops that were actual drops, referred to as precision, and the false positive rate were evaluated [19].

For the linear models the output was continuous and a metric like accuracy did not make sense, since the model almost never estimated the exact value. Instead three different metrics were used. The three metrics were mean squared error, MSE, the R^2 metric, and a metric called ordered accuracy. MSE and R^2 were based on the errors in the predictions. Given the true output of the test set, y_i, the predictions, f_i, and the mean true output, $\bar{y}$, the metrics were calculated as follows:

$$\text{MSE} = \frac{1}{n}\sum_i (y_i - f_i)^2 = \frac{1}{n}\sum_i e_i^2, \tag{3.6}$$

$$R^2 = 1 - \frac{\Sigma_1}{\Sigma_2}, \qquad \Sigma_1 = \sum_i e_i^2, \qquad \Sigma_2 = \sum_i (y_i - \bar{y})^2. \tag{3.7}$$

MSE is a measure of the total error between prediction and true output, whereas R2 is a measure of how well the model is able to explain the variance in the true output. A positive R2 score means that the model can explain the output better than the mean value of the test set, a negative value means that the mean value is a better predictor than the model [20].

Finally, the ordered accuracy was used. This metric was used to be able to compare the linear models to a CNN. As explained earlier counting the fraction of correctly predicted outputs would be unfair. Instead this was done pairwise. This means that a large number of pairs of samples were evaluated. If the sample with a larger output also had a larger predicted output this was considered a correctly predicted pair. The ordered accuracy was defined as the fraction of correctly predicted pairs. Pairs with the same true output were not considered when this metric was calculated.

3.4 Control algorithms

The algorithms were created and tested in two steps. First the algorithm was evaluated on a simulation of the process. If the algorithm behaved decently in the simulation it was then tested on the actual process. The entire control system or

the machine is outside the scope of this project, however the relevant detail is that it is based on a loop. The machine performs different actions for each iteration, one such action is determining the setpoint using the algorithm. This means that the setpoint is determined once per iteration of the loop, either by updating the setpoint or keeping the last used setpoint based on some criteria.

3.4.1 Simulations

The simulation was created in a simple way, the simulated process had a slowly changing true limit, and a setpoint decided by the algorithm. The probability of drops was calculated as one minus the cumulative distribution function (CDF) of a normal distribution with mean equal to the true limit and standard deviation σ evaluated at the setpoint. Further the simulation was set up so that the drop duration was normally distribute around 10 ms with standard deviation of 0.5 milliseconds. The intervals between the starts of consecutive drops were normally distributed around 70 ms, standard deviation 15 ms. This was done to mimic the behaviour of drops from the actual process and based on data from previous processes. A time step of 1 ms was used, with $\sigma = 0.5\,\mathrm{S}$. The true limit was updated at each time step by adding a change ΔG drawn from a normal distribution, $\Delta G \sim \mathcal{N}(0, 0.0001)$.

3.4.2 Algorithm design

One algorithm was considered, based on a context-aware filter. The algorithm depended on the assumption that the probability for drops was normally distributed around an unknown threshold, G_{lim}, with variance σ^2. The algorithm updated the predicted mean value of G_{lim}, μ, as well as the uncertainty of that prediction, P. The algorithm was a one-dimensional version of the context-aware filter described in the literature [21]. The values were updated as follows:

$$\mu_{k+1} = \mu_k + \frac{P_k}{\sqrt{P_k + \sigma^2}}\alpha(z_k), \tag{3.8}$$

$$P_{k+1} = \max\left(Q,\, P_k\left(1 - \frac{P_k\alpha(z_k)\Big(z_k + \alpha(z_k)\Big)}{P_k + \sigma^2}\right)\right) + Q \tag{3.9}$$

$$z_k = \frac{\mu_k - G_{0,k}}{\sqrt{P_k + \sigma^2}}, \tag{3.10}$$

$$\alpha(z_k) = \begin{cases} \dfrac{\phi(z_k)}{\Phi(z_k)}, & \text{for a positive observation,} \\ -\dfrac{\phi(z_k)}{1 - \Phi(z_k)}, & \text{otherwise.} \end{cases} \tag{3.11}$$

Here, z_k denotes the normalized distance between the predicted limit μ_k and the setpoint $G_{0,k}$. The normalization is performed with respect to the total standard deviation, $\sqrt{P_k + \sigma^2}$, where P_k represents the variance of the predicted limit and σ^2

represents the variance associated with the drop probability model. Furthermore, $\Phi(x)$ and $\phi(x)$ denote the cumulative distribution function and probability density function of the standard normal distribution, respectively.

The function $\alpha(z_k)$ corresponds to an inverse Mills ratio and determines the correction applied to the predicted mean after observing a binary event [21]. Its form depends on whether a drop is observed or not. Intuitively, $\alpha(z_k)$ weights the update according to both the likelihood of the observed event and the distance of the prediction from the threshold. Thus, unlikely observations lead to larger corrections, while observations that are already consistent with the prediction result in smaller updates. Observations made close to the predicted limit are also more informative than observations far from the limit, where the outcome is less sensitive to small changes in the estimated limit. The derivation of this update rule is described in further detail in the literature [21].
Note that the estimated threshold, μ, and the setpoint used in the algorithm, G_0, could be the same value, but does not have to for the algorithm to work. Since setting G_0 to the estimated value of the threshold would result in almost constant drops, one would like to use a setpoint slightly above the threshold. This could be accomplished in different ways that are discussed in further detail in section 4.5.1, however, one way that was used in this project was to introduce a buffer between μ and G_0. This buffer was based on the uncertainty of the prediction, P, meaning that in each iteration we get $G_0 = \mu + c\sqrt{P}$.
Finally whether an observation was determined positive or not was based on the occurrence of drops. Two versions of the algorithm were used and since they were slightly different in nature the two versions determined observations as positive in slightly different ways. The first version is the intuitive implementation where the algorithm is evaluated once every iteration and if a drop has occurred since the previous iteration it is classified as a positive observation, otherwise it is not. The other version of the algorithm was based on time intervals. At the end of every interval the observation was determined to be negative and every time a drop occurred it was positive. The two versions are described in further detail below.

Intuitive algorithm

As previously mentioned the intuitive algorithm made an observation and updated μ in each iteration. The algorithm operated at approximately 8 Hz, which is fast relative to the dynamics of the system. Furthermore, the algorithm used values from the ring buffer since the previous iteration to check for drops. If a drop had occurred during this interval, the process was considered to be in a dropping state in the current iteration. As mentioned in the simulation section, individual drops typically lasted approximately 10 ms, while the interval between consecutive drops was roughly 70 ms. Consequently, during the first 10 ms after the start of a drop, the process is likely to remain in the dropping state regardless of the setpoint. Similarly, between approximately 10 and 70 ms after a drop, the occurrence of another drop is unlikely, again largely independently of the setpoint. This dynamic had to be taken into account in the algorithm and was implemented as damping coefficients. The

resulting update of μ was given by

$$\mu_{k+1} = \mu_k + k_{\text{damp}} \frac{P_k}{\sqrt{P_k + \sigma^2}} \alpha(z_k) \tag{3.12}$$

where k_{damp} is a damping coefficient that depends on the elapsed time since the start of the most recent drop. It was calculated according to

$$k_{\text{damp}}(t, t_0) = \frac{1}{(1 + \exp(-k(t - t_0)))} \tag{3.13}$$

where t is the time since the start of the most recent drop. Note that the actual time is unknown. t is the time since the most recent drop was first recorded. The parameter k determines the shape and steepness of the damping function, while t_0 specifies the time at which the damping coefficient equals 0.5. Different parameter values were used depending on whether or not a drop was observed. The values used are summarized in Table 3.2.

Table 3.2: Parameters used for the damping coefficient.

Parameter	Value
$t_{0,\text{drop}}$	10 ms
$t_{0,\text{no drop}}$	500 ms
k_{drop}	$1.1\,\text{ms}^{-1}$
$k_{\text{no drop}}$	$0.0014\,\text{ms}^{-1}$

Using $t_{0,\text{drop}} = 10$ ms is intuitive since that is the length of time that drops lasted on average. The constant $t_{0,\text{nodrop}}$ was set to 500 instead of 70 ms in order to decrease the drop frequency. We expect drops to appear after 70 ms if the setpoint is low enough, however if $t_{0,\text{nodrop}}$ is set to 70 ms, the setpoint will decrease rapidly shortly after a drop and result in a new drop increasing the drop frequency. Based on experience from Procada a drop frequency of 2 Hz was acceptable, therefore the value was set to 500 ms. The k values were selected through testing in the simulation and there is no specific reason for the exact values.

Time interval algorithm

The second algorithm was based on time intervals. This algorithm used every drop as a positive observation and the end of each interval as a negative observation, whether or not a drop had occurred in that interval. Note that each drop was only used once. This means that if a drop was detected two iterations in a row, the second observation was ignored. This algorithm originally used intervals containing at least one drop as positive and other intervals as negative, however it was altered as described in order to faster respond to drops. The time interval used for this algorithm was 500 ms.

3.4.3 Experiment

An experiment was performed to test both versions of the algorithm. During this project only one experiment was performed. The experiment was performed by constructing a single bead thick ring. The first two layers of the ring were built with the setpoint being controlled by the operator. Then the time interval algorithm controlled the setpoint for two layers, after which the intuitive algorithm was activated. The time interval algorithm used a time interval of 0.5 s and had the buffer applied as described in section 3.4.2. The parameters used for the intuitive algorithm are summarized in Table 3.2. The algorithm used a buffer similar to the time interval algorithm.

3.5 Visualisations

Another way to study the correlation between input and output was through visualisations. The primary objective of these visualisations was as the name implies to visualize the correlation in order to better understand the process rather than creating something that could be used as a base for an algorithm.

3.5.1 PCA and PLS plots

When PCA and PLS were used to analyse the data, the data points were divided into three groups: shortly before a drop, shortly after a drop, and far from any drop. The data points from the different groups were then plotted against two principal components in the case of PCA, or two latent variables in the case of PLS. Different thresholds for what counted as shortly before or after drops were investigated. The three groups were used to construct a target matrix for PLS. Data points far from drop, shortly before drop and shortly after drop were represented by the vectors $[1, 0, 0]$, $[0, 1, 0]$ and $[0, 0, 1]$ respectively.
These plots were used to investigate whether the different groups formed distinguishable regions in the reduced-dimensional representation of the data. The slow dataset was used for PCA and PLS since it contains a relatively large number of variables. In contrast, dimensionality reduction was not considered for the fast data set, which contains only one or two variables.

3.5.2 FFT spectrograms

Another way of finding patterns related to drops was through FFT spectrograms. These were analysed in areas shortly before drops as well as far from drops in order to find patterns or behaviours in the frequency spectrum related to drops. These spectrograms were created and analysed mainly for the conductance signal. An FFT was performed for datasets where both the current and potential were available in a high sampling frequency, therefore the frequency spectrum for the potential, current and power signals was analysed as well. For the FFT a window size of 4096 samples was used. Before applying the FFT, a linear trend was removed from each window

to reduce discontinuities between its endpoints. A Hann window was then applied to further reduce spectral leakage [22].

3.5.3 Image analysis

Each process run is accompanied by a video of the melt pool during the run. This video was investigated for one of the runs in order to see if the cause of the drops could be categorised into different groups based on patterns seen in the video.

4
Results and discussion

4.1 Visualisations

In this section the results of the different attempts to visualize data and find patterns related to drops are presented.

4.1.1 PCA and PLS plots

Scatter plots for different combinations of the first five principal components were constructed. The first two components are shown in Figure 4.1 where there is no clear difference between data points close to or far from drops. The data appear to form two separate regions as a result of different parameters being used by the machine. However, both regions are seen close to and far from drops. Similar results were obtained for different thresholds defining proximity to a drop and for other combinations of the first five principal components.

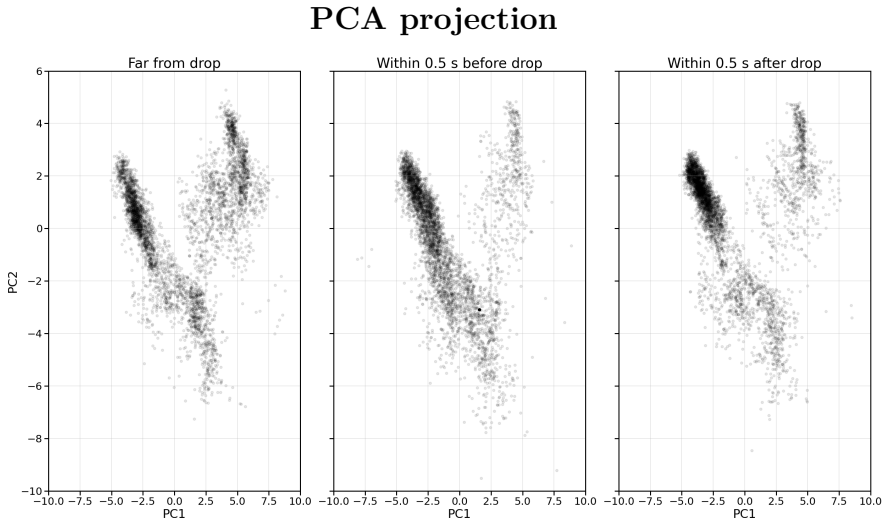


Figure 4.1: All data points projected on the first two principal components. To the left data points far from drops are projected, in the centre data points shortly before drop and to the right data points shortly after drop.

The data points projected onto the first two latent variables constructed using PLS are seen in Figure 4.2. For this analysis all data were used to fit the latent variables and there is a pattern found only before drop. As seen in Figure 4.2 the second latent variable only takes values below -2.5 close before drops. It is however worth noting

that the corresponding direction in the output matrix is $c = [0.048, -0.058, 0.01]$. This indicates that the second latent variable was constructed primarily to distinguish between data points far from drops and those shortly before drops.

In Figure 4.3, the latent variables were constructed using data from 75 % of the files, while data from the remaining 25 % were projected onto these variables. In this case, no clear difference can be observed between the three groups. This indicates that a correlation associated with data points shortly before a drop could be identified when all data were included in the construction of the latent variables. However, this correlation did not necessarily generalize to files that were not used to construct the variables.

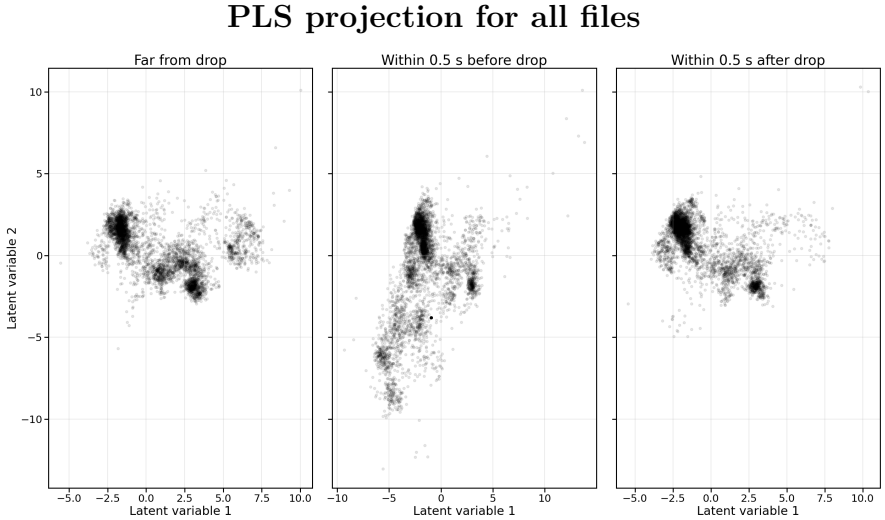


Figure 4.2: All data points projected on the two first latent variables constructed using the same data. To the left data points far from drops are projected, in the centre data points close before drop and to the right data points close after drop.

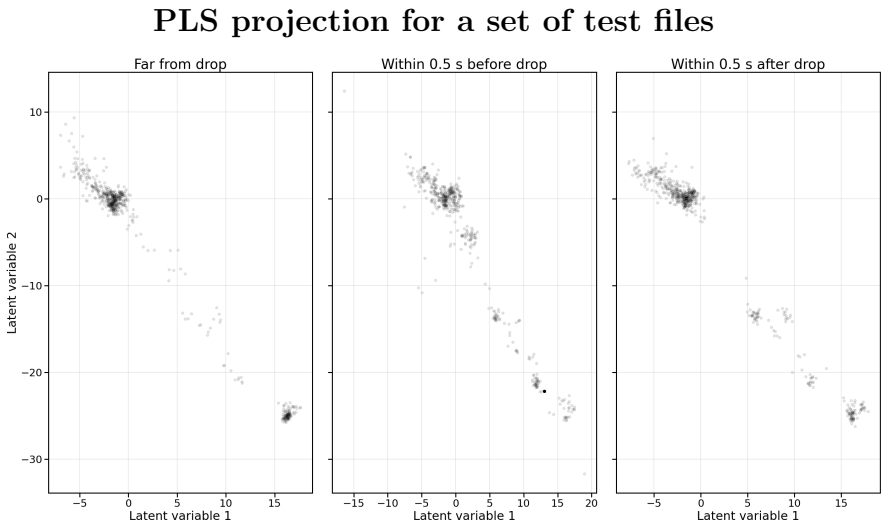


Figure 4.3: Data points from a set of test files projected on the two first latent variables constructed using the data from different files. To the left data points far from drops are projected, in the centre data points close before drop and to the right data points close after drop.

4.1.2 FFT spectrogram

The frequency spectrum of the conductance signal was studied in different regions to investigate if there exists patterns related to drops. Figure 4.4 shows the conductance signal as well as the frequency spectrum between two drops. The only noticeable pattern in the frequency spectrum is the lines of high intensity across all frequencies corresponding to the peaks in the signal. Figure 4.5 shows the frequency spectrum for the same region but only includes the low frequencies. When the low frequencies are studied the lines corresponding to peaks are less evident. However there is a high-intensity band at around 20 Hz. This band varies somewhat in frequency, but with no clear pattern corresponding to drops. Similar results were seen in all investigated regions as well as for the current, voltage, and power signals.

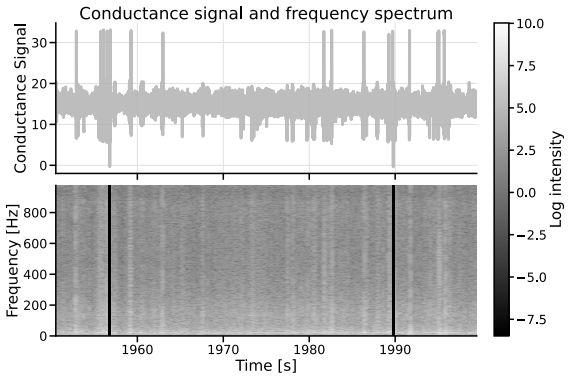


Figure 4.4: Comparison of the conductance signal and a large range of the frequency spectrum in a region between two drops.

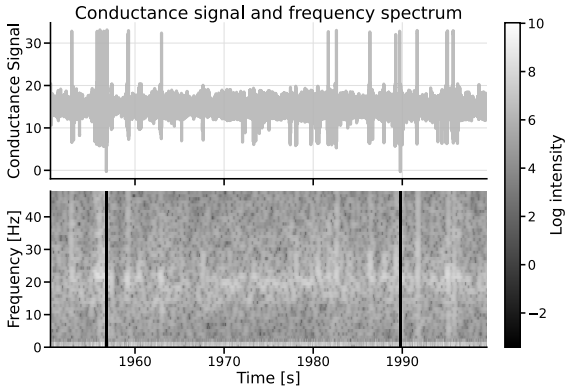


Figure 4.5: Comparison of the conductance signal and the lower part of the frequency spectrum in a region between two drops.

4.1.3 Image analysis

When the video was analysed, four different patterns related to drops were observed. Note that these patterns appeared in regions far from drops as well, and for some drops more than one pattern was observed. The four patterns were scales on the bead, varying bead thickness, the wire moving in the frame and flashes of light.
The scales appeared in regions with many drops, where an irregular surface is expected. The flashes of light appeared to be caused by the laser hitting smoke from the process. The author does not have sufficient process-specific knowledge to determine the underlying causes of the smoke, the wire movement, or the variations in bead thickness.

4.2 Linear model

The linear model was evaluated using one out of the 38 files as a test set and the data from the other files as training data. Then the R^2 score and ordered accuracy were calculated for the test file.

4.2.1 R^2 score

As explained earlier the R^2 score describes how well the model explained the variance in the output. In Figure 4.6 this is given as a percentage, meaning that a score of 100 means that all the variance in the output is explained by the model. A score of 0 would imply that the model is an equally good predictor as the mean value of the test output. As can be seen in Figure 4.6 the linear model has a negative R^2 score for all input types for most test files. We see scores close to zero, or slightly positive, when the average output for the test file is close to the average output of the training data. However if the mean value of the test file is different from that of the training set the R^2 score becomes negative. This implies that there is a difference between the files, at least when they correspond to different runs, that the model is unaware of but that has an effect on the probability of drops. This

Table 4.1: Mean and standard deviation of the order accuracy across all test files for the different inputs and outputs.

Input	Mean [%]	Std. [%]
Single frame, ramp	60.1	15.1
Single frame, density	59.1	9.8
Window mean/std, ramp	70.1	12.6
Window mean/std, density	64.3	8.6
PCA single frame, ramp	61.0	17.6
PCA single frame, density	57.7	10.4
PCA window mean/std, ramp	71.5	14.0
PCA window mean/std, density	63.7	8.5

could have multiple causes, such as a different wire being used, or the part that is being built is single-bead instead of multi-bead.

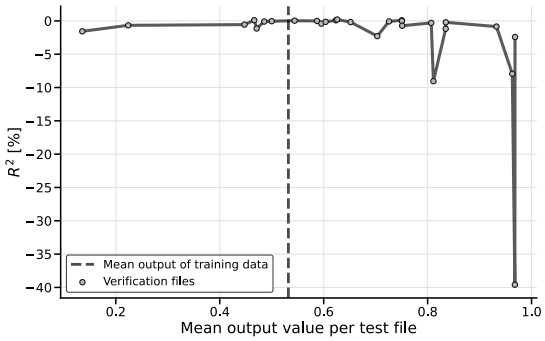
Further, even when the mean value of the test set is the same as the training set the R^2 score is less than one percent, and only positive in a few cases. This implies that the model showed limited predictive performance. It cannot perform better than the mean value, and can only perform similar to the mean value for a certain mean value, the same as in the training set.

4.2.2 Ordered accuracy

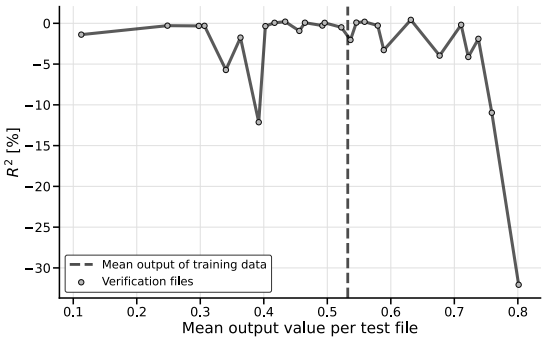
In Figure 4.7 the ordered accuracy for the different kinds of input can be seen for the linear model. Based on how the ordered accuracy is defined, a model that just assigns a random value to each sample would get a score of 50 %. As can be seen in Figure 4.7 the ordered accuracy does not seem to depend on the mean value of the test set. The mean score for each combination of input and output is presented in Table 4.1. When mean and standard deviation were used as an input, either for all parameters or a few PCA components, the model seemed to perform slightly better. However these inputs resulted in an ordered accuracy over 50% for most, but not all, test files, and the value was not clearly above 50% for most test files. The model performed slightly worse for single-time-step inputs. Using all parameters performed better than using a few PCA components. Looking at the summary for all files in Table 4.1, the mean score ranged from 57.7 to 71.5%. This means that the model performed better than random guessing for all combinations. However, even the highest mean score of 71.5% indicates that the model does not provide sufficiently reliable predictions.

4.2.3 Summary

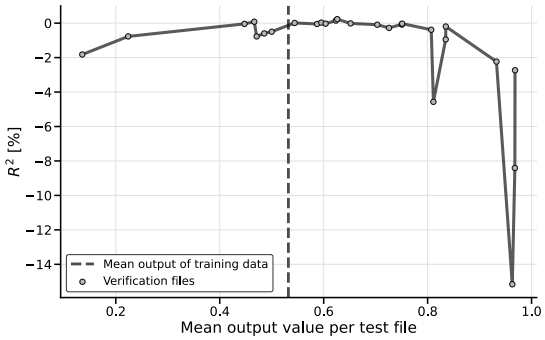
Both the poor R^2 score and ordered accuracy imply that this sort of model is not suitable as the basis for an algorithm. The R^2 score showed that the model is bad



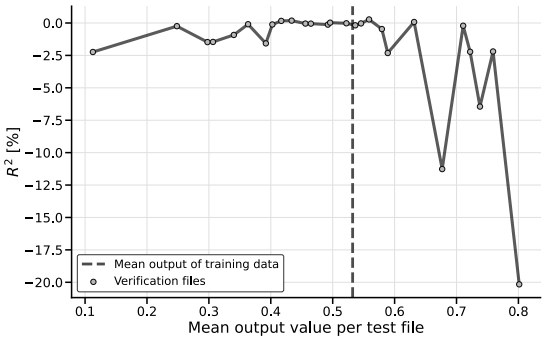
(a) Single data point, ramp



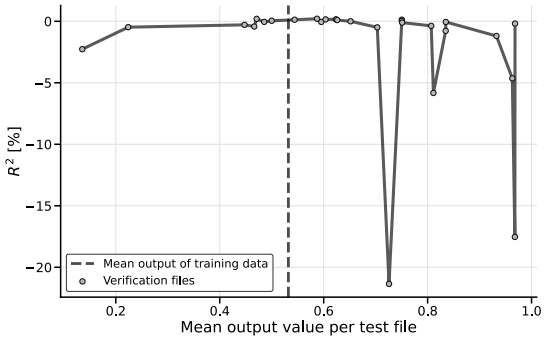
(b) Single data point, density



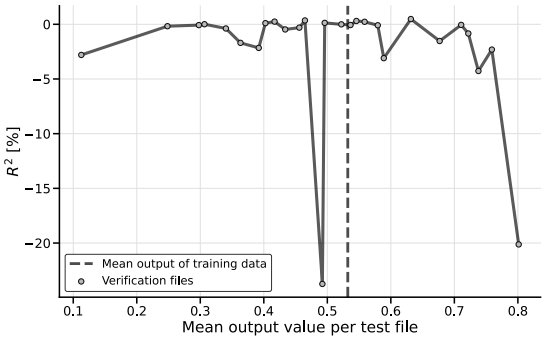
(c) PCA components, ramp



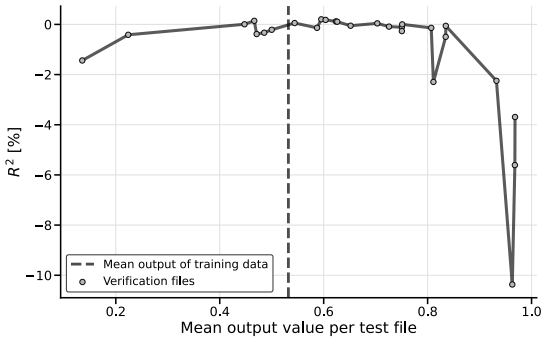
(d) PCA components, density



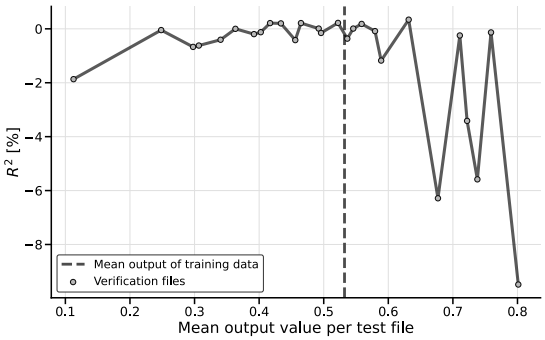
(e) Window mean/std, ramp



(f) Window mean/std, density

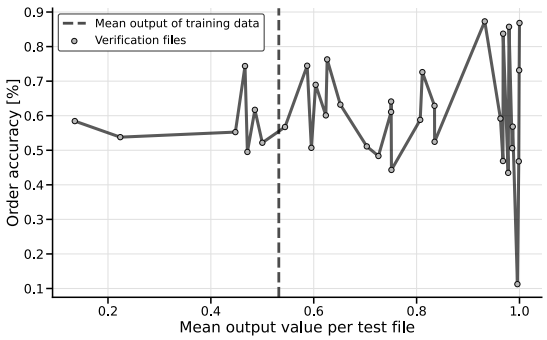


(g) PCA window mean/std, ramp

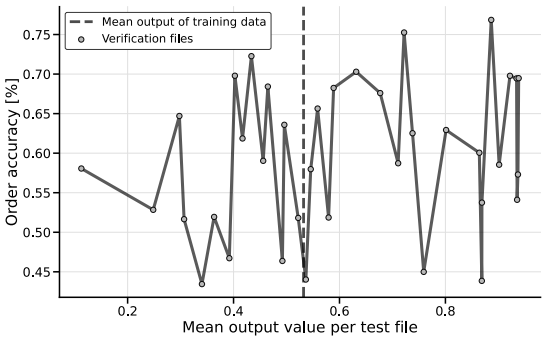


(h) PCA window mean/std, density

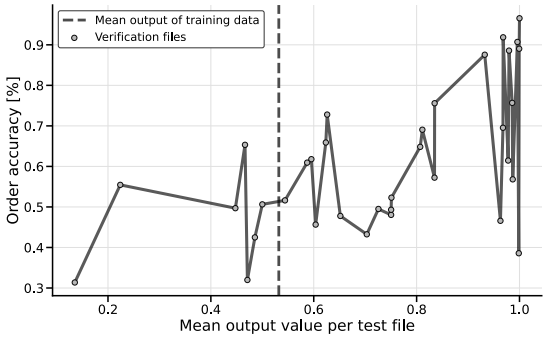
Figure 4.6: R^2 scores for all input methods and both output targets.



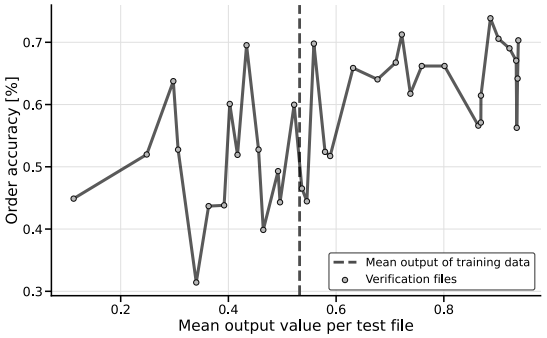
(a) Slow single, ramp



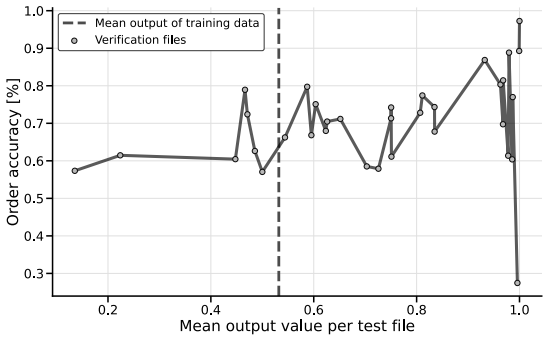
(b) Slow single, density



(c) PCA single, ramp



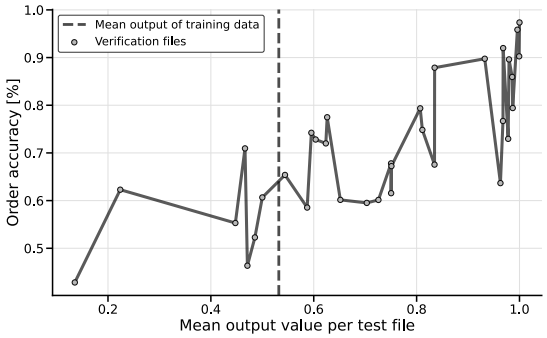
(d) PCA single, density



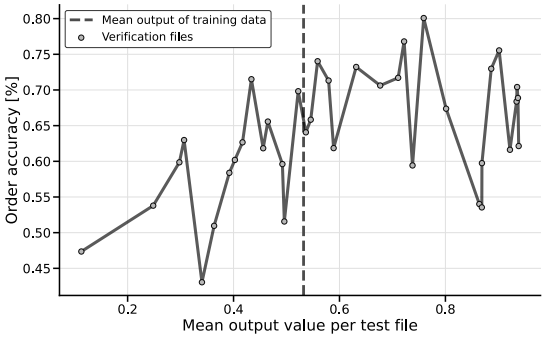
(e) Slow mean/std, ramp



(f) Slow mean/std, density



(g) PCA mean/std, ramp



(h) PCA mean/std, density

Figure 4.7: Ordered accuracy scores for all input methods and both output targets.

at adapting to new runs, and even setups close to the ones in the training data get a prediction similar to the mean value. The low ordered accuracy implies that the model cannot reliably differentiate between two samples within the same run and determine for which condition the process has a higher likelihood of drop.

4.3 Linear classifier

The linear classifier was evaluated similarly to the linear model, with one file being used as test data and the other files for training. The sensitivity, precision, fraction of false positives and balanced accuracy are summaries in table 4.2. The fraction of false positives is defined as the number of false positives compared to the total number of negative samples, describing how many of the negative samples were assumed to be positive.

Table 4.2: Classification results for each file used as test data. For each file the sensitivity, precision, false positive rate, balanced accuracy and class balance are presented.

Test file	Sens. [%]	Prec. [%]	FPR [%]	Bal. acc. [%]	Class bal. [%]
1	26.3	10.1	15.5	55.4	93.8
2	81.5	14.3	16.7	82.4	96.7
3	82.9	6.7	11.8	85.6	99.0
4	96.6	8.2	16.4	90.1	98.5
5	37.6	6.7	13.6	62.0	97.5
6	41.2	19.1	13.8	63.7	92.7
7	77.1	11.3	12.4	82.4	98.0
8	52.0	21.2	16.7	67.7	92.0
9	79.3	63.2	30.2	74.5	60.4
10	73.3	47.8	66.4	53.5	54.7
11	90.9	23.0	55.5	67.7	84.6

The balanced accuracy of the model was above 50 % for all files. The sensitivity varied a lot between the different test files, ranging between 26.3 % and 96.6 %. The difference is relatively large. This may indicate that the drops for some process parameters are easier to find than others, however this was not investigated further. It should be noted that the duration of one drop is much shorter than the time between frames and it is therefore unlikely that the performance comes from actual drops being seen in one of the frames. The sensitivity is lower than the balanced accuracy for most files, implying that the model has a bias towards predicting not drop. This is expected since that is more common in the data, above 90% for most files making it more common in the training data [17]. As mentioned the sensitivity varied a lot between files, the model found a large fraction of the drops for some files but less than half for others, meaning that it does not work for an arbitrary process. It is reasonable to assume that it would be harder to find almost-drops. Since the results did not indicate that the model would be able to find more than half of the instances of almost-drop, this model was not investigated further.

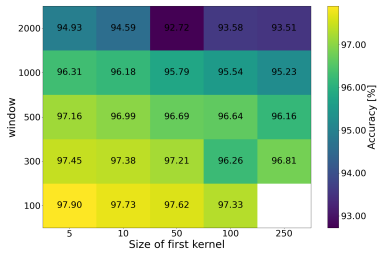
4.4 CNN

Due to time constraints, the development and evaluation of the CNN models were limited. The results presented in this section should therefore be considered preliminary. The performance of the one-layer and three-layer CNN based on the size of the window and first kernel is shown in Figure 4.8. The results are the average over seven runs for each set of parameters. Both models show similar trends in accuracy, the best set of parameters was a window size of 100, and a first kernel size of 5. In principle, a larger first kernel could represent patterns captured by a smaller kernel by assigning low weights to the additional kernel positions [23]. However, larger kernels also introduce more trainable parameters and may therefore require more data or longer training to achieve comparable or better performance. Similarly, a larger window size should not necessarily reduce the accuracy either. However due to the nature of how the training data was collected a larger window meant that fewer windows could be used for training and testing explained in section 3.3.3. For reference, the training set for $w = 2000$ was roughly 20 % smaller than the training set for $w = 100$, which could explain the drop in accuracy.

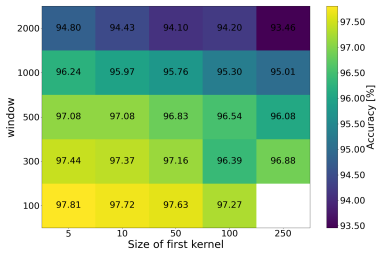
Figure 4.8 also shows that the two models show similar patterns for the sensitivity as well, the best sensitivity was still obtained with a small first kernel, but increased with window size. The highest sensitivity was found for $w = 2000$ and a first kernel size of 5. A larger window length appeared to increase the sensitivity drops, but this was accompanied by lower accuracy. It could potentially be because a larger window means more possible patterns leading to more drops being found but also more false positives, seen in the lower accuracy but higher sensitivity drops.

In both cases a larger first kernel led to worse results. One explanation could be the number of training epochs. A larger kernel means more trainable parameters which should require more epochs. However due to a lack of time the number of epoch was set to 40. In [16], CNNs of similar or greater size converged after 40 epochs, and therefore 40 epochs were used here. However the number of epochs needed could depend on the input as well, and additional runs should be performed to confirm that these CNNs had time to converge. However, this was not been done, due to the lack of time. The good performance obtained with $s_{fk} = 5$ suggests that relevant local patterns may be captured at short time scales, although this does not exclude the presence of longer-term dependencies.

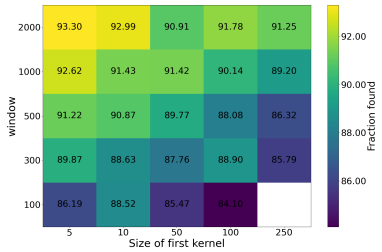
The dependency on d_{far} and d_{short} is shown in Figure 4.9, again the result is the average over seven runs. This analysis was only performed for the one-layer CNN, since it showed similar performance to the three-layer CNN while having lower computational cost. Two window lengths were evaluated: $w = 100$, which gave the highest accuracy, and $w = 2000$, which gave the highest sensitivity. Sensitivity, precision, false positive rate, balanced accuracy and class balance are presented in Table 4.3.
Both accuracy and sensitivity showed similar trends for the two window lengths. The performance improved when d_{short} was reduced, meaning that the positive training



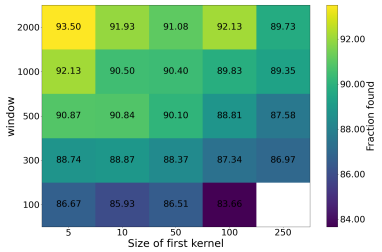
(a) accuracy for single layer CNN



(b) Accuracy for three layer CNN



(c) Sensitivity for single layer CNN



(d) Sensitivity for three layer CNN

Figure 4.8: Accuracy and sensitivity for single- and three-layer CNN as a function of window, w and size of the first kernel. The blank cell in the bottom right of each plot is caused by the fact that the size of the kernel cannot be larger than the window size.

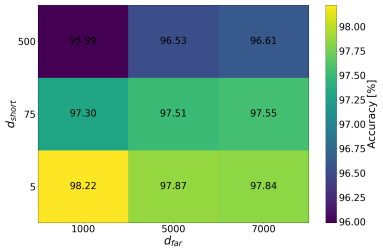
windows were selected closer to the following drop. This suggests that more patterns related to drops appear shortly before the drop occurs.

The performance showed less dependence on d_{far}, at least for the values evaluated. If d_{far} is too small, windows containing drop-related patterns may be included in the negative class. This could make the classification task less clear and reduce model performance. However, no clear decrease in performance was observed for the evaluated values of d_{far}.

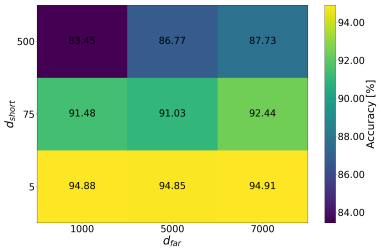
The rapid decrease in performance with increasing d_{short}, combined with the limited dependence on d_{far}, suggests that most of the patterns used by the CNN occur shortly before drop formation. In the evaluated data, the relevant patterns appeared to occur within approximately 1000 samples, and possibly within 75 samples, before the drop. With a sampling frequency of 4000 Hz, this corresponds to approximately 0.25 s and 0.01875 s, respectively.

Table 4.3: Sensitivity, precision, false positive rate, balanced accuracy and class balance for the different CNN configurations. All values are given in percent.

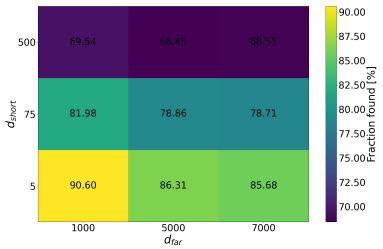
Window	Layers	Sens.	Prec.	FPR	Bal. acc.	Class bal.
100	1	86.2	94.7	0.6	92.8	88.7
2000	1	94.5	93.2	4.8	94.8	59.3
100	3	86.7	93.4	0.8	93.0	88.7
2000	3	93.5	93.7	4.3	94.6	59.3



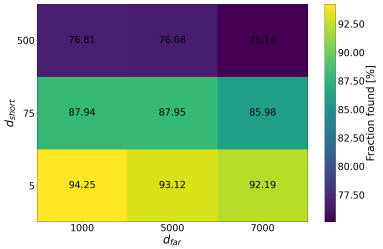
(a) accuracy for window = 100



(b) accuracy for window = 2000



(c) Sensitivity for window = 100



(d) Sensitivity for window = 2000

Figure 4.9: Accuracy and sensitivity for two single-layer CNNs using a window size of 100 and 2000 as a function of d_{short} and d_{far}

4.5 Algorithm performance

This section presents the performance of the intuitive and time-interval algorithms both in a simulation and in a real process.

4.5.1 Simulation

This section shows the performance of both algorithms in a simulation.

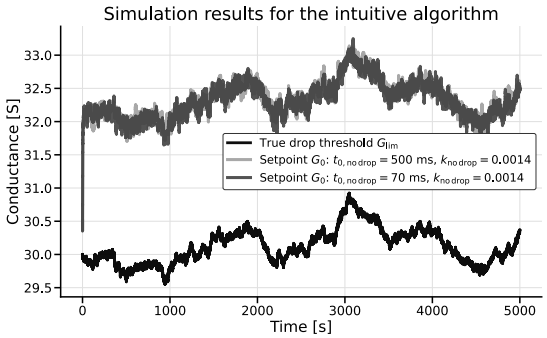
Intuitive algorithm

The results of the simulation can be seen in Figure 4.10. The figure shows the results for different values of the parameters $t_{0,\text{no drop}}$ and $k_{\text{no drop}}$. The results are compared to the values that were used in the final version of the algorithm that was applied to the real process. These values were $t_{0,\text{no drop}} = 500$ ms and $k_{\text{no drop}} = 0.0014$. The ability to follow the threshold value was similar for the different parameters. The largest difference was seen in drop frequency which can be seen in Table 4.4. Increasing $t_{0,\text{no drop}}$ led to a lower drop frequency and decreasing it led to a higher drop frequency. This is expected since $t_{0,\text{no drop}}$ determines how long after a drop the gain reaches 0.5. Moving that point closer to 0 ms means that the setpoint will decrease faster after a drop, increasing the frequency. Note that each simulation starts with many rapid drops since the setpoint is too close to the threshold. These drops were not included in the drop frequency. Instead the drop frequency was calculated starting from 10 s.

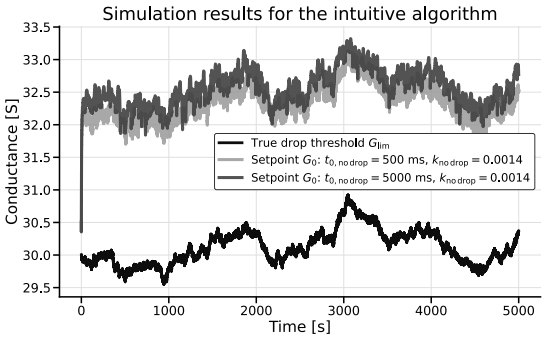
Table 4.4: Simulated drop frequency for the intuitive algorithm.

$t_{0,\text{no drop}}$ [ms]	$k_{\text{no drop}}$	drop frequency [drops / s]
500	0.0014	0.807
70	0.0014	0.917
5000	0.0014	0.287
500	0.00001	0.661
500	0.1	0.822

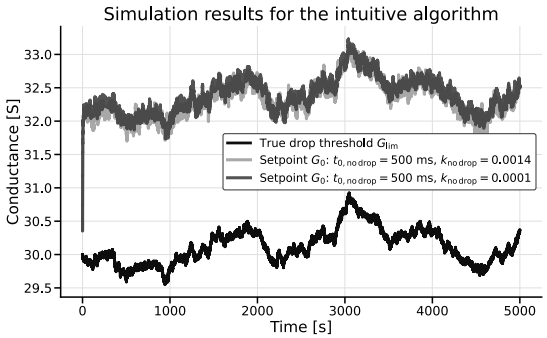
When $t_{0,\text{no drop}}$ was chosen, two things were considered, the drop frequency and how fast the algorithm is able to respond to an unexpected decrease in threshold. If a drop occurs immediately before a decrease in the threshold, $t_{0,\text{no drop}}$ will determine roughly how long it will take for the algorithm to react to the decrease. This is why the value 500 was chosen over 5000 even though it resulted in a higher drop frequency. Since the drop frequency for $t_{0,\text{no drop}} = 500$ and 70 ms was similar a lower value could have been used, but was not.
The k value determined the shape of the gain function, where a lower value means a smoother curve and a higher value results in a sharper curve. The algorithm used a k value of 0.0014 and here the effect of the k value is shown. The selected k value was compared to a larger and smaller value, 0.1 and 0.0001. The gain as a function of time since a drop for different k values can be seen in Figure 4.11. The effect of



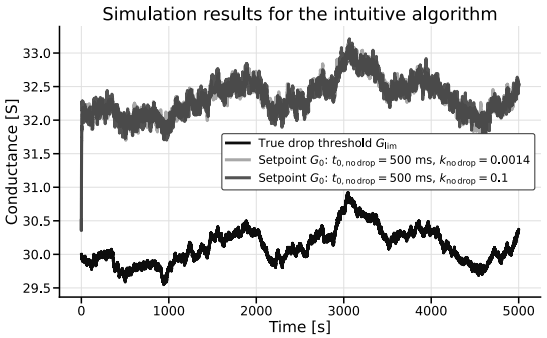
(a) $t_{0,\text{no drop}} = 70$ ms, $k_{\text{no drop}} = 0.0014$



(b) $t_{0,\text{no drop}} = 5000$ ms, $k_{\text{no drop}} = 0.0014$



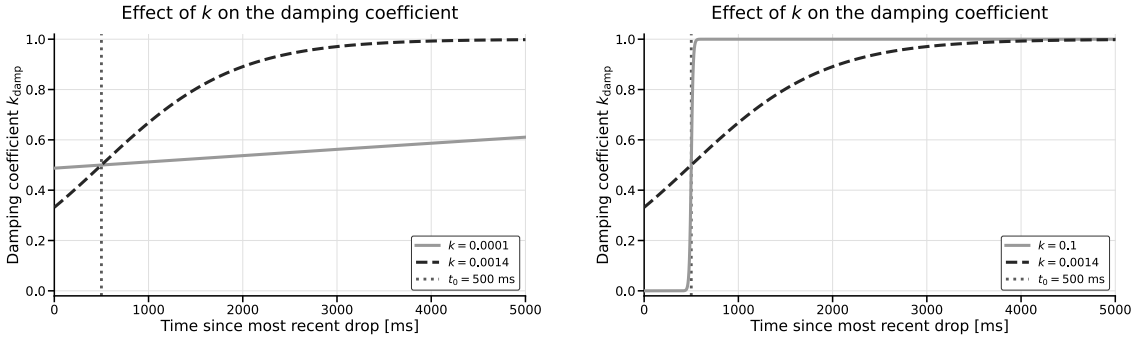
(c) $t_{0,\text{no drop}} = 500$ ms, $k_{\text{no drop}} = 0.0001$



(d) $t_{0,\text{no drop}} = 500$ ms, $k_{\text{no drop}} = 0.1$

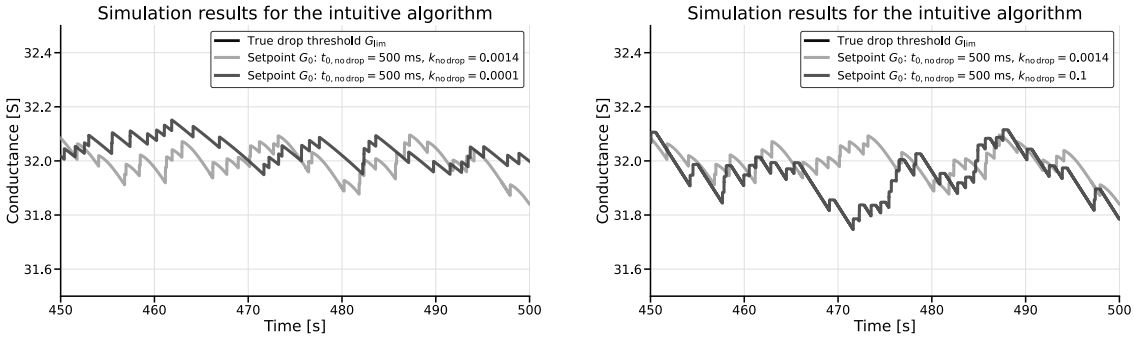
Figure 4.10: Simulation results for the intuitive algorithm using different values of $t_{0,\text{no drop}}$ and $k_{\text{no drop}}$. Each case is compared with the reference parameters $t_{0,\text{no drop}} = 500$ ms and $k_{\text{no drop}} = 0.0014$.

the gain on the actual setpoint can be seen in Figure 4.12. In Figure 4.12 it is shown that a low k value results in a low, close to constant, negative gradient. This is also seen in Figure 4.11 as the gain is close to constant for the first five seconds and the average time between drops is less than two seconds. Furthermore, in Figure 4.12 it is shown that if the k value is too high, we have almost no response until $t_{0,\text{no drop}}$ is reached, then there is a constant negative gradient. This is expected based on the gain seen in Figure 4.11. The main drawback of a k value that is too low is that the algorithm is much slower at reacting to a sudden drop in the threshold. This is illustrated in Figure 4.13, where it is shown that a k value of 0.0001 takes approximately 15 seconds longer to react to a drop in threshold compared to 0.0014, whereas there is no clear difference between 0.0014 and 0.1. Since a slow reaction to a drop in threshold could lead to the setpoint being too high for a period of time and hence result in stubbing, the low k value was not used. The shape of the decrease in setpoint after a drop is different for a k value of 0.0014 and 0.1, and 0.0014 was chosen because of the gradually decreasing gradient. It is however worth noting that no clear difference in performance was shown in the simulation, for drop frequency or reaction to a decrease in threshold between a k value of 0.0014 and 0.1.



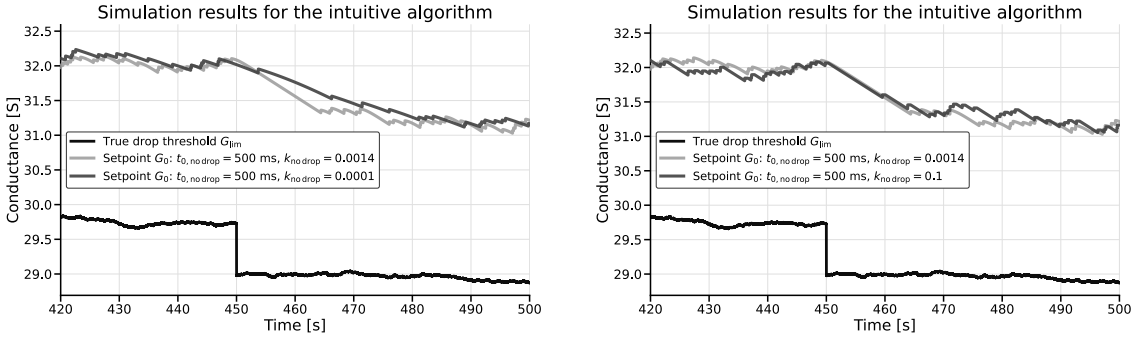
(a) $k_{\text{no drop}} = 0.0001$ **(b)** $k_{\text{no drop}} = 0.1$

Figure 4.11: Damping coefficient as a function of time since the most recent drop for different values of $k_{\text{no drop}}$ compared to $k_{\text{no drop}} = 0.0014$.



(a) $k_{\text{no drop}} = 0.0001$ **(b)** $k_{\text{no drop}} = 0.1$

Figure 4.12: Behaviour of the algorithm for different values of $k_{\text{no drop}}$ compared to $k_{\text{no drop}} = 0.0014$.

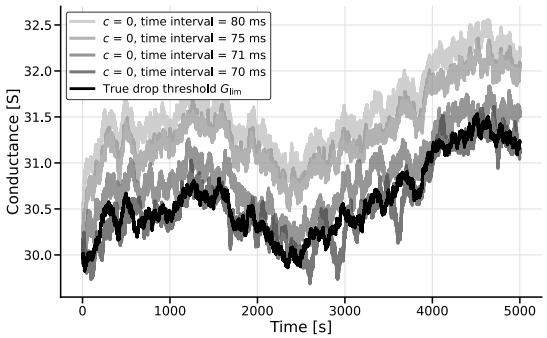


(a) $k_{\text{no drop}} = 0.0001$ **(b)** $k_{\text{no drop}} = 0.1$

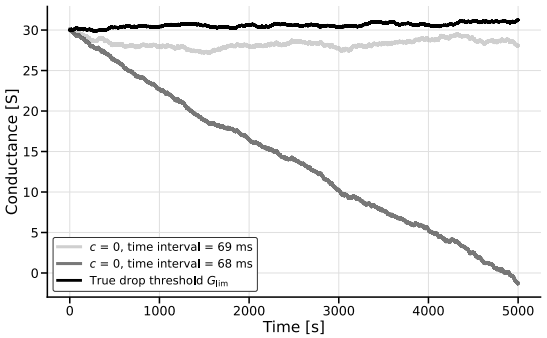
Figure 4.13: Reaction to drop in threshold for different values of $k_{\text{no drop}}$ compared to $k_{\text{no drop}} = 0.0014$.

Time interval algorithm

The main parameter that was altered for the time interval algorithm was the length of the time interval. As described in section 3.4.1 the minimum time between drops was roughly 70 ms. The simulation shows that the result depended heavily on whether the time interval was shorter, equal to or larger than 70 ms. Given a c value of 0, if the time interval is longer than 70 ms, then the setpoint will reach a level over the threshold where the drop frequency corresponds to the time interval. This can be seen in Figure 4.14a, as well as in Table 4.5. For a c value of zero the steps up for drops and down at the end of each interval are close to equal. This explains this behaviour since if the time between drops is shorter than the time interval the setpoint will increase slightly over time until equilibrium is reached and the other way around if the time between drops is longer than the time interval. However since there is a maximum drop frequency the same is not true if the time interval is shorter than the minimum time between drops. Then the process is not allowed to drop faster and in the simulation this results in the setpoint decreasing over time regardless of the threshold, this can be seen in Figure 4.14b. In a real setting this would result in an undesirable process. By introducing the buffer the algorithm was able to follow the threshold even when the time interval was shorter than 70 ms as seen in Figure 4.15. Although the algorithm is able to follow the threshold at these short time intervals, the drop frequency is too high, the drop frequencies for the simulation in which the buffer was used can be seen in Table 4.6. Table 4.6 shows that using this buffer the drop frequency is reduced for a time interval of 70 ms, 104 ms between drops instead of 72, but it is still too high. Further it can be seen that a time interval of 47 ms resulted in 71 ms between drops, and if the interval was reduced to 46 the simulation resulted in a similar behaviour that was seen for a time interval of 68 ms without the buffer in Figure 4.14b. As mentioned in section 3.4.2 Procada estimated that a drop frequency of 2 Hz was acceptable, this corresponds to 500 ms between drops. The algorithm was evaluated for a time interval of 500 ms, the results of the simulation with and without a buffer are shown in Figure 4.16. Figure 4.16 shows that the performance with regard to following the threshold is very similar, even though with the buffer the distance between the threshold and G_0 is slightly larger, as expected. There is a difference between the two with regard to drop frequency. Without the buffer the drop frequency is close to 2 Hz, corresponding to 500 ms between drops, very close to the time interval. With the buffer the drop frequency is notably lower at 1.38 Hz, or 726 ms between drops. Since a time interval of 500 ms leads to a natural buffer between the threshold and G_0 as well as results in a drop frequency of 2 Hz which was acceptable, the algorithm could probably be run without the buffer. However in the performed experiment the buffer was used.



(a) Time intervals slightly larger than 70 ms.



(b) Time intervals slightly lower than 70 ms.

Figure 4.14: Simulated setpoint for time intervals close to 70 ms when no buffer is applied.

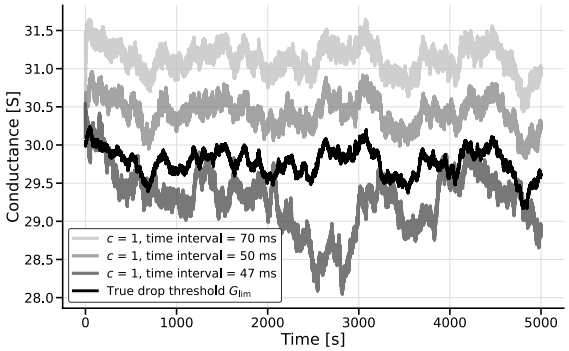


Figure 4.15: Simulated setpoint for time intervals close to 70 ms when a buffer is applied.

Table 4.5: Simulated drop frequency for the time interval algorithm without a buffer.

time interval [ms]	drop frequency [drops / s]	time between drops [ms]
70	13.98	71.54
71	13.79	72.53
75	13.08	76.47
80	12.28	81.44
500	2.01	498.65

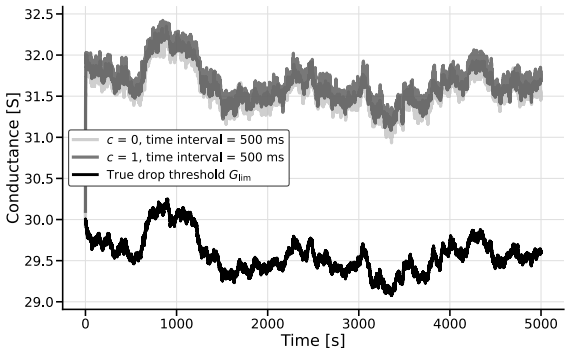


Figure 4.16: Comparison of the simulated setpoint with and without the buffer for a time interval of 500 ms. A value of one corresponds to the buffer being applied.

Table 4.6: Simulated drop frequency for the time interval algorithm with a buffer.

time interval [ms]	drop frequency [drops / s]	time between drops [ms]
47	14.12	70.84
50	13.31	75.12
70	9.62	103.95
500	1.38	726.27

4.5.2 Real process

Note that only one experiment was performed and evaluated. That should be kept in mind when reading comments about the result. Furthermore, the results are from the construction of a cylinder. The experiment was performed on an already partly built cylinder, meaning that the first layer shown in the experiment is deposited on a bead from a previous layer and not on a substrate.
The performance of the algorithm is shown in Figure 4.17. This figure shows the conductance setpoint over time and drops are illustrated by vertical lines. The time at which the setpoint is controlled by the algorithm instead of the operator is indicated by a dashed vertical line. This figure shows the process before and after the time interval algorithm is applied, the intuitive algorithm is not included in

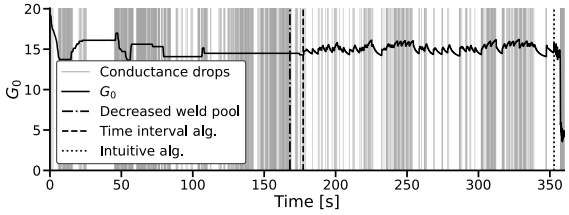


Figure 4.17: Illustration of the experiment performed to evaluate the algorithms. The vertical grey lines indicate drops and the dashed lines indicate different sections of the experiment. The conductance varies close to 15 S both when controlled by an operator and the time interval algorithm. Shortly after the intuitive algorithm was activated the experiment crashed.

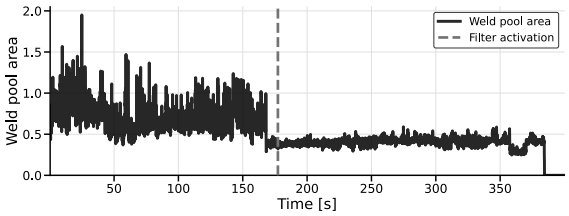


Figure 4.18: Plot of the area of the melt pool over time. The dashed vertical line indicates when the time interval algorithm was activated. Shortly before the algorithm was activated the melt pool decreases in size and the area is more stable.

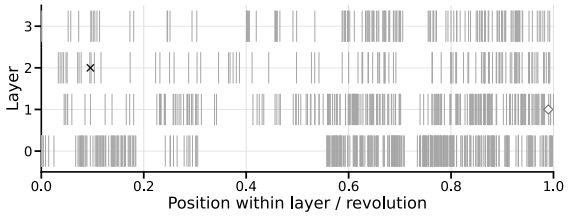


Figure 4.19: Illustration of the drop positions within each layer during the experiment. The marker indicates when the time interval algorithm was activated. The x-axis represents the normalized position within each layer, from the beginning to the end of the layer.

the figure, but will instead be analysed separately. The activation of the algorithm is one event that could influence the results. There is another interesting event that is worth discussing. Figure 4.18 shows the area of the melt pool in pixels based on video data. The algorithm is activated at 177 s. At 168 s the melt pool area decreases and becomes more stable as seen in Figure 4.18. The cause of the decreased size is unknown and has not been investigated. Possible reasons could be that the process reaches a certain temperature [24], or that the oxygen concentration in the tent reaches a certain level [25]. The event is close to the activation of the algorithm, but happened before the algorithm is activated. This implies that the algorithm is not the cause. Furthermore, the effect of this decreased and stable area is also unknown. Since the event is close to the activation of the algorithm it is unclear whether effects observed during the time when the algorithm was active were caused by the algorithm or the decreased melt pool. This means that three different periods were evaluated, the first period from t = 0, until the melt pool is decreased, the short period between the decreased melt-pool area and the activation of the algorithm, and the time from the activation of the time interval algorithm until the intuitive algorithm was applied at t = 350 s.

Process information from the three different time periods is summarized in Table 4.7. The drop frequency is decreased by a factor of 2.5 when the algorithm is active compared to before the decrease in melt-pool area, which is a substantial decrease. Since the algorithm was used with the buffer the drop frequency is also very close to the predicted drop frequency in the simulation seen in Table 4.6, the error is less than 1 %. Since the resulting drop frequency is that close to the prediction, it is reasonable to assume that the frequency is determined by the algorithm and not

by the decreased melt pool. However the effect might be exaggerated. In the short period between the decreased melt pool and the activation of the algorithm the drop frequency is also reduced compared to before the decreased melt pool. Since this period is relatively short, the drop frequency is somewhat unreliable. Assuming that the event at t = 168 s results in a drop frequency of 1.61 Hz, the decrease in drop frequency caused by the algorithm is 15 %. However, both values are below the 2 Hz limit accepted by Procada. Regardless of the drop frequency before the algorithm the experiment implies that the algorithm is able to control the process in a way that leads to a predictable drop frequency, which is more important than the decrease in frequency caused by the algorithm. Further the employees at Procada could confirm that the process seemed to run at least as smooth when the filter was used compared to the operator.

Table 4.7: Drop statistics for the different process periods.

Period	Duration [s]	Drops	Drops/s
No filter active	168.0	588	3.50
Decreased melt pool	9.3	15	1.61
Algorithm active	172.7	240	1.39

Another observation that can be made in Figure 4.17 is that the drops appear in a weak but noticeable pattern. The pattern is more evident in Figure 4.19 where the drops for each layer are shown. The x-axis in Figure 4.19 is the position in each layer. Since the ring that was built was only one bead thick, $x = 0$ represents the start and $x = 1$ the end of each layer. Figure 4.19 shows that if a drop occurs in one of the layers, it is likely that another drop will occur at a similar position but in the next layer. The pattern is observed before and after the algorithm is active. Although the drop frequency is closely related to the setpoint, Figure 4.19 shows that another important factor is the smoothness of the previous layer as reported in the literature [4].
Lastly the intuitive algorithm was evaluated. As seen in Figure 4.17 it was activated at t = 353 s, the process crashed after 4 seconds at t = 357 s and the experiment was terminated. The crash occurred four seconds after activation and was suspected to be related to the intuitive control algorithm.

5

Conclusion and outlook

The purpose of this project was to investigate whether process data could be used to support real-time adjustment of the conductance setpoint and thereby reduce the need for manual supervision. Two main goals were defined: first, to develop and evaluate a model capable of assessing whether the current setpoint was too high or too low based on proximity to a drop state, and second, to design and evaluate an algorithm that used this information to adjust the setpoint in real time.

The first goal was only partially achieved. Several models were developed and evaluated, but the models based on the slow dataset did not perform well enough to reliably predict the probability of drops for previously unseen process runs. Their predictive performance was therefore considered insufficient for use as the basis of a control algorithm. One likely explanation is that the slow dataset lacks information that is important for determining the probability of drops. During one of the experimental runs, the conditions of the previous layer appeared to influence the behaviour of the following layer. Other potentially relevant variables that were not included in the dataset include the wire material and whether the process was a single-bead or multi-bead process.

The CNN based on the fast conductance signal showed more promising results. Using a window shortly before a drop occurred, the CNN was able to identify most drops and distinguish these regions from regions farther from drop formation. However, the model was not developed far enough to reliably determine how the conductance setpoint should be adjusted. Consequently, no control algorithm based on the CNN output was implemented during the project.

The second goal was not fully achieved, since the final control algorithm did not use the output from a predictive model. Instead, a simpler control algorithm based on observed drops and a context-aware filter was developed and evaluated. This algorithm adjusted the conductance setpoint without using the output of a trained predictive model. Only one experimental run was performed, so the results should be regarded as preliminary. Nevertheless, the algorithm produced a drop frequency close to that predicted by the simulations and the process appeared to run at least as smoothly as under manual operator control. These results indicate that automatic adjustment of the conductance setpoint based on observed process events is feasible, although further experiments are required before stronger conclusions can be drawn.

Overall, the project showed that the available slow process data were insufficient for reliable prediction of drop probability, while the high-frequency conductance signal contained information that could be used to identify states shortly before drop formation. The project also demonstrated a preliminary method for automatically adjusting the conductance setpoint without continuous operator intervention. How-

ever, the original goal of combining a predictive model with the control algorithm was not reached.

5.1 Future work

Several aspects should be investigated further before a fully automatic control system can be developed. First, the CNN should be evaluated on additional process runs and further developed to determine whether its predictions can be used directly for control. In particular, false positives could be analysed to determine whether they correspond to near-drop states or precede subsequent drops, rather than representing arbitrary misclassifications.
The available process data could also be expanded with variables that were not considered in the present work. These include information about the previous layer, wire material, and whether the process is single-bead or multi-bead. Including such information may help explain differences between process runs that the current models were unable to capture.
The present work mainly considered drops, which represent only one limit of the desired operating range. Stubbing represents the opposite limit and is generally more detrimental to the process. Future work should therefore investigate methods for detecting or predicting stubbing and incorporate both drop and stubbing information into the control algorithm. This could allow the setpoint to be controlled relative to both limits of the operating window.
The measured process signals contained substantial noise. However, minimal work was performed to investigate the characteristics and sources of this noise. Further analysis aimed at identifying and reducing the noise, both mechanically and through signal processing, could improve the predictive performance of the different models.
Finally, the control algorithm should be evaluated in additional experimental runs and under different process conditions to assess its robustness and general applicability.

Bibliography

[1] A. Kisielewicz et al. “Hot-Wire Laser-Directed Energy Deposition: Process Characteristics and Benefits of Resistive Pre-Heating of the Feedstock Wire”. In: *Metals* 11.4 (2021), p. 634. DOI: 10.3390/met11040634.

[2] Chunyang Xia et al. “A Review on Wire Arc Additive Manufacturing: Monitoring, Control and a Framework of Automated System”. In: *Journal of Manufacturing Systems* 57 (2020), pp. 31–45. DOI: 10.1016/j.jmsy.2020.08.008.

[3] Mohammad Abuabiah et al. “Advancements in Laser Wire-Feed Metal Additive Manufacturing: A Brief Review”. In: *Materials* 16.5 (2023), p. 2030. DOI: 10.3390/ma16052030.

[4] Almir Heralić et al. “Increased Stability in Laser Metal Wire Deposition through Feedback from Optical Measurements”. In: *Optics and Lasers in Engineering* 48.4 (2010), pp. 478–485. DOI: 10.1016/j.optlaseng.2009.08.012.

[5] Petter Hagqvist et al. “Resistance Measurements for Control of Laser Metal Wire Deposition”. In: *Optics and Lasers in Engineering* 54 (2014), pp. 62–67. DOI: 10.1016/j.optlaseng.2013.10.010.

[6] Braden McLain et al. “Machine Vision to Provide Quantitative Analysis of Meltpool Stability for a Coaxial Wire Directed Energy Deposition Process”. In: *Materials* 17.21 (2024), p. 5311. DOI: 10.3390/ma17215311.

[7] Almir Heralic. “Monitoring and Control of Robotized Laser Metal-Wire Deposition”. PhD thesis. Gothenburg, Sweden: Chalmers University of Technology, 2012.

[8] Petter Hagqvist. “Non-intrusive Instrumentation and Estimation: Applications for Control of an Additive Manufacturing Process”. PhD thesis. Gothenburg, Sweden: Chalmers University of Technology, 2015. ISBN: 978-91-7597-148-3.

[9] Almir Heralic and Petter Hagqvist. “Conductance-Based Control System for Additive Manufacturing”. WO 2021/110793. Procada AB. 2021.

[10] Dina Becker et al. “Influence of a Closed-Loop Controlled Laser Metal Wire Deposition Process of S Al 5356 on the Quality of Manufactured Parts before and after Subsequent Machining”. In: *Production Engineering* 15 (2021), pp. 489–507. DOI: 10.1007/s11740-021-01030-w.

[11] Ian T. Jolliffe and Jorge Cadima. “Principal Component Analysis: A Review and Recent Developments”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202. DOI: 10.1098/rsta.2015.0202.

[12] Sijmen de Jong. “SIMPLS: An Alternative Approach to Partial Least Squares Regression”. In: *Chemometrics and Intelligent Laboratory Systems* 18.3 (1993), pp. 251–263. DOI: 10.1016/0169-7439(93)85002-X.

[13] Matteo Stocchero et al. “PLS2 in Metabolomics”. In: *Metabolites* 9.3 (2019), p. 51. DOI: 10.3390/metabo9030051.

[14] Serkan Kiranyaz et al. “1D Convolutional Neural Networks and Applications: A Survey”. In: *Mechanical Systems and Signal Processing* 151 (2021), p. 107398. DOI: 10.1016/j.ymssp.2020.107398.

[15] Maria Vakalopoulou et al. “Deep Learning: Basics and Convolutional Neural Networks (CNNs)”. In: *Machine Learning for Brain Disorders*. Ed. by Olivier Colliot. Humana, 2023. DOI: 10.1007/978-1-0716-3195-9_3.

[16] Gianmarco Baldini et al. “Convolutional Neural Networks Combined with Feature Selection for Radio-Frequency Fingerprinting”. In: *Computational Intelligence* 39.5 (2023), pp. 734–758. DOI: 10.1111/coin.12592.

[17] Haibo He and Edwardo A. Garcia. “Learning from Imbalanced Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284. DOI: 10.1109/TKDE.2008.239.

[18] Kay H. Brodersen et al. “The Balanced Accuracy and Its Posterior Distribution”. In: *Proceedings of the 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 3121–3124. DOI: 10.1109/ICPR.2010.764.

[19] Takaya Saito and Marc Rehmsmeier. “The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets”. In: *PLOS ONE* 10.3 (2015), e0118432. DOI: 10.1371/journal.pone.0118432.

[20] Davide Chicco, Matthijs J. Warrens, and Giuseppe Jurman. “The Coefficient of Determination R-Squared Is More Informative than SMAPE, MAE, MAPE, MSE and RMSE in Regression Analysis Evaluation”. In: *PeerJ Computer Science* 7 (2021), e623. DOI: 10.7717/peerj-cs.623.

[21] Radoslav Ivanov et al. “Continuous Estimation Using Context-Dependent Discrete Measurements”. In: *IEEE Transactions on Automatic Control* 64.1 (2019), pp. 238–253. DOI: 10.1109/TAC.2018.2797839.

[22] Fredric J. Harris. “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform”. In: *Proceedings of the IEEE* 66.1 (1978), pp. 51–83. DOI: 10.1109/PROC.1978.10837.

[23] Donghao Luo and Xue Wang. “ModernTCN: A Modern Pure Convolution Structure for General Time Series Analysis”. In: *The Twelfth International Conference on Learning Representations*. 2024.

[24] Rajib Halder et al. “The Effect of Interlayer Delay on the Heat Accumulation, Microstructures, and Properties in Laser Hot Wire Directed Energy Deposition of Ti-6Al-4V Single-Wall”. In: *Materials* 17.13 (2024), p. 3307. DOI: 10.3390/ma17133307.

[25] Caterina Iantaffi et al. “Oxidation Induced Mechanisms during Directed Energy Deposition Additive Manufactured Titanium Alloy Builds”. In: *Additive Manufacturing Letters* 1 (2021), p. 100022. DOI: 10.1016/j.addlet.2021.100022.

DEPARTMENT OF PHYSICS AND ASTRONOMY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se



CHALMERS
UNIVERSITY OF TECHNOLOGY