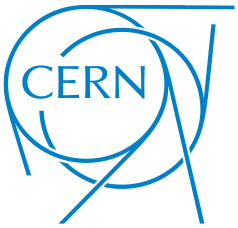
SAPIENZA
UNIVERSITÀ DI ROMA



FACOLTÀ DI INGEGNERIA DELL' INFORMAZIONE
INFORMATICA E STATISTICA
Corso di Laurea Magistrale in Ingegneria Elettronica

Long Short-Term Memory Recurrent Neural Network for the Fully-Automatic Collimator Beam-Based Alignment in the Large Hadron Collider (LHC)

Relatore:
Prof. **Andrea Mostacci**

Correlatore:
Prof. **Marco Balsi**

Supervisori esterni:
Dott.sa **Gabriella Azzopardi**

Dott. **Stefano Redaelli**

Studente:
Gianmarco Ricci

Matricola **1661975**

ANNO ACCADEMICO 2019/2020

Contents

1 Introduction **10**

2 LHC Collimation System **12**
2.1 Collimator Setup 13
2.2 Beam-based Alignment (BBA) procedure 14
2.3 Beam Loss Signal 15
2.4 Phases of the LHC machine cycle 17
2.5 Fully-Automatic Spike Detection for Auto BBA 18
2.6 Time Performance of the Current Fully-Automatic BBA . . 19

3 Background and Literature Review **20**
3.1 Long Short Term Memory (LSTM) Recurrent Neural Network 20
3.1.1 Recurrent Neural Network 20
3.1.2 LSTM networks 21
3.2 Machine Learning Tools Used 24
3.3 Time series classification 27
3.3.1 Range-Doppler Map Time Series Classification 27
3.3.2 Classification of EEG Signals 28

4 Data Analysis and Constraints of the Fully-automatic BBA Spike Detection Algorithm **29**
4.1 Dataset 29
4.1.1 Data Description 29
4.2 Data Analysis 30
4.2.1 Decay Time 30
4.2.2 Feature Correlation 33
4.3 Conclusions 34

5 Alignment Spike Classification with LSTM **36**
5.1 Proposed Framework 36
5.1.1 Network Architecture 36
5.1.2 Network Input 37
5.1.3 Network Ouput 38
5.2 LSTM Training 38
5.3 Comparing to Bidirectional LSTM 40
5.4 LSTM model results 42
5.5 Conclusions 44

6 The New Classification Model for the Automatic BBA **45**
6.1 Model Benchmarking 45
6.2 Proposed Implementation 48
6.3 Example of use case 50
6.4 Conclusions 52

7 Collimator alignment with machine learning for ion beams **53**
7.1 Data analysis 53
7.2 Can supervised learning be used to classify ion beam alignment spike? 55
7.3 Can we predict the decay length? 56
7.4 Can LSTM be used to classify ion beam alignment spike? . . 56
7.4.1 Decay analysis 56
7.4.2 LSTM test on Ion Alignments 58
7.5 Conclusions 59

8 Conclusions **60**

List of Figures

1.1 Interaction points, from [2] 10

2.1 Collimator Hierarchy in the LHC, from [3] 12
2.2 The LHC collimation system at the start of Run 3, from [5] . 13
2.3 The four-stage beam-based alignment procedure for collimator *i*, from [7] 14
2.4 Scraping of the beam halo when the collimator jaw is aligned to the beam. The scattered secondary particles are detected by a BLM downstream and result in a time-varying signal, from [4] 15
2.5 Clear alignment spike 16
2.6 BLM signal while jaw is moving. This is identified as an alignment spike: the rise of loss corresponds to the moment when the jaw touches the beam halo. 16
2.7 BLM signal while jaw is moving. This is identified as a spurious signal above threshold that can arise due to beam instabilities, mechanical vibrations of the opposite jaw when close to the beam, or drifts of the beam position. 16
2.8 Stages of the LHC, from [7] 17

3.1 Unrolled recurrent neural network, from [11] 20
3.2 The repeating module in an LSTM contains four interacting layers, from [11] 21
3.3 Cell state, from [11] 21
3.4 LSTM Gate, from [11] 22
3.5 Forget Gate Layer, from [11] 22
3.6 Input gate layer, from [11] 22
3.7 Update, from [11] 23
3.8 Output, from [11] 23
3.9 Bidirectional LSTM, from [14] 24
3.10 Initially, network error decreases rapidly on all datasets. Soon however it begins to level off and gradually rise on the test set. The dashed line indicates the point of best performance, from [21] 26
3.11 Relationship between training error and testing error, from [21] 26
3.12 Range-Doppler map 27

3.13 The network architecture is a combination of a time distributed convolutional neural network and a LSTM network, from [23] . . . 27
3.14 The structure of EEG signal changes over time, from [24] . . 28
3.15 Proposed model for LSTM, from [25] . . . 28

4.1 Signal exceeding threshold but not corresponding to a jaw aligned (Figure a) and signal exceeding threshold and corresponding to a jaw aligned (Figure b). (BLM values logged at 25 Hz and collimator jaw position moving towards the beam, logged at 1 Hz) . . . 30
4.2 Time required ($t_{\frac{1}{2}}$) for the decaying quantity ($N(t)$) to fall to one half of its initial value, from [27] . . . 32
4.3 Decay time at injection . . . 32
4.4 Decay time at flat top . . . 32
4.5 Examples of two alignment spike with short decay . . . 33
4.6 Heatmap of feature correlation . . . 34

5.1 Network Architecture . . . 37
5.2 Dimension shuffle . . . 38
5.3 Example of time series of BLM signal used for test n. 1 . . . 39
5.4 Example of time series of BLM and position in sigma signals used for test n. 2 . . . 39
5.5 Example of time series of BLM and position in sigma signals used for the test n. 3. The BLM signal is normalized with the last position in sigma logged when the threshold is exceeded. 39
5.6 Precision and Loss of the model trained with BLM signal normalized . . . 40
5.7 Bi-LSTM architecture . . . 41
5.8 Precision and Loss of the model trained with Bi-LSTM . . . 41
5.9 Confusion matrix . . . 42
5.10 Alignment spike correctly classified . . . 42
5.11 Alignment spike correctly classified . . . 43
5.12 Spurious signal correctly classified . . . 43
5.13 Spurious signal correctly classified . . . 44

6.1 Clear Spike Classification . . . 46
6.2 Example of computation of the difference between the BLM signal value when the probability is 80% and the average of the last second of time series . . . 46
6.3 Boxplot of differences between the BLM signal value at different probabilities and the average of the steady state after the spike . . . 47
6.4 Flowchart to align a jaw using new LSTM model . . . 49
6.5 Alignment of the TCSG.A6R7.B2 collimator during 2018 commissioning at flat top, adapted from [5]. . . . 51
6.6 Left Jaw First Spike . . . 51

6.7 Right Jaw First Spike 52

7.1 Heatmap of feature Spearman correlation 54
7.2 Confusion matrix 55
7.3 Ions (Collision Stage) 56
7.4 Decay time ions 57
7.5 Examples of two alignment spike with short decay (the upper plot represents the signal with the smoothing filter applied, the lower represents the original BLM signal) 57
7.6 Confusion matrix 58
7.7 Ion alignment spike correctly classified 59
7.8 Ion spurious signal exceeding threshold correctly classified . 59

List of Tables

2.1 Theoretical minimum time to align a collimator at injection in steps of 10 μm with a time interval of 0.02 seconds, and starting from 8 mm position, from [7] 19

5.1 Network Architecture Layers and Parameters 37
5.2 Network Architecture Layers and Parameters 41

6.1 Variables used in the flowchart in Figure 6.4 48

7.1 Classification report 55
7.2 Classification report 58

Dare mighty things.

- *Perseverance*

Abstract

The Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) is the world's largest particle accelerator. Due to the high energy and high luminosity that the LHC can reach, a complex beam collimation system, comprising some 100 collimators is required to clean beam losses before they damage sensitive equipment. Collimators consist of two movable jaws made of robust materials whose purpose is to dispose of the halo around the proton or ion beams, to prevent quenching of superconducting magnets and protect the machine itself against damage. These jaws are positioned around the beam following well established automated beam-based alignment (BBA) techniques. The latter, use time series of beam losses detected by Beam Loss Monitors (BLM) to classify whether a jaw is aligned or not.

The aim of this thesis is to analyze all aspect underlying the collimators automatic alignment based on machine learning, analyze the data gathered during previous alignment campaigns, determine the possibility of speeding up the automatic collimators alignment and propose a new machine learning model that can accomplish this goal. Since simple neural networks are not suitable for solving sequence problems, a Long Short-Term Memory Recurrent Neural Network has been designed and tested, obtaining an excellent precision on proton data.

The new machine learning model developed has shown to be more efficient than the previous one whilst maintaining the high accuracy needed for such sophisticated collimation system as that of the LHC. The work described in this dissertation is planned to be tested in accelerator operating conditions in Run 3 starting in 2022.

In addition, the currently used supervised learning models have been cross-checked with heavy ions beams, which have different loss patterns, to study the possibility to apply machine learning also to these beams.

Acknowledgments

Foremost, I would like to express my gratitude to Prof. Andrea Mostacci and Dr. Stefano Redaelli for giving me the opportunity to to join the collimation team. I also would like to thank Dr.ssa Gabriella Azzopardi and Prof. Marco Balsi for providing invaluable guidance throughout this research.

A special thanks also to my colleagues and friends Alessandro, Francesca, Matteo and Chiara. This accomplishment would not have been possible without them. Thank you.

Infine, sono estremamente grato ai miei genitori per la fiducia che hanno avuto nei mei confronti.

Chapter 1

Introduction

The Large Hadron Collider (LHC) is the largest and highest-energy particle collider in the world. It was built by the European Organization for Nuclear Research (CERN) in collaboration with over 10,000 scientists and hundreds of universities and laboratories, as well as more than 100 countries. Is installed in a tunnel 27 km in circumference beneath the France–Switzerland border near Geneva. The first collisions were achieved in 2010 at an energy of 3.5 teraelectronvolts (TeV) per beam and after various upgrades it now reaches 6.5 TeV per beam. The LHC tunnel contains two adjacent parallel beam pipes such that two beams travel in opposite directions around the ring. The LHC has four interaction points (IPs) where the four main experiments are located: ATLAS, ALICE, CMS and LHCb [1]. The LHC ring layout is shown in Figure 1.1.

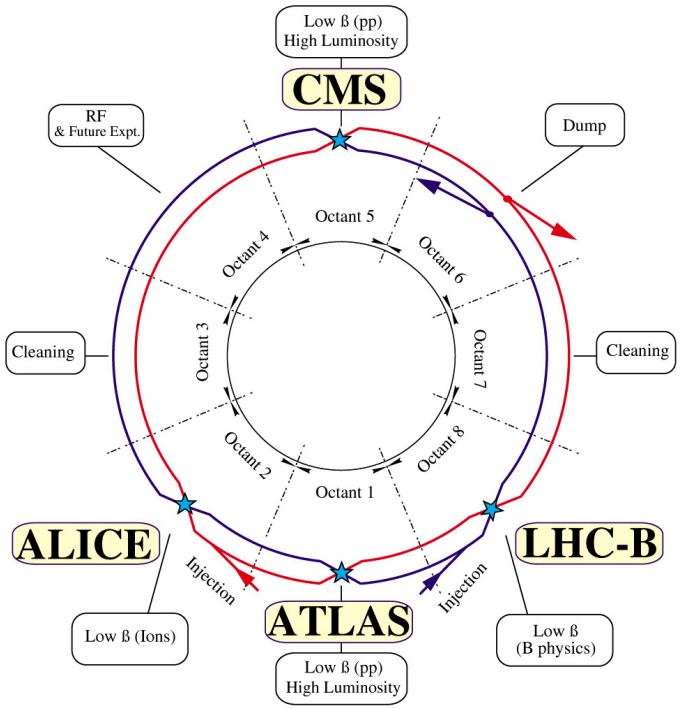


Figure 1.1: Interaction points, from [2]

A total of 1232 super-conducting dipole magnets is used to keep the beams on their circular path, while an additional 392 quadrupole magnets are used to keep the beams focused, such that stronger quadrupole magnets are placed close to the intersection points to maximize the chances of interaction where the two beams cross. The LHC is susceptible to beam losses that may significantly damage the magnets by generating a transition from superconducting to normal conducting state (quench) of the magnets [1].

For this reason, a multistage collimation system is installed, consisting of more than 100 movable devices to absorb such beam losses before they can damage any sensitive equipment. Collimators are aligned to the LHC beam using an automated beam-based alignment (BBA). This technique makes use machine learning to classify time series of beam losses detected by Beam Loss Monitors (BLM). In addition, the possibility to use existing ML algorithms, which were developed for collimator alignments with proton beams, also for the automated alignment of heavy ion beams is explored.

The aim of this thesis is to analyze whether the fully-automatic BBA can be sped up and propose a new method that can achieve this goal.
This dissertation is structured as follows:

- In chapter 2, the LHC collimation System and the beam loss signal available for collimator alignments classification is introduced;
- In chapter 3, the Long Short-Term Memory Recurrent Neural Network is explained in details. In addition, similar works of time series classification are presented;
- In chapter 4, the data analysis that led to understand the constrains of the spike detection algorithm of the fully automatic beam-based alignment is presented;
- In chapter 5, the new Machine Learning model to classify BLM signals is presented, that makes use of the LSTM RNN;
- In chapter 6 it is described, how the new classification model proposed in chapter 5 can be exploited to further enhance the fully-automatic alignment;
- In chapter 7, it is verified if the current spike classification algorithm, used so far in the fully automatic BBA of proton beam, can be also used with ion beam.

Chapter 2

LHC Collimation System

The LHC collimation system is designed to dispose of the beam losses and prevent quenches in the LHC's superconducting magnets. Each collimator consists of two jaws, such that they are placed one on each side of the beam. Each jaw can be independently positioned around the beam. Collimators are positioned trasversally in a multi-stage hierarchy in order to capture primary, secondary and tertiary beam halos. The collimator hierarchy is organized as follows; the primary collimators (TCP) are closest to the beam, followed by the secondary collimators (TCSG), tertiary collimators (TCT), and absorbers (TCLA) (Figure 2.1) [3].

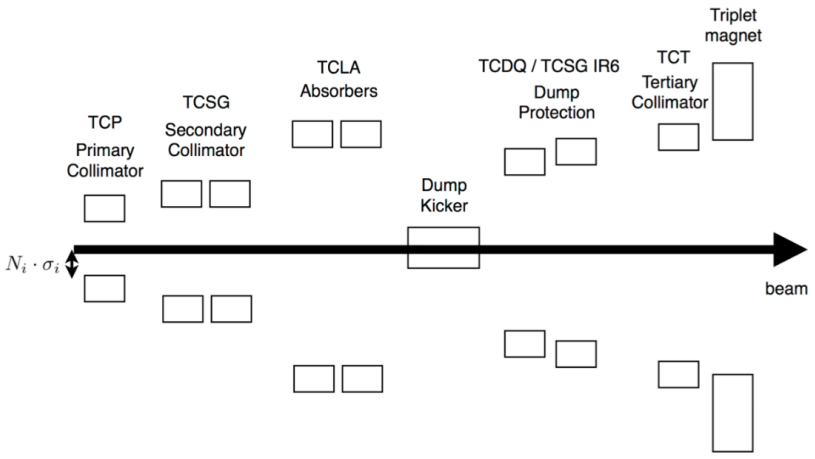


Figure 2.1: Collimator Hierarchy in the LHC, from [3]

2.1 Collimator Setup

During normal operation, it is important to maintain the correct collimator hierarchy to guarantee the required level of machine protection. Therefore every few months of operation, regular beam-based alignments (BBAs) need to be performed due to unpredictable drifts in the beam orbit. Such alignments are required to determine the beam centres and beam sizes at each collimator, from which the hierarchy settings are calculated. Different collimator settings are required at different machine states (injection, flat top, after squeeze or in collisions). According to their cleaning capabilities, they are placed in different Insertion Regions (IRs) in the LHC tunnel at a certain distance, from the beam [4].The collimators are mainly concentrated in two dedicated cleaning insertion regions IR7 and IR3, as shown in Figure 2.2.

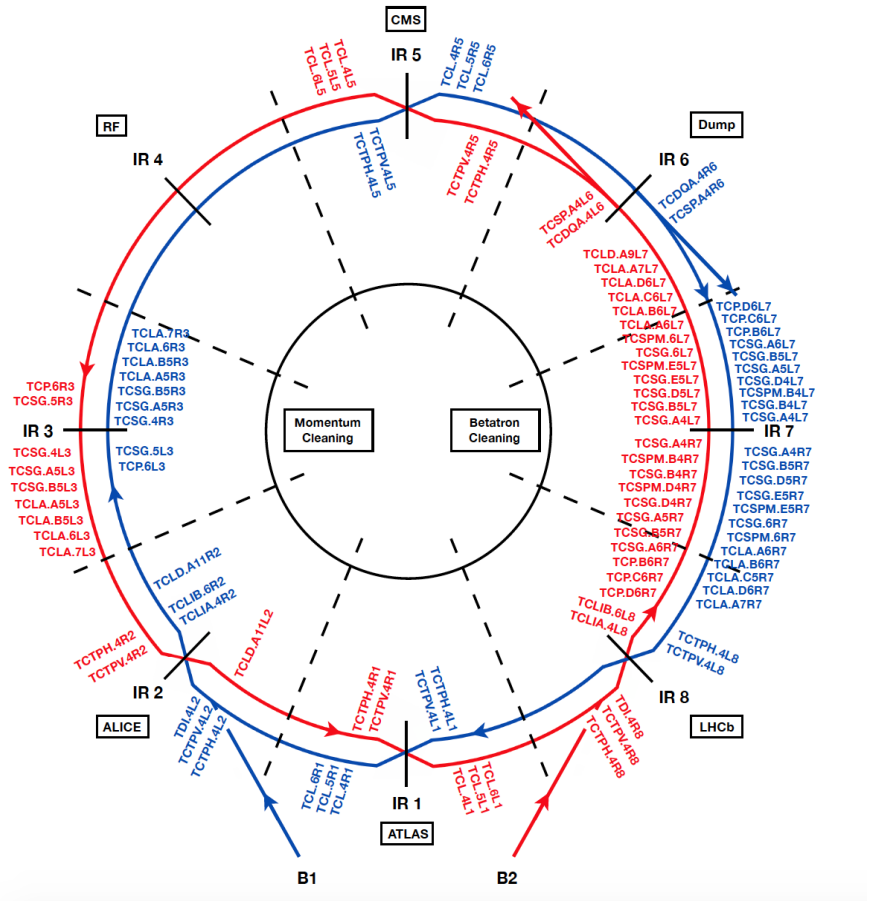


Figure 2.2: The LHC collimation system at the start of Run 3, from [5]

2.2 Beam-based Alignment (BBA) procedure

The beam-based alignment involves aligning each collimator i using a four-step procedure (Figure 2.3), as established in [6].

1. The jaws of a reference collimator are moved in steps towards the beam to form a reference cut in the beam halo (Figure 2.4). The IR7 TCP in the same plane (horizontal, vertical, or skew) as the collimator i is used as a reference collimator;
2. After aligning the IR7 TCP, the same procedure is performed for the collimator i;
3. Then the IR7 TCP is realigned;
4. Finally collimator i is retracted to its position in the hierarchy.

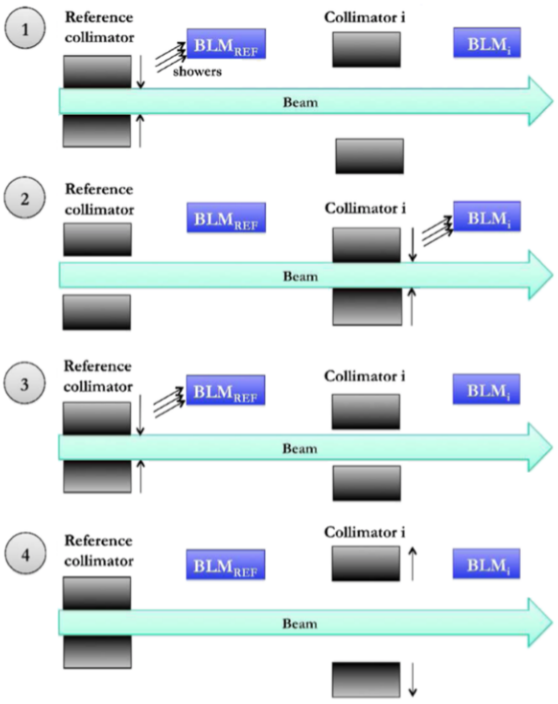


Figure 2.3: The four-stage beam-based alignment procedure for collimator i, from [7]

The beam center can then be determined from the final jaw positions of collimator i:

$$\Delta x_i = \frac{x_i^{L,m} + x_i^{R,m}}{2} \tag{2.1}$$

where $x_i^{L,m}$ and $x_i^{R,m}$ are the measured left and right jaw setup positions. While the beam size is expressed as a function of the half gap, with n_1 being the cut of the reference collimator in units of σ:

$$\sigma_i = \frac{x_i^{L,m} - x_i^{R,m}}{2n_1} \tag{2.2}$$

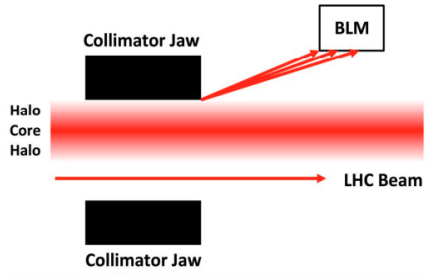


Figure 2.4: Scraping of the beam halo when the collimator jaw is aligned to the beam. The scattered secondary particles are detected by a BLM downstream and result in a time-varying signal, from [4]

2.3 Beam Loss Signal

Beam losses around the ring are detected by a Beam Loss Monitor (BLM) located downstream of each collimator. Such devices, are used to detect beam losses generated when halo particles impact the collimator jaws [7]. A collimator jaw is defined to be aligned to the beam if a jaw movement towards the beam produces an increase of the BLM signal that exceeds a predefined threshold and a clear beam loss signal (Figure 2.5) is observed. BLM thresholds are defined to stop the jaw movement when the alignment spikes are recorded by a nearby BLM, preventing jaw movements deeper into the beam than necessary. The algorithm that automatize the threshold selection have been developed in [8]. In addition to BLMs, inside the LHC are available the Beam Position Monitors (BPM). Such devices, are used to measure the beam trajectory within the machine.

The BLM signal referred to as an alignment spike consists of four phases:

1. a steady-state signal before the spike;
2. the loss spike;
3. a temporal decay of losses;
4. a steady-state signal after the spike.

The steady-state BLM signal is the result of beam dynamics processes such as intrabeam and beam-gas scattering, and increases with the cut made by a collimator jaw in units of beam σ [10]. The loss spike consists in a sharp increase in the beam losses registered by the BLM and is due to the scraping of beam halo particles. The secondary particles formed as a result of the scraping are scattered into the BLM, and ionize the chamber to produce the BLM signal. After the spike, the losses gradually decay [4].

BLM signals can have different shapes as can be seen in the Figures 2.6 and 2.7. In those signals the threshold have been exceeded but not all of them are alignment spikes. A typical alignment spike is shown in Figure 2.6: this signal was generated by a collimator touching the beam indicating

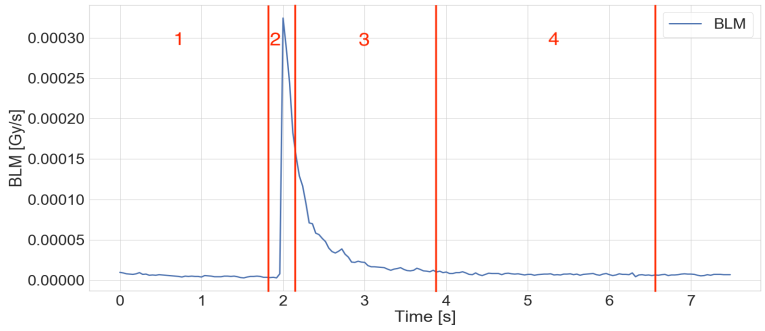


Figure 2.5: Clear alignment spike

that is now aligned. Other signals which do not follow this pattern are classified as spurious signals above threshold. A typical spurious signal above threshold is shown in Figure 2.7: this was not generated by the collimator touching the beam. The latter, doesn't have a fixed structure, and can contain spurious high signals that can arise due to various reasons, such as: beam instabilities, mechanical vibrations of the opposite jaw when close to the beam, or drifts of the beam position [7].

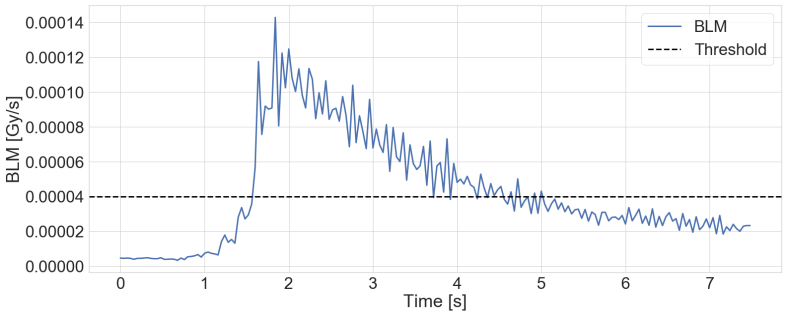


Figure 2.6: BLM signal while jaw is moving. This is identified as an alignment spike: the rise of loss corresponds to the moment when the jaw touches the beam halo.

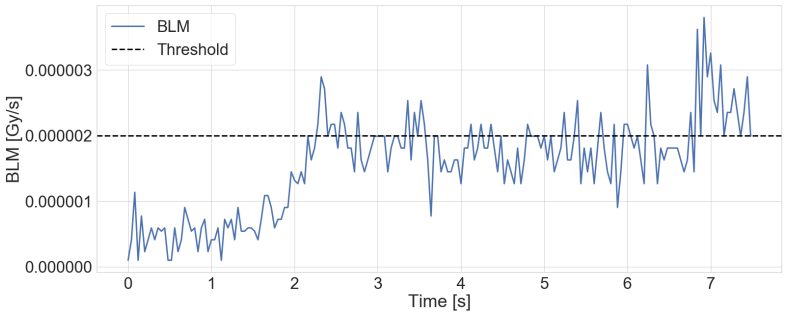


Figure 2.7: BLM signal while jaw is moving. This is identified as a spurious signal above threshold that can arise due to beam instabilities, mechanical vibrations of the opposite jaw when close to the beam, or drifts of the beam position.

2.4 Phases of the LHC machine cycle

The operation of the LHC follows a machine cycle composed of seven stages [7] (Figure 2.8):

1. In the **Injection** stage the accelerator is filled with two 450 GeV beams from the Super Proton Synchrotron (SPS);
2. Once the filling procedure is complete, the beams are **ramped up** until they reach top energy;
3. The third stage is **Flat top**. Between 2015-2018 the LHC was operating at a maximum at 6.5 TeV for proton beams;
4. The fourth stage is the **squeeze** which involves shrinking the beam size at the experiments to increase the rate of collisions;
5. The fifth stage is **stable beams** whereby at this point the beams are brought into collision in stable condition;
6. At any point the beams may be extracted or **dumped**. This operation could be programmed or triggered by the protection systems to avoid damage of the accelerator;
7. Finally, the machine is **ramped down** bringing the magnets to their injection current in preparation for the next fill.

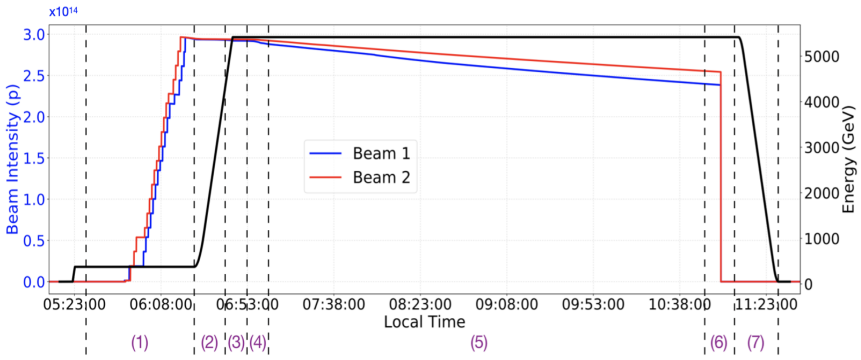


Figure 2.8: Stages of the LHC, from [7]

Collimator alignments are performed at injection, flat top and collision with the use of Beam Loss Monitors, whereas during the squeeze collimators are aligned with Beam Position Monitors.

2.5 Fully-Automatic Spike Detection for Auto BBA

The alignment procedure for the LHC collimators, uses an automatic approach based on supervised machine learning. Once the collimator stops moving, this system makes use of supervised learning to automatically detect an alignment spike by performing the following steps:

1. Movement of the collimator jaw until the threshold is exceeded;
2. Wait for losses to decay. From operational experience the maximum time required for the beam losses to decay is:
   - 4 s at injection;
   - 6 s at flat top.

   When performing alignments it is important not to stop a collimator alignment if the decay has not completely occurred. Otherwise after a few alignment the beam losses would be too high and a beam dump may be triggered.
3. Features extraction; in this step from an initial set of measured data, features (useful for the ML algorithm) are delivered. The values extracted from the BLM signal are:
   (a) Spike height: determined by taking the maximum BLM value observed after the jaws have stopped moving and subtracting from it the average BLM steady state before the spike;
   (b) Exponential fit to the decay of losses:
      i. y-Intercept;
      ii. Gradient;
      iii. Horizontal asymptote;
   (c) Position in sigma: A beam size invariant way of expressing the fraction of the normally distributed beam interrupted by the jaw, as the beam size in mm varies across locations in the accelerator [9].
4. Classification by applying pattern recognition to identify whether there is an alignment spike.The classification task is performed by an Ensemble of six classification models: Logistic Regression, Neural Network, Support Vector Machine, Decision Tree, Random Forest and Gradient Boost. The machine learning models currently in use achieved an accuracy of 96%.

2.6 Time Performance of the Current Fully-Automatic BBA

The time taken to set up collimators is an important indicator of the efficiency of a setup algorithm. The total alignment time, in the present implementation [7], is the sum of different contributions:

- The time to move the collimator jaw from the initial transverse position until it finds/touches the beam reference halo. This is driven by the maximum jaw movement speed that is 1 mm/s;
- The time to measure the beam loss spike with collimator not moving (upon reaching BLM threshold), including the decay time which depends on the beam energy;
- The time to classify the spike, by applying pattern recognition to identify whether the signal corresponds to an alignment spike;
- The time to align the two collimator jaws individually, by observing 2 spikes per jaw;
- The time to re-align the primary collimator jaws. This collimator is already defining the beam halo, therefore only a few steps per jaw are required.

Table 2.1: Theoretical minimum time to align a collimator at injection in steps of 10 μm with a time interval of 0.02 seconds, and starting from 8 mm position, from [7]

Action	**Time [s]**	**Comments**
Move both jaws to 4 mm	8	Assume clear alignment spike first try
Wait for losses to decay	x	Take x as arbitrary wait time
Classify spike	1	Time to classify spike
Align Left Jaw Align Right Jaw	$2(0.1 + x + 1)$ $2(0.1 + x + 1)$	Assume no tilt and beam centre $\sim= 0 \rightarrow$ 2 clear spikes after $\sim$10 steps each
TCP before Left Jaw TCP before Right Jaw	$0.1 + x + 1$ $0.1 + x + 1$	Assume primary was aligned before $\rightarrow$ both jaws with instant clear spikes
TCP after	$2(0.1 + x + 1)$	Left and Right jaws
Total	$17.8 + 9x$	At injection $x \geq 4$, flat top $x \geq 6$

As a result, the theoretical minimum time required to align a single collimator at injection is 53.8 seconds, assuming that every spike is an alignment spike and the collimator is not tilted [7].

Chapter 3

Background and Literature Review

As we have seen in section 2.5 the automatic spike detection before applying the classification algorithm, extracts features to detect local patterns in the BLM time series. Therefore, the efficiency and precision of the classification algorithm is heavily dependent on the quality of variables and on the time. Recent improvements in the deep learning domain offer opportunities to avoid such an intensive feature engineering, an example is the Long Short Term Memory Recurrent Neural Network.

3.1 Long Short Term Memory (LSTM) Recurrent Neural Network

3.1.1 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. The main difference between feedforward and RNN is the presence in the latter, of loops that form infinite sequences in time. This feature allows RNNs to exhibit temporal dynamic behaviour.

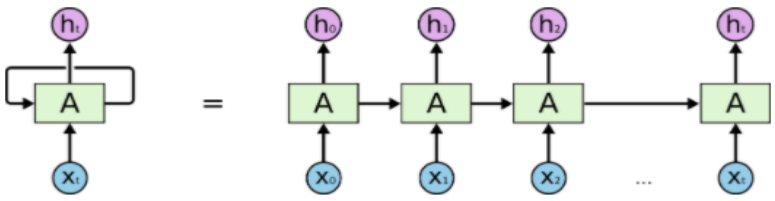


Figure 3.1: Unrolled recurrent neural network, from [11]

In Figure 3.1 a block of an unrolled neural network. A, looks at an input x_t and outputs a value h_t. A loop allows information to be passed from one step of the network to the next [11].

3.1.2 LSTM networks

Long Short Term Memory networks (LSTM) are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber in 1997. LSTM networks are explicitly designed to avoid the long-term dependency problem by remembering information for long periods of time. The latter, is a common problem of RNN that manifests itself if in a time series the gap between the relevant information and the place that its needed is large making impossible for the newtwork to remember past information. LSTMs also have a chain-like structure similar to RNNs, but the repeating module has a different structure whereby instead of having a single neural network layer, there are four [11].

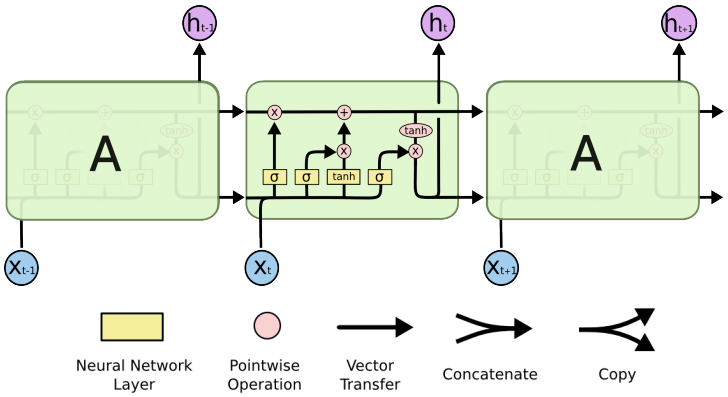


Figure 3.2: The repeating module in an LSTM contains four interacting layers, from [11]

In Figure 3.2, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represents point-wise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations [11].

Core idea

The key to LSTMs is the cell state, represented by the horizontal black line in Figure 3.3. It runs straight through the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged [11].

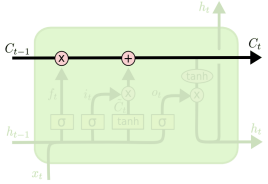


Figure 3.3: Cell state, from [11]

The LSTM has the ability to remove or add information to the cell state, carefully regulated by structures called gates (Figure 3.4). Gates are a way to optionally let information through. They are composed of a sigmoid neural network layer and a point-wise multiplication operation [11].

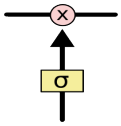
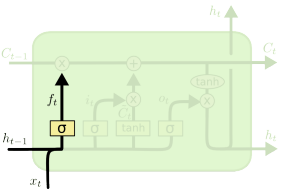


Figure 3.4: LSTM Gate, from [11]

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through.

First step

The **first step** in the LSTM is to decide what information needs to be removed from the cell state. This decision is made by a sigmoid layer called the "forget gate layer". It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 means that the information needs to be saved while a 0 represents that the information needs to be discarded [11].

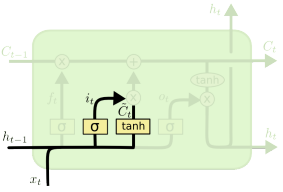


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 3.5: Forget Gate Layer, from [11]

Second step

The next step is to decide what new information needs to be to stored in the cell state. This process has two phases. First, a sigmoid layer called the "input gate layer" decides which values will be updated. Next, a hyperbolic tangent activation layer (that applies the tanh function on the layer inputs) creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state [11].



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 3.6: Input gate layer, from [11]

Third step

The **third step** consists of updating the previous state of the cell (C_{t-1}) to the new state (C_t). It multiplies the old state (C_{t-1}) by f_t (output of the first step), in this way the information that was deemed for discarding in the first step, is eliminated. Then add the product $i_t * \tilde{C}_t$ which represent the new values of the state [11].

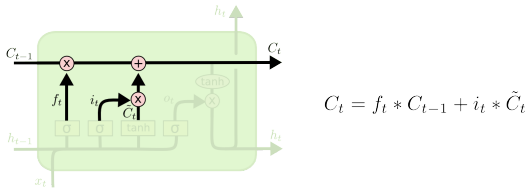


Figure 3.7: Update, from [11]

Fourth step

Finally in the **fourth step** we define the output which is based on the filtered values contained in the state. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to [11].

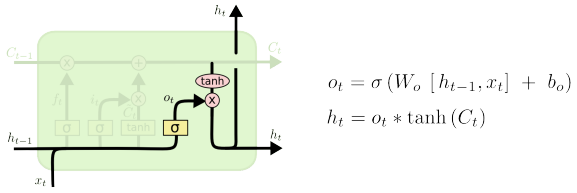


Figure 3.8: Output, from [11]

Bidirectional LSTM

Bidirectional LSTMs are an extension to typical LSTMs that can improve performance of the model on sequence-classification problems [12]. Bi-LSTMs consist of two LSTM forward and backward layers that utilizes additional information and enhance memory capability. The first layer processes the input time series as it is, the second layer takes a reversed copy. As illustrated in Figure 3.9, Bi-LSTM first computes the forward hidden sequence $\overrightarrow{h}$ and the backward hidden sequence h respectively, and then combines $\overrightarrow{h_t}$ and h_t to generate output y_t [14].

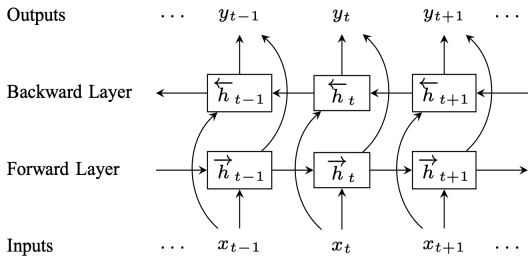


Figure 3.9: Bidirectional LSTM, from [14]

Simple neural networks are not suitable for solving sequence problems, since in these problems account should also be taken of the previous inputs as well. Neural Networks with memory as LSTM are more suited to solving sequence problems [13]. Since LSTMs are effective at capturing long-term temporal, they have been used to advance the state of the art for many difficult problems. This includes handwriting recognition and generation, language translation, speech synthesis, prediction of future values of a time series, audio analysis, and video data among others.

3.2 Machine Learning Tools Used

Dropout Layer

Dropout Layer is a regularization techniques used to reduce overfitting in the deep learning models. Overfitting, is a problem that occurs when a machine learning model shows more accuracy on the training data but less accuracy on the test data or unseen data. The dropout layer avoids this problem by randomly setting input units to 0 at a predefined frequency at each step during training time [15].

Dense Layer

The dense layer is a neural network layer that is fully connected, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer performs a matrix-vector multiplication where values used in the matrix are weights that can be trained and updated with the help of backpropagation. The output generated by the dense layer is an 'm' dimensional vector that represents the number of outputs of the machine learning model [16].

Activation function

The activation function introduces the non-linearity into neural networks and allows them to learn more complex models. In this work we adopt *sigmoid* function. The latter, has the property to map the entire range in

numerical sets into a small range such as between 0 and 1. One common use of a sigmoid function in machine learning is to convert a real value into one that can be interpreted as a probability [17].

$$S(t) = \frac{1}{1 - e^{-t}} \tag{3.1}$$

Loss function

Loss functions for classification are computationally feasible functions representing the price paid for inaccuracy of predictions in classification problems. The loss function used in this dissertation is the *binary crossentropy* that is specifically used for binary classification tasks [18].

Adam Optimizer

Optimizers are used to update weights and biases in a neural network, to reduce the error. The Adam optimization algorithm (used in this work) is an extension to stochastic gradient descent. This optimizer is computationally efficient, appropriate for non-stationary objectives, appropriate for problems with very noisy/or sparse gradients, well suited for problems that are large in terms of data and/or parameters and has little memory requirements [19].

Early Stopping Technique

Early stopping is a form of regularization used to avoid the problem of overfitting while training a learner with an iterative method, such as adam optimization. Such methods update the learner in order to make it better fit the training data at each iteration. Early stopping rules provide guidance as to how many epochs can be run before the model begins to have better prediction results on the train set rather than on the test set. For instance in the model loss diagram in Figure 3.10 it can be seen that the in order to get the model properly trained it is enough to only use 3 epochs because after that point the loss on the test set start increasing while the the loss on training set continues decreasing. Another diagram that shows the relationship between training error and testing error is shown in Figure 3.11 [20].

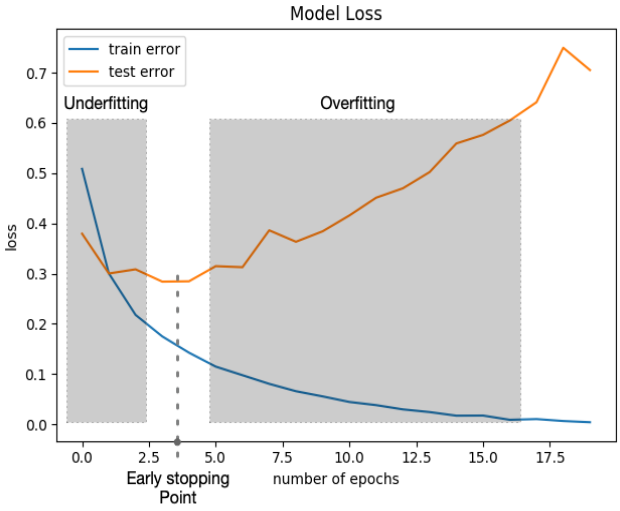


Figure 3.10: Initially, network error decreases rapidly on all datasets. Soon however it begins to level off and gradually rise on the test set. The dashed line indicates the point of best performance, from [21]

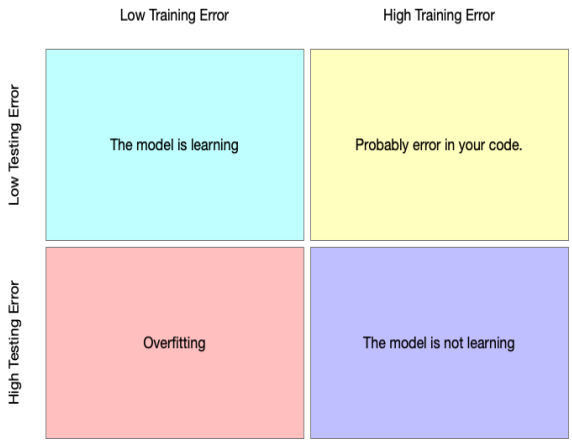


Figure 3.11: Relationship between training error and testing error, from [21]

Z-Score

Z-Score is a technique to rescale the features such that they have the properties of a standard normal distribution with mean, $\mu = 0$ and standard deviation, $\sigma = 1$ [22]. Standard scores of the samples x are calculated as follows:

$$z = \frac{x - \mu}{\sigma} \tag{3.2}$$

3.3 Time series classification

Researchers developed many models for classification of time series using LSTM RNN. In this section two applications of LSTMs are reviewed.

3.3.1 Range-Doppler Map Time Series Classification

Recent work in [23] developed a model to classify time series of Range-Doppler maps. Range-Doppler maps are two-dimensional matrices containing distance and velocity information. These matrices can be interpreted as grayscale images and typical image classification neural network layers can be used. A time series or sequence of Range-Doppler maps adds a third dimensionality regarding time [23].

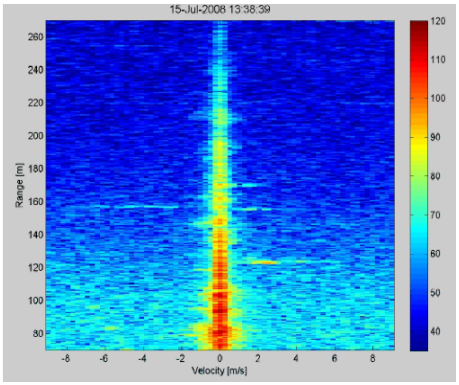


Figure 3.12: Range-Doppler map

The model shown in Figure 3.13 consists of a combination of convolutional and recurrent neural network layers. The time distributed convolutional part is used as a feature extraction network. The recurrent part does the classification considering the current and previous Range-Doppler maps [23].

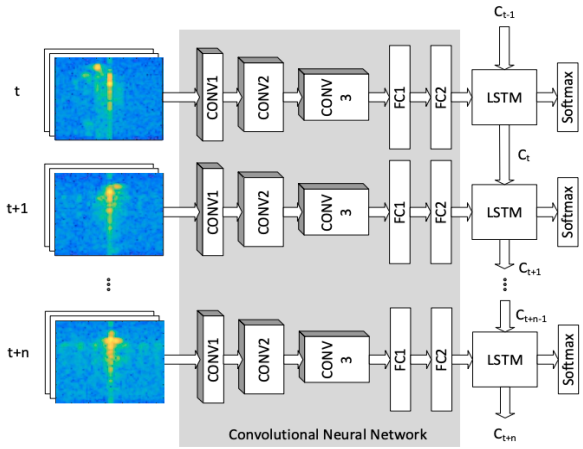


Figure 3.13: The network architecture is a combination of a time distributed convolutional neural network and a LSTM network, from [23]

A dataset of 30,000 Range-Doppler maps in six different classes was created along cooking processes. The RangeDoppler maps are divided in 1.500 sequences of length 20 each (2.6 seconds long). The network's performance was evaluated and the scored accuracy for a 50%-50% split dataset is 98.02% [23].

3.3.2 Classification of EEG Signals

Electroencephalography (EEG) is one of the electrophysiological tests commonly used to record different responses of brain's neural network with different conditions. Recent work in [25] provided further evidence that

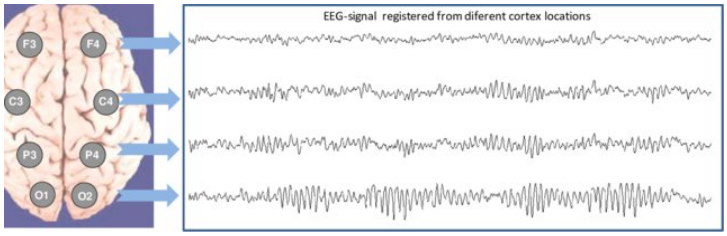
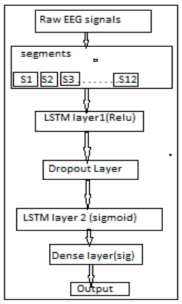


Figure 3.14: The structure of EEG signal changes over time, from [24]

LSTM networks can be used effectively for time series classification problems. The dataset used in [25] is composed of 500 multivariate time series pre-processed integer real valued data of 23.6 seconds each. The EEG recordings are divided into 4 different categories based on different brain activities/disorders:

1. EEG from tumour located area;
2. EEG of normal healthy brain;
3. EEG signals in eyes closed condition;
4. EEG signals in eyes open condition.



0.25

Figure 3.15: Proposed model for LSTM, from [25]

The accuracy scored in classification of time series of the four classes listed, with a learning rate of 0.01 and for 300 epochs is of 71% [25].

Chapter 4

Data Analysis and Constraints of the Fully-automatic BBA Spike Detection Algorithm

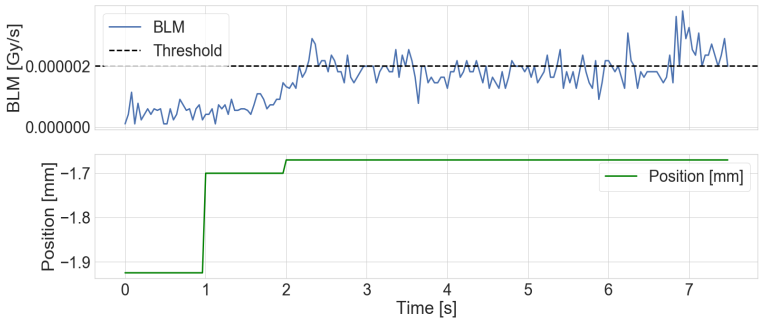
This chapter presents an analysis of the spike detection algorithm used as part of the automatic beam-based alignment and will lead to identify possible improvements.

4.1 Dataset

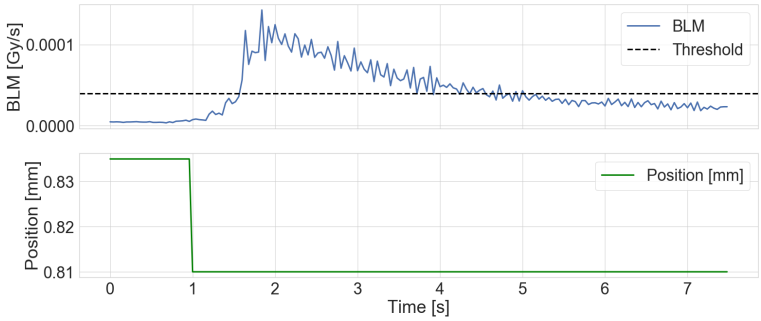
The dataset consists of collimator alignment campaigns performed in 2016 and 2018, both at injection and at flat top.

4.1.1 Data Description

The data logged during various 2016 alignment campaigns consists of 100 Hz BLM signals recorded at each collimator and jaw positions logged (in mm) at a frequency of 1 Hz. Alignment campaigns in 2018 used instead, 25 Hz BLM signals while the collimator jaw positions was acquired at the same frequency. In order to use both data sets and make them consistent, the 2016 BLM data have been down-sampled to 25 Hz by taking 1 sample every four of the original 100Hz signal. The time window taken into consideration for the spike analysis starts 2 s before the BLM threshold is exceeded up until the spike is decayed. As a result, the dataset is composed of time series of 7.5 s of BLM signals and collimator positions. The BLM signal in each data sample was individually analyzed and labelled by experts into two classification classes (alignment and non-alignment spikes), as per the example shown in Figure 4.1. The dataset used in this work includes a total of 2980 samples, 1551 classified as alignment spikes and 1429 classified as non-alignment spikes.



(a) Spurious signal above threshold (Label = 0)



(b) Alignment Spike (Label = 1)

Figure 4.1: Signal exceeding threshold but not corresponding to a jaw aligned (Figure a) and signal exceeding threshold and corresponding to a jaw aligned (Figure b). (BLM values logged at 25 Hz and collimator jaw position moving towards the beam, logged at 1 Hz)

4.2 Data Analysis

The purpose of this section is to study whether the fully-automatic alignment can be sped up from a better understanding of the BLM signal used for spike detection. As described in section 2.5 the decay time in the current spike classification algorithm is fixed to the maximum decay time, therefore this section provides a more in-depth look of the decay length, to determine possible gain in time.

4.2.1 Decay Time

As already mentioned in section 2.5 the beam loss decay is fixed at 4 seconds at Injection and 6 seconds at Flat Top, as these are approximately the maximum time required for BLM signals to decay at the respective machine states. This section will analyze the actual length of decay required at each machine state, to identify possible room for improvement when detecting spikes.

Mean Lifetime and Half-Life

Definition 4.2.1 *Mean Lifetime represents the average time that an element remains in a set [26].*

Definition 4.2.2 *Half-Life represents the time required for the decaying quantity to fall to one half of its initial value [26].*

Both equations can be derived starting from the equation that describes exponential decay of a quantity N.

$$\frac{dN}{dt} = -\lambda N, \tag{4.1}$$

where λ is the decay constant of the decaying quantity. Applying the separation of variables technique to Equation 4.1:

$$\frac{dN}{N} = -\lambda dt. \tag{4.2}$$

Integrating in time the 4.2 results in:

$$\ln N = -\lambda t + C, \tag{4.3}$$

where C is the constant of integration, and hence:

$$N(t) = N_0 e^{-\lambda t}. \tag{4.4}$$

Were $N_0 = e^C$ is the initial value of N, $N_0 = N(t = 0)$ at the start of the decay.

From Equation 4.4 one can derive the **Mean Lifetime**, in which the exponential time constant, τ, relates to the decay rate, λ, in the following way:

$$\lambda = \frac{1}{\tau}. \tag{4.5}$$

Given a process of exponential decay, we can ask how long it would take for half of the original amount to remain. By setting $t = T_{\frac{1}{2}}$ and $N\left(T_{\frac{1}{2}}\right) = \frac{N_0}{2}$ in Equation 4.4:

$$N\left(T_{\frac{1}{2}}\right) = \frac{N_0}{2} e^{-\lambda T_{\frac{1}{2}}} \tag{4.6}$$

Dividing Equation 4.6 by N_0 and taking the logarithm leads to:

$$\ln \frac{1}{2} = -\lambda T_{\frac{1}{2}} \tag{4.7}$$

Solving 4.7 for $T\frac{1}{2}$ finally gives:

$$T_{\frac{1}{2}} = \frac{\ln 2}{\lambda} \tag{4.8}$$

Equation 4.8 represents the **Half Life** (also depicted in Figure 4.2) [26].

Equations 4.5 and 4.8 have been used to compute the decay length for a subset of the alignment spikes in the dataset (649 BLM time series at injection and 684 at flat top). The calculation have been done by taking as

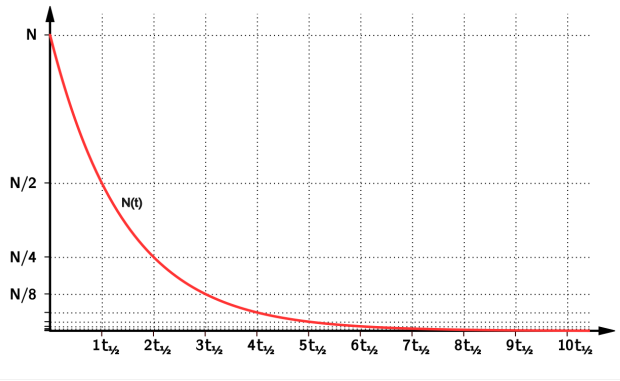


Figure 4.2: Time required ($t_{\frac{1}{2}}$) for the decaying quantity ($N(t)$) to fall to one half of its initial value, from [27]

λ the gradient of the exponential fit for every alignment spike. The result of this analysis is shown in Figures 4.3 and 4.4. In particular from Figure 4.3 it is possible to see that the average ('+' symbol) decay time at injection is 1.44 s while at flat top it is 3.34 s (Figure 4.4). This result shows that it is not necessary to always wait 4 s at injection or 6 s at flat top for the spike to decay but that the classification of an alignment can be done much earlier to save time.

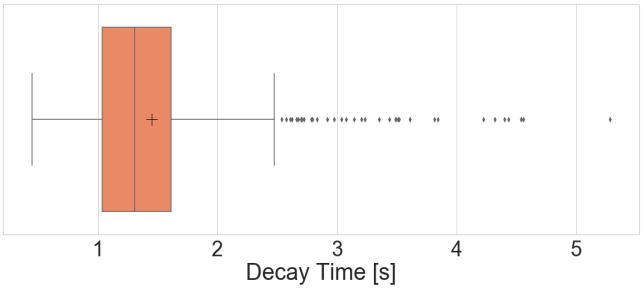


Figure 4.3: Decay time at injection

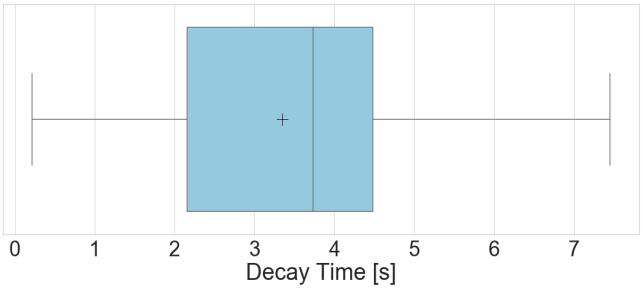
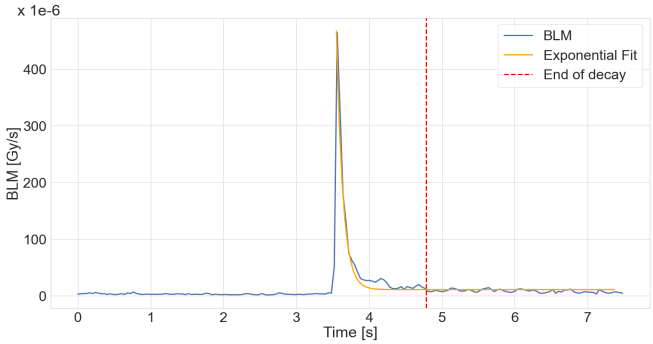
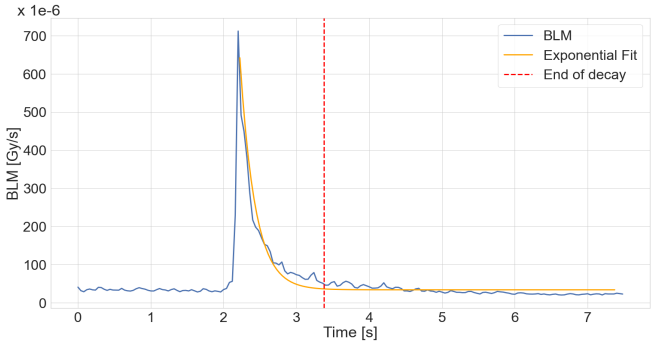


Figure 4.4: Decay time at flat top



(a) Example of short decay at injection



(b) Example of short decay at flat top

Figure 4.5: Examples of two alignment spike with short decay

Figure 4.5 shows two examples of spike alignment with short decay. In Figure 4.5a the decay time is less than 1 s while in Figure 4.5b is around 1.4 s.

4.2.2 Feature Correlation

Spearman Correlation

The Spearman's rank coefficient of correlation is a nonparametric measure of rank correlation (statistical dependence of ranking between two variables). It assesses how well the relationship between two variables can be described using a monotonic function. The sign of the Spearman correlation indicates the direction of association between x (the independent variable) and y (the dependent variable). If y tends to increase when x increases, the Spearman correlation coefficient is positive and viceversa (if y tends to decrease when x increases, the Spearman correlation coefficient is negative). A Spearman correlation of zero indicates that there is no tendency for y to either increase or decrease when x increases. The Spearman correlation increases in magnitude as x and y become closer to being perfectly monotone functions of each other. When x and y are perfectly monotonically related, the Spearman correlation coefficient becomes 1 [28].

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \tag{4.9}$$

This analysis was conducted to determine if the decay length can be predicted from other features. In order to do this, the Spearman correlation between them was calculated. The features are:

1. Maximum value of the signal;
2. Spike height (Maximum value of the spike - BLM average before the spike);
3. BLM average before the spike;
4. BLM average after the spike;
5. Decay Length (calculated with Equation 4.8);

The result of this analysis is displayed in Figure 4.6 where we can observe that the decay length of the spike is not significantly correlated with any other feature. On the other hand, the BLM average before the spike, BLM average after the spike and spike height, are correlated with the maximum value of the spike. The spike height and the mean before the spike are also correlated with the mean after the spike. From these correlations we can conclude that when the mean of BLM before the spike is high then the maximum value is high, however these two things are not correlated with the duration of the decay. This analysis shows that the length of the decay cannot be predicted based on other features in the BLM signal.

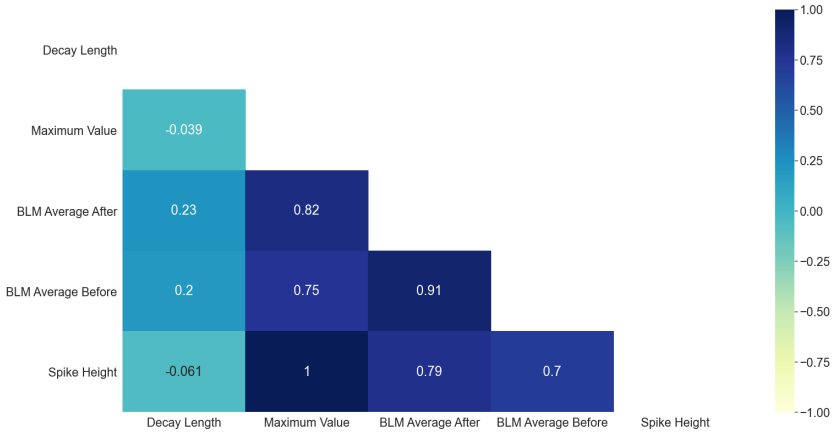


Figure 4.6: Heatmap of feature correlation

4.3 Conclusions

Two types of data analysis were performed in this chapter. The first has shown that the collimator automatic alignment can be sped up through a real-time classification of the signal coming from the beam loss monitors, because the decay time used in the present implementation is larger than the average value found in measurement. The second analysis has shown that

the decay length cannot be predicted form other features such as decay length, BLM average before the spike, BLM average after the spike, and spike height. In the next chapter we will focus on the development of the new model for alignment spike classification.

Chapter 5

Alignment Spike Classification with LSTM

This chapter demonstrates that one can to use LSTM to classify BLM signals in real-time to speed-up the fully automatic alignment.

5.1 Proposed Framework

In this section the proposed solution is to explore a new classification method for BLM signals. Three tests on three different datasets have been conducted. The model developed for end-to-end time series classification does not require heavy pre-processing on the data or feature engineering.

5.1.1 Network Architecture

The network was developed with the use of the deep learning library Keras [29] with TensorFlow [30] for the backend. The full network is depicted in Figure 5.1. The details of the network are listed in Table 5.1 which includes:

- Two LSTM layers with 128 and 64 hidden neurons, respectively. These are used to process the learning data produced by the preprocessing unit;
- A dropout layer with rate of 0.4;
- A dense layer with one neuron and sigmoid activation function;
- Binary crossentropy as loss functions;
- Adam optimizer with a learning rate of 0.0003;
- Early stopping technique with a 'patience' (number of epochs with no improvement after which training will be stopped) of 3 and 'min' mode, which stops the training when the quantity monitored has stopped decreasing.

Since of the small number of model layers, the grid search method was used to adjust the parameters.

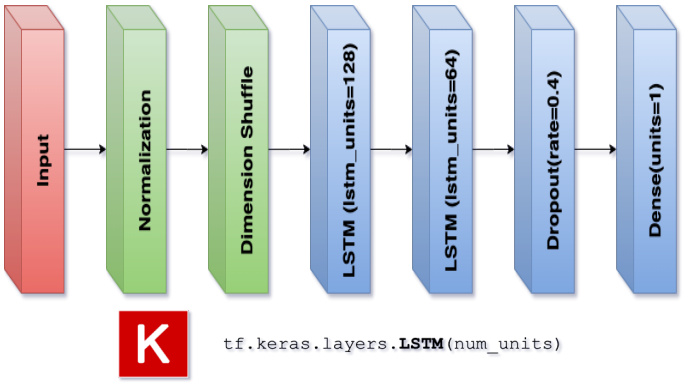


Figure 5.1: Network Architecture

Table 5.1: Network Architecture Layers and Parameters

Layer	Output Shape
LSTM	(None, 188, 128)
LSTM	(None, 64)
Dropout	(None, 64)
Dense	(None, 1)

5.1.2 Network Input

The network inputs are single variate of BLM values or multivariate time series of BLM values and position in sigma of fixed length.

Definition 5.1.1 *Univariate time series is a sequence of data points, measured typically at successive points in time spaced at uniform time intervals. A univariate time series can be denoted as* $T = (t_1, t_2, ..., t_n)$*, and* n *is the length of* T *[31].*

Definition 5.1.2 *Multivariate time series is a set of time series with the same timestamps. For a multivariate time series* M*, each element* m_i *is a univariate time series. At any timestamp* t*,* $m_i = (m_{1t}, m_{2t}, ..., m_{lt})$*, where* l *is the number of univariate time series in* M *[31].*

Before feeding the data into the LSTM layer a Z-Score normalization (Equation 3.2) at each sample separately is applied. Lastly in order to give a three-dimensional array as an input to the LSTM network a dimension shuffle is applied (Figure 5.2). The first dimension represents the batch size, the second dimension represents the time-steps and the third dimension represents the number of features in one input sequence. For example, in the case under study the input shape is (*none*, *time_steps* = 188, *features* = 1).

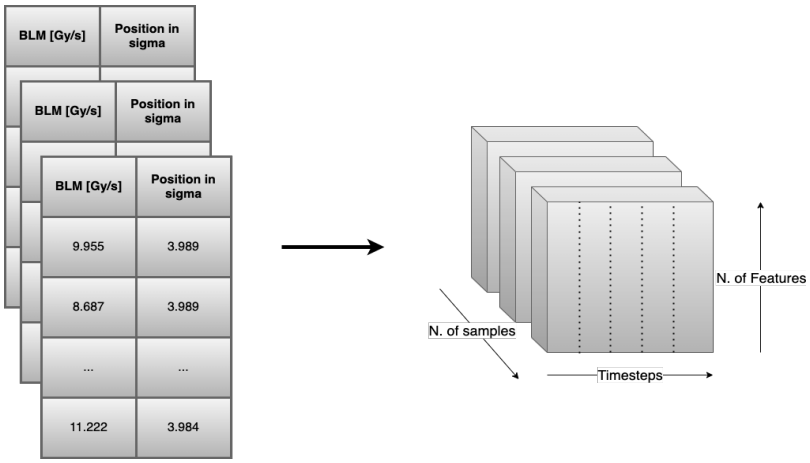


Figure 5.2: Dimension shuffle

5.1.3 Network Ouput

An argument of the LSTM layer is *return_sequences*. This argument specifies whether to return the output at each time step instead of the final time step. If *return_sequences* is set to "True", the output shape becomes a 3D array, instead of a 2D array. For this model the parameter *return_sequences* was set to "False", so the output is returned on the final time step. The output of the dense layer is a probability, precisely the probability that the time series fed into the network is a spike alignment.

5.2 LSTM Training

The data described in subsection 4.1.1 are time series of BLM signal and collimator position. From these available data, three datasets were considered for training the LSTM in three different experiments:

1. In the first test the model was trained using single variate time series of BLM (Figure 5.3). This model did not satisfy our requirements as we need to consider the collimator position;

2. In the second test the model was trained using multivariate time series of BLM and Position in sigma taking into account all the possible variation of the collimator position (Figure 5.4). BLM signal is recorded at 25Hz and the position of the collimator is recorded at 1Hz, this causes the 2 time series to be out of sync making this dataset not suitable for training;

3. In the third test the model was trained using the BLM signal normalized with the final collimator position (Figure 5.5) at the point of classification as the current implementation, in [7], does.

In all experiments we utilize the same structure, and number of epochs of the model explained in subsection 5.1.1.

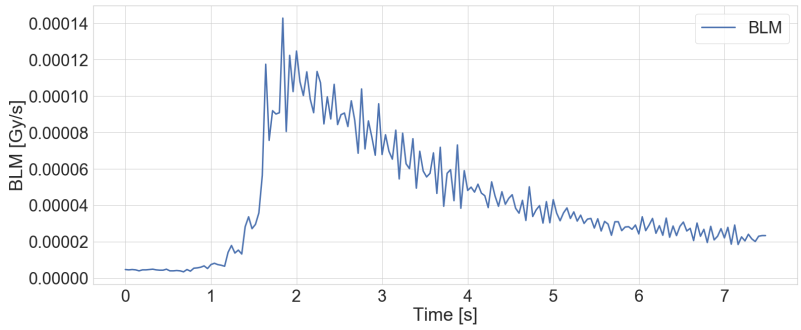


Figure 5.3: Example of time series of BLM signal used for test n. 1

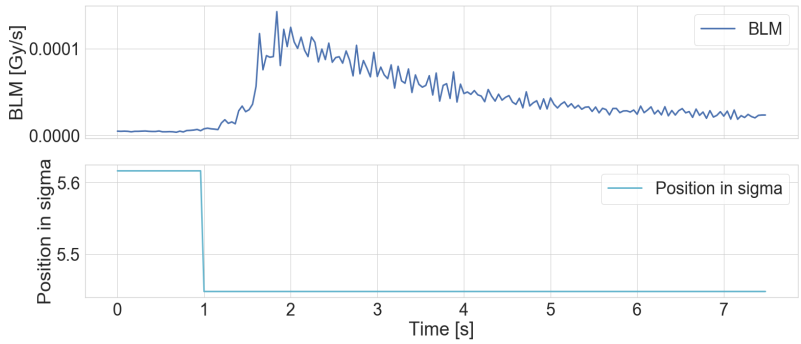


Figure 5.4: Example of time series of BLM and position in sigma signals used for test n. 2

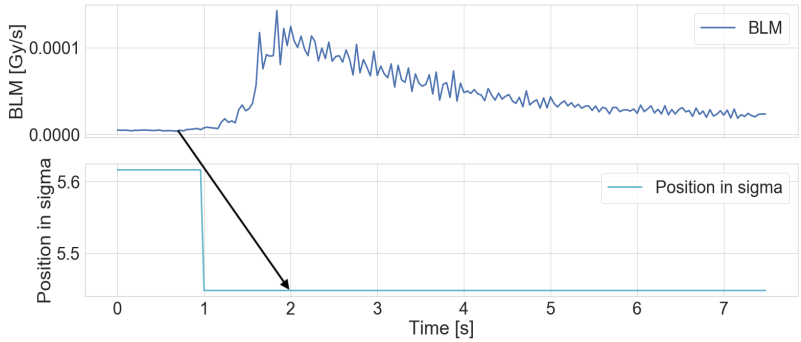


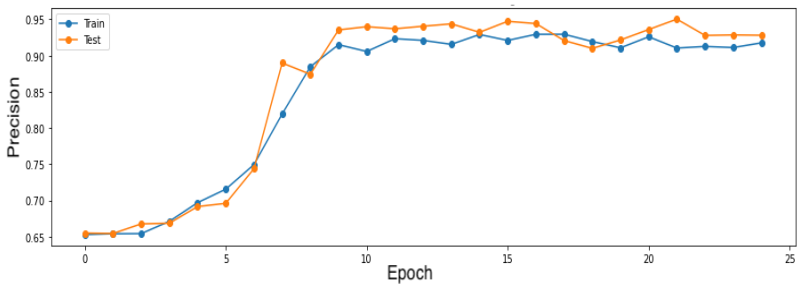
Figure 5.5: Example of time series of BLM and position in sigma signals used for the test n. 3. The BLM signal is normalized with the last position in sigma logged when the threshold is exceeded.

Since false detection of an alignment spike is more grievous than not detecting an alignment spike, precision[1] will be used as the main performance metric [7].

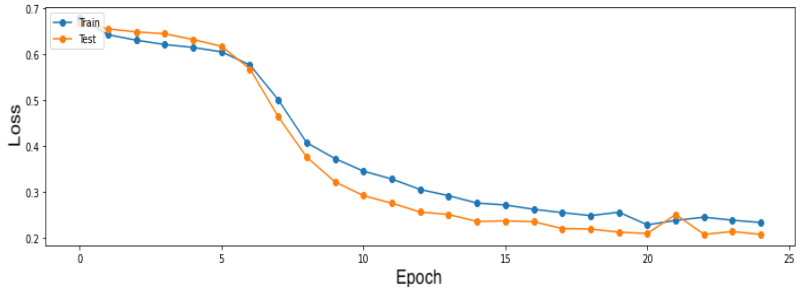
$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{5.1}$$

[1]Precision is the ratio between true positives and the sum of true positives and false positives. Where, a **true positive** is a time series that represents a spike alignment correctly classified by the model; while a **false positive** is a time series that represents a spurious signal above threshold misclassified as an alignment spike by the model.

The precision obtained by the chosen model in the third test is of 92.8% (Figure 5.6a). While, the loss (Figure 5.6b) shows that the model has converged and has low value ($\sim$ 0.2) on the test and train set. As it can



(a) Precision



(b) Loss

Figure 5.6: Precision and Loss of the model trained with BLM signal normalized

be seen from Figure 5.6 the training and validation loss curves decreases to a point of stability with a minimal gap between the two final loss values. This is a sign that a good fit has been found.

5.3 Comparing to Bidirectional LSTM

In this section the proposed LSTM framework described in the previous section is compared with a model that makes use of the Bidirectional LSTM. The model architecture used in this experiment (shown in Figure 5.7 and described in Table 5.2) is composed of:

- One Bi-LSTM layer with 128 hidden neurons;
- Dropout layer with rate of 0.4;
- Dense layer with one neuron and sigmoid activation function;
- Binary crossentropy as loss functions;
- Adam optimizer with a learning rate of 0.0001;
- Early stopping technique with a 'patience' of 3 and 'min' mode.

Train and test have been performed on the dataset used in the third experiment of section 5.2.

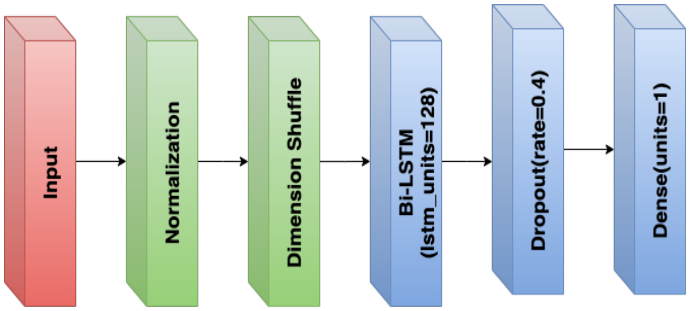
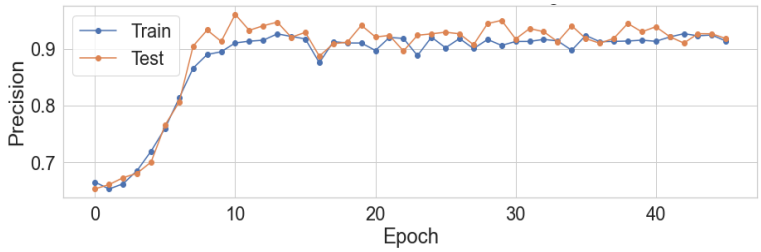
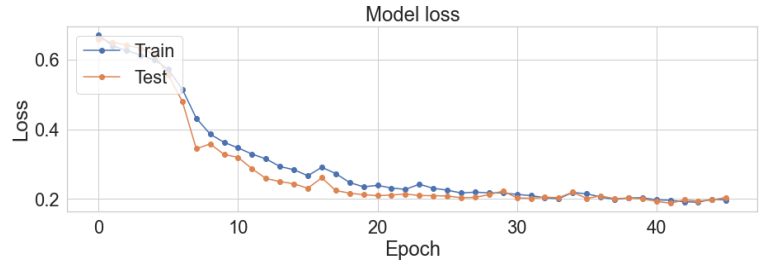


Figure 5.7: Bi-LSTM architecture

The learning curves (Figure 5.8) show that a good fit has been found.



(a) Precision



(b) Loss

Figure 5.8: Precision and Loss of the model trained with Bi-LSTM

Layer	**Output Shape**
Bi-LSTM	(None, 188, 128)
Dropout	(None, 64)
Dense	(None, 1)

Table 5.2: Network Architecture Layers and Parameters

In this experiment the model with one layer of Bidirectional LSTM scored a precision of 91.76%. This value proved to be lower than the one scored by the model without Bi-LSTM layer and will not be used for classification of alignment spikes.

5.4 LSTM model results

In this section the results of the model developed will be presented. The model (described in subsection 5.1.1) trained with BLM signals normalized by position in sigma correctly classified 452/487 time series of BLM signal. The network's performance was evaluated and the average precision obtained for a 80%-20% split dataset is 92.81% (Figure 5.9).

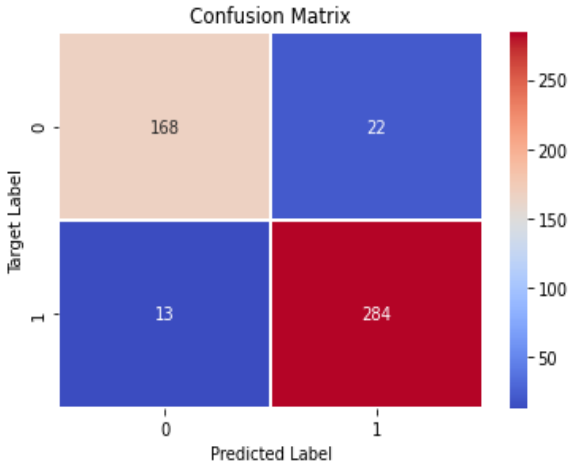


Figure 5.9: Confusion matrix

Figures 5.10 and 5.11 represent two alignment spikes correctly classified by the model. As we can see the classification task begins when the threshold (black line) is exceeded by the BLM signal (blue line), the probability (purple line) reaches the maximum value asymptotically when the decay ends.

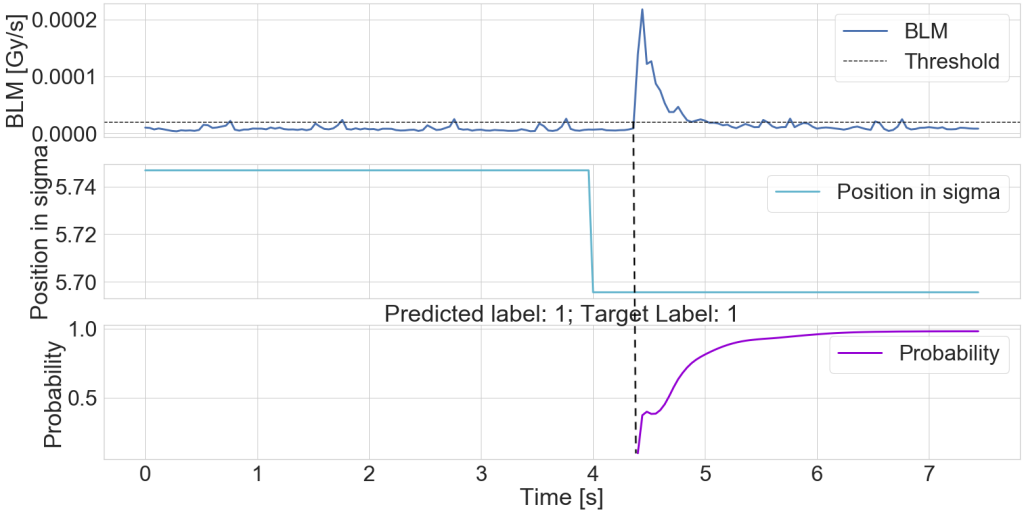


Figure 5.10: Alignment spike correctly classified

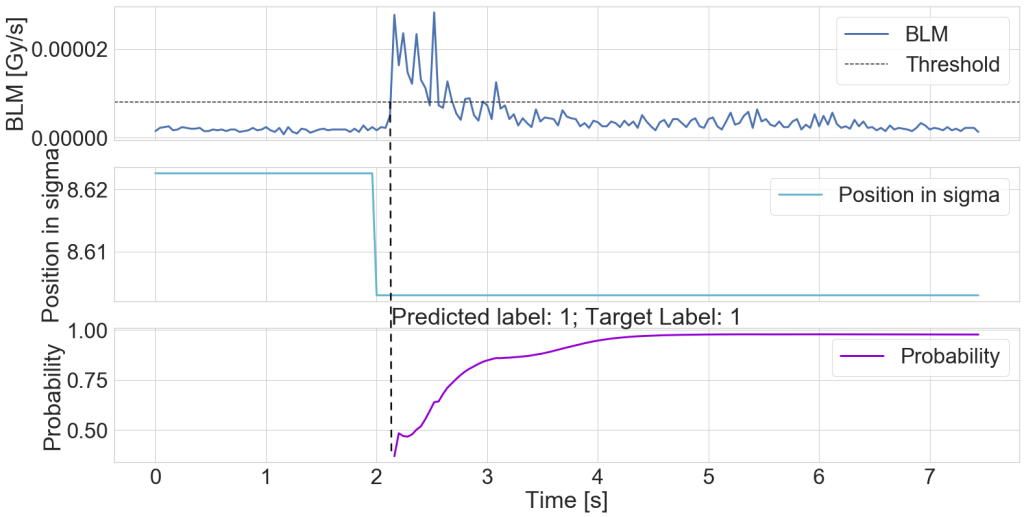


Figure 5.11: Alignment spike correctly classified

In Figures 5.10 and 5.11 are depicted two correctly classified spurious signal exceeding threshold, whereby in each case the probability is always below 50%.

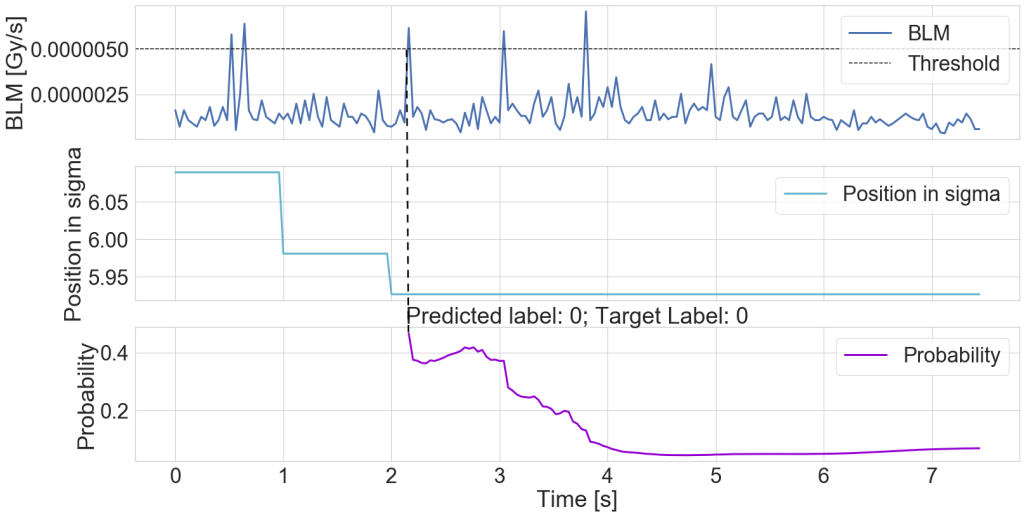


Figure 5.12: Spurious signal correctly classified

As shown in Figures 5.10, 5.11, 5.12 and 5.13, the proposed network shows a smooth transition between the classes (0/1). The network has learned that class transitions are not spontaneous and need time, thus making this model suitable for alignment spike classification.

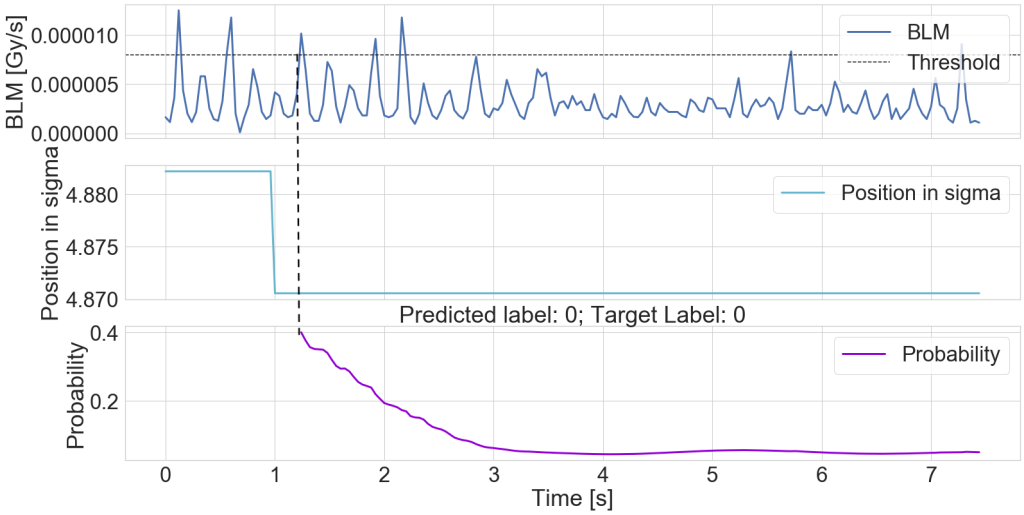


Figure 5.13: Spurious signal correctly classified

5.5 Conclusions

In order to improve the collimator alignment task, we developed a deep learning framework to classify time series. Experimental results show that the model is not only more efficient than the previous one (because the decay constraint set at 4 or 6 seconds described in section 2.5 is no longer used) but also competitive in precision.

Chapter 6

The New Classification Model for the Automatic BBA

When aligning a collimator, the left jaw is first aligned until an alignment spike is observed, followed by the right jaw until an alignment spike is observed. In most cases this is repeated for each jaw to obtain a second alignment spike, to ensure that the collimator has touched the beam. Sometimes, the user, based on their confidence may decide to look only for 1 alignment spike. This chapter describes how the new classification model proposed in chapter 5 can be exploited to further enhance the automatic alignment.

6.1 Model Benchmarking

Through hundreds of observations and with the experience obtained in the development of this model it has been possible to notice that a clear classification of an alignment spike (depicted in Figure 6.1) commonly meets the following requirements:

1. 1 second after the threshold is exceeded the output probability of the model is over 80%;
2. 2 seconds after the threshold is exceeded the probability is over 90%;
3. The probability reaches a maximum value of $\sim$ 97% when the spike decay ends.

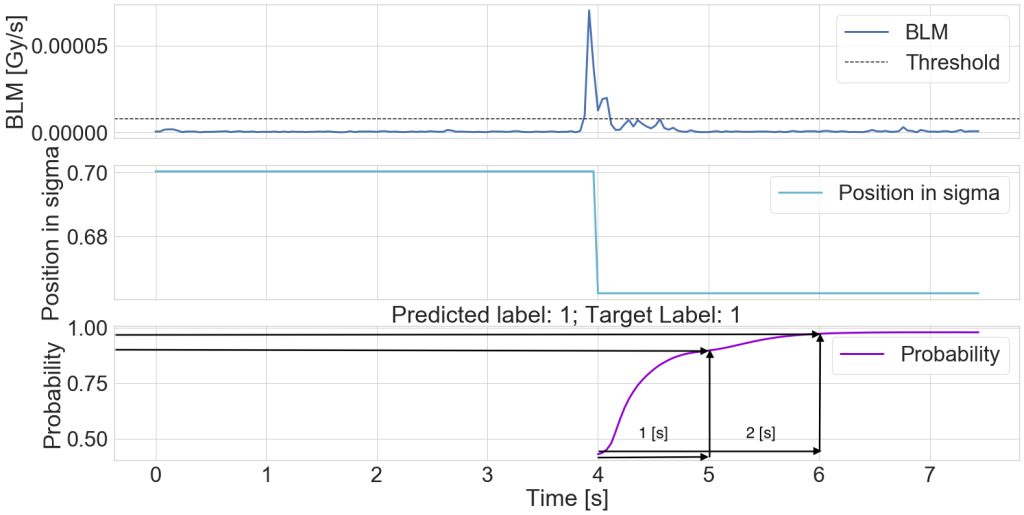


Figure 6.1: Clear Spike Classification

The observation described in item 3 is proven in this work with a suitable analysis using a subset of 1073 time series of alignment spikes, however those in item 1 and item 2 are not included in this thesis due to time constraints. The observation in item 3 is proven by computing the differences between the BLM signal when the LSTM reached the values of probability listed in the x-axis of Figure 6.3 and the average of the last second of signal in which the spike is always decayed. In Figure 6.2 is shown an example of how the difference has been computed.

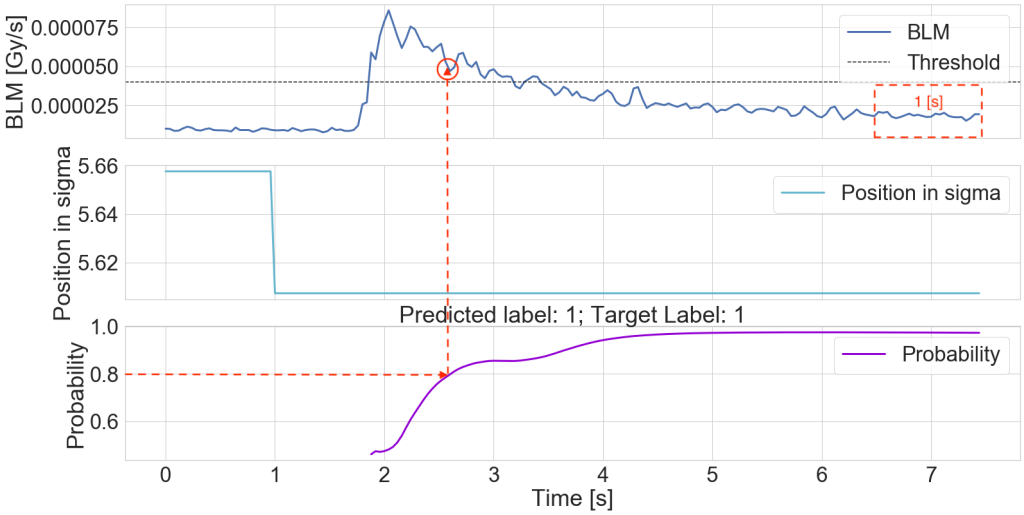


Figure 6.2: Example of computation of the difference between the BLM signal value when the probability is 80% and the average of the last second of time series

As showed in Figure 6.3 the difference is high when the probability is 80% but then it gets smaller as it gets closer to 97%. This analysis proves that when the LSTM probability is over 97% the spike is significantly reduced. One should note that the probability threshold of 97% has been chosen

because it is a perfect balance between BLM signal reduced and confidence that the latter is not misclassified.

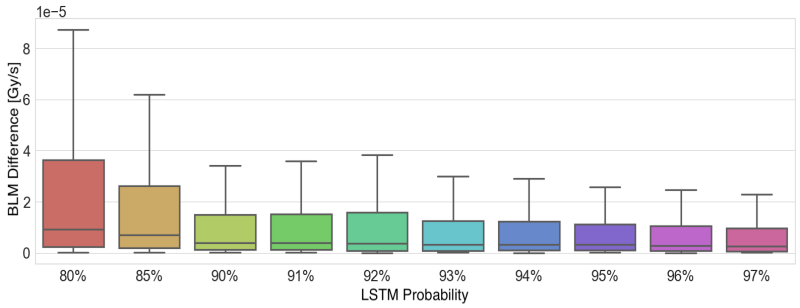


Figure 6.3: Boxplot of differences between the BLM signal value at different probabilities and the average of the steady state after the spike

6.2 Proposed Implementation

A flowchart incorporating the new LSTM model into the fully-automatic BBA is shown in Figure 6.4. The classification algorithm makes use of the current supervised machine learning model for testing purposes. The following bullet list describes its operation:

1. The alignment starts with the movement of the jaw towards the beam until the threshold is exceeded;
2. When the threshold is exceeded a delay of 1 [s] is applied and we verify if the probability outputted by the LSTM is over 80%;
3. Two seconds after the threshold has been exceeded the second verification on the LSTM probability is performed;
4. If the probability is over 90% a counter is incremented and we monitor the probability until it reaches 97%. At the same time we also check if the timer has exceeded 4 or 6 seconds (it depends on the beam state as explained in section 2.5);
5. If one of these conditions is true it means that the decay has completely occurred (as confirmed in the analysis performed in section 6.1) and the counter is incremented again;
6. At this point the supervised learning model developed in [7] is applied and if also it says that the BLM time series is an alignment spike the jaw is classified as aligned.

Variable Name	**Description**
Threshold	BLM threshold needed to stop the jaw movement
Timer	Timer needed to check if the theoretical decay time has elapsed
Probability	LSTM output
Flag	Boolean variable needed to check if the probability is over 80% after 1 s
Counter	Counter needed to check how many alignment spikes occurred
ML 1	Supervised learning models currently in use for spike classification

Table 6.1: Variables used in the flowchart in Figure 6.4

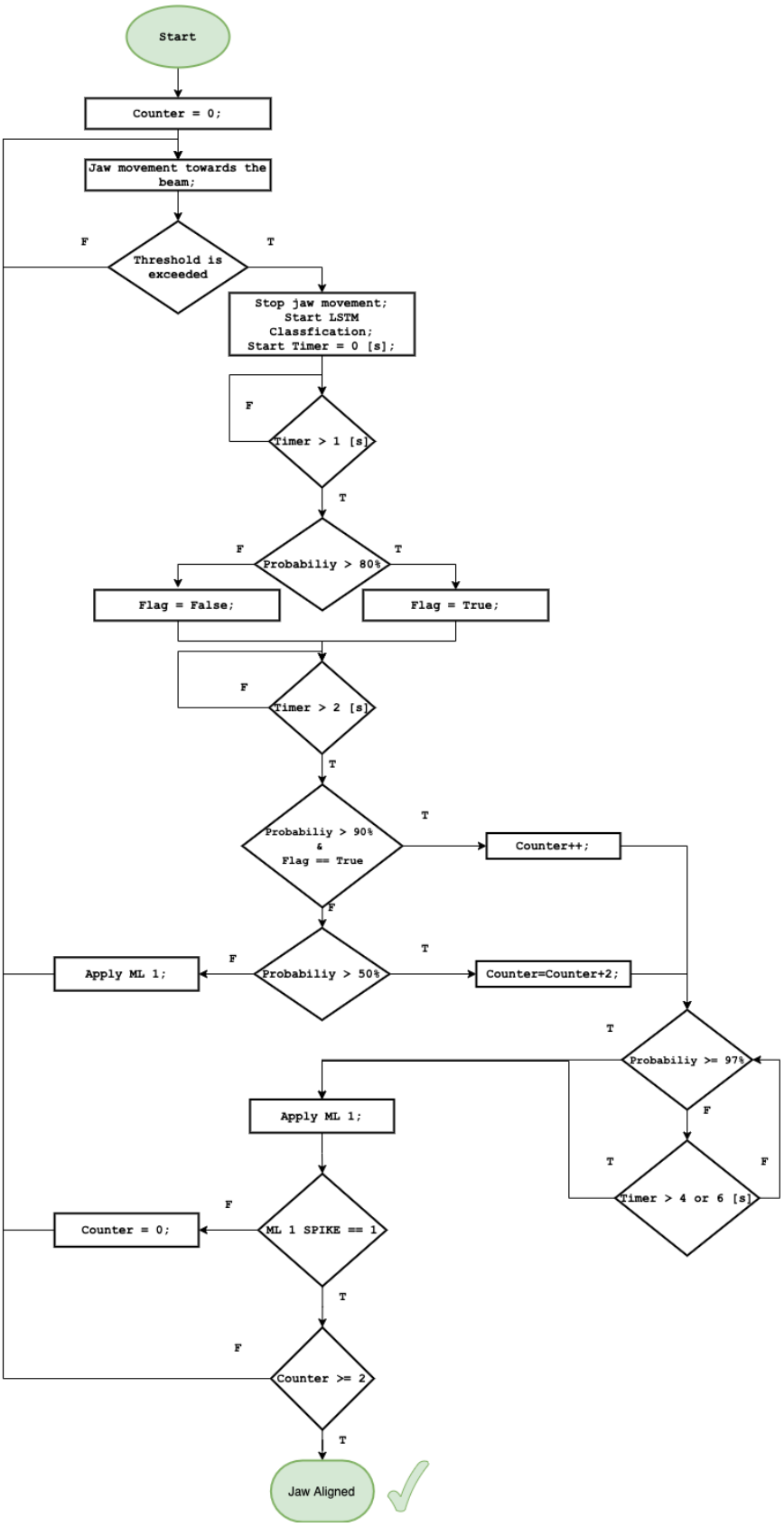


Figure 6.4: Flowchart to align a jaw using new LSTM model

Since false negatives are less grievous (as explained in subsection 5.1.3), and would only bring a small disadvantage in terms of time (because after a misclassification of an alignment spike the alignment continues until a spike is observed); if a time series (after two seconds from the moment that the threshold is exceeded) is classified as spurious signal above threshold by the LSTM the supervised learning model is used for logging purposes only.

The first big innovation of this new classification method is that it offers the possibility to keep the decay time variable while performing classification of the BLM signal in real time. Therefore it is not necessary to wait a fixed amount of time (4 s at injection and 6 s at flat top) to extract features from the exponential fit. Another upgrade in the automatic spike classification algorithm that this implementation offers is the possibility to not need a second spike per jaw if the conditions listed in section 6.1 are satisfied.

6.3 Example of use case

As already mentioned when aligning a collimator first both jaws are moved towards the beam simultaneously, than left jaw is aligned followed by the right jaw and this is repeated to obtain a second alignment spike per jaw. The second spike per jaw is set in the automatic BBA to have an extra safety that the collimator is aligned but sometimes users in the control room can decide that the first spike is enough based on their confidence looking at the data. In this section it will be shown an example of alignment of a collimator in which the additional spikes would not be needed with the new machine learning model. The latter indeed can give feedback of how sure it is that the BLM signal is actually a spike through a probability. This aspect can be used to exploit the LSTM to react like a user would.

An example of alignment of the TCSG.A6R7.B2 collimator is depicted in Figure 6.5, the latter is extracted form LHC logging from an alignment campaign done on 08/04/2018. As we can see from Figures 6.6 and 6.7 the first left and right alignments have been correctly classified by the LSTM and the probability of all spikes meets the requirements of a clear spike explained in section 6.1. For this reason with the new model developed the second alignment spike on both jaws wouldn't have been necessary, saving 12 s on aligning this collimator.

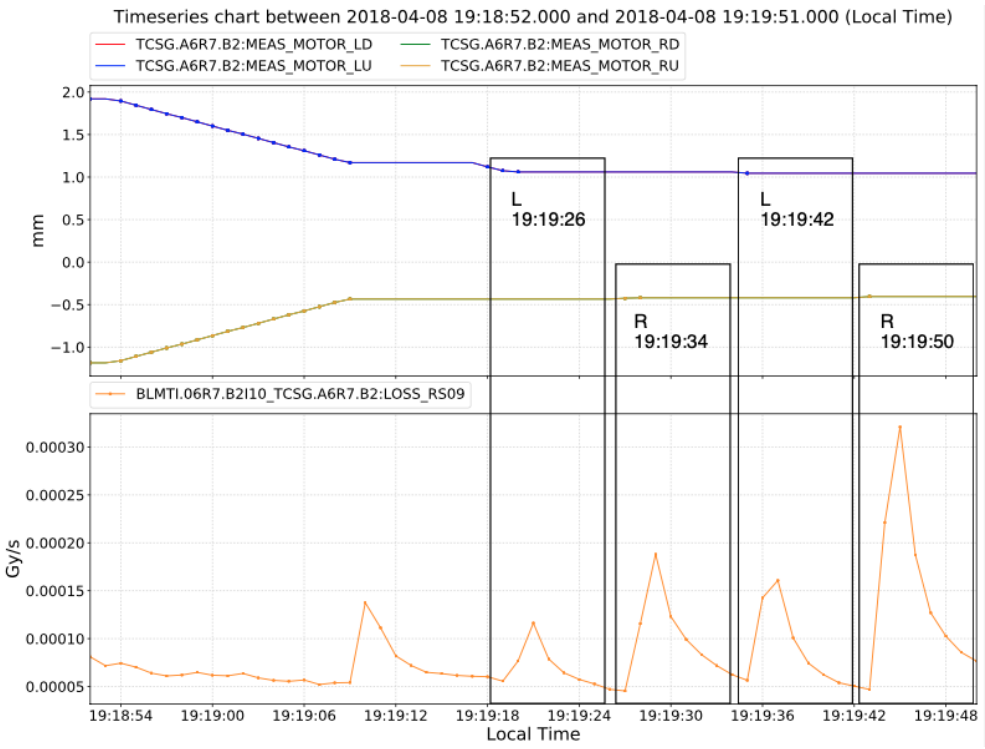


Figure 6.5: Alignment of the TCSG.A6R7.B2 collimator during 2018 commissioning at flat top, adapted from [5].

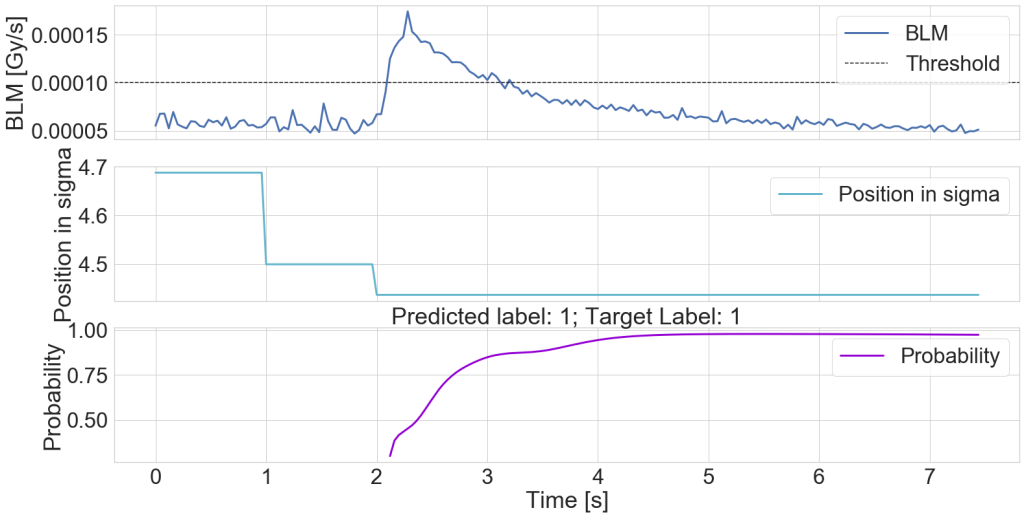


Figure 6.6: Left Jaw First Spike

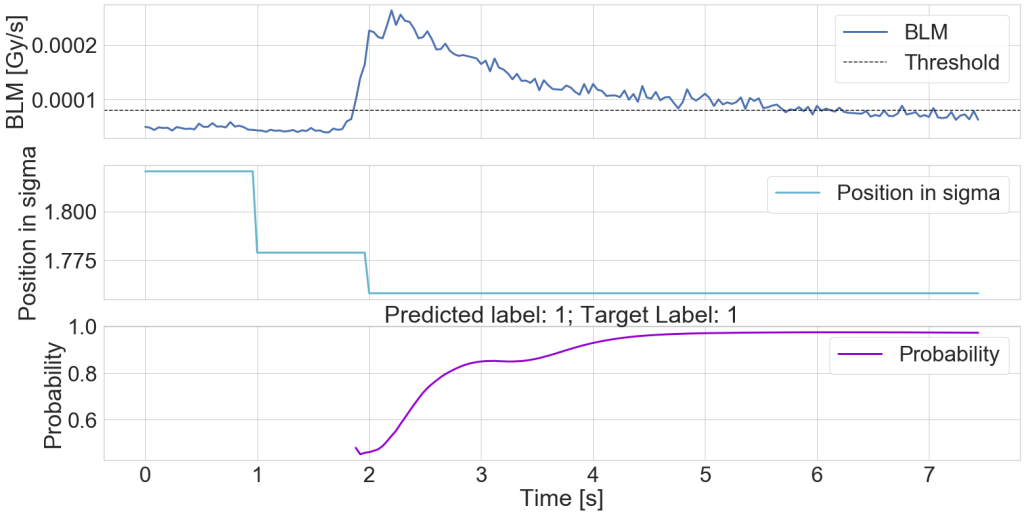


Figure 6.7: Right Jaw First Spike

6.4 Conclusions

As it has been shown, the new classification method is more robust than the previous one because it can classify earlier the signal coming from the beam loss monitors. This means that the theoretical time required to wait that the decay ends (x in Table 2.1), fixed in the previous implementation, can be variable in future upgrades, allowing to classify the spike when the decay is almost over. Thanks to the analysis performed in subsection 4.2.1 in combination with the analysis performed in section 6.1 the average time to classify BLM signals is estimated to be on average 2 s at injection and 4 s at flat top which is close to half the time that was required in the previous implementation. The second innovation of this new method is behind the numbers of alignment required per jaw, that was fixed at 2 but now, based on the observation listed in section 6.1, can be reduced to 1 allowing to save extra time. This could enhance the operational efficiency of the machine by decreasing the time needed to align all collimators in the LHC.

Chapter 7

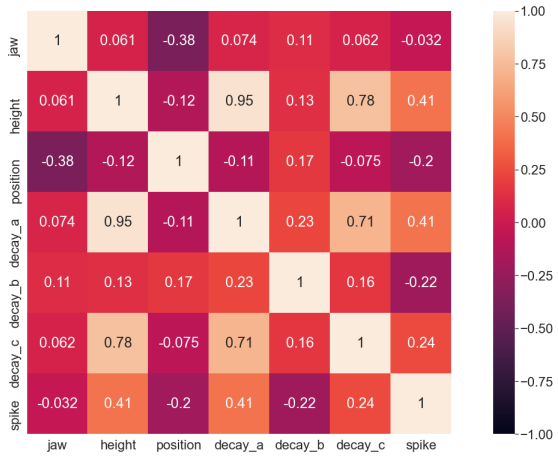
Collimator alignment with machine learning for ion beams

The LHC is designed to collide protons at an energy of 7 TeV per beam as well as heavy-ions at the equivalent magnetic rigidity. While the major hardware systems of the LHC ring appear compatible with heavy ions operation, the beam dynamics and performance limits with ion beams are quite different from those of protons in a number of aspects. This is due to the much higher ionization energy loss of ions compared to protons [32]. Once the ions have fragmented the resulting hadronic shower behaves similarly for both particle species and the heat deposition is proportional to the beam energies. The collimation cleaning efficiency is substantially worst for ions than for protons [33]. The purpose of this chapter is to verify if the current automatic BBA can be used with ion beams and to understand if it is more beneficial to use the new machine learning model described in chapter 5.

The data set used in this section are obtained from alignments performed in 2018 during the collision stage. The data consists of the 25 Hz BLM signals and the 1 Hz collimator jaw positions. The BLM signal in each data sample was individually analyzed and labelled by experts into the two classification classes. The ion dataset used in this work includes a total of 470 samples, 399 classified as alignment spike and 71 classified as spurious signals above threshold.

7.1 Data analysis

This section describes some analyses whose purpose is to understand if the supervised learning models currently used for proton alignment spikes classification can be used also with ions. As explained in section 2.5 the current spike classification system used for protons requires 5 features that can be extracted from a time series of a BLM signal. The first analysis concerned the study of the correlation between these features. The latter was carried out to see how close two variables are to having a linear relationship with each other and also to see how much the alignments made with protons at flat top are similar from those made with ions in collision.



(a) Ions (Collision Stage)

(b) Protons (Flat Top Stage)

Figure 7.1: Heatmap of feature Spearman correlation

As can be seen from Figures 7.1a and 7.1b decay_a, decay_c and spike are correlated with the feature height; the feature position is not correlated with any feature; The feature decay_c is highly correlated with the feature decay_a; The only difference between Figure 7.1a and Figure 7.1b stands in the correlation between the spike class and the features height and decay_a that is higher with protons. This analysis showed that the relationship within the data is the same in the two cases under study.

7.2 Can supervised learning be used to classify ion beam alignment spike?

In this section it will be analyzed further if the ensemble of the six machine learning models can be used to classify ion alignment spikes. This analysis will be performed by comparing the classification results of the supervised learning models logged during the 2018 alignment campaign with the opinion of collimation experts who have reclassified all BLM signals. The summary of this test is listed in Table 7.1 and depicted in Figure 7.2, in which it can be seen that the average precision between the two classes is 0.832%. This analysis confirms that the supervised learning models can be used to classify alignment spikes of ion beams, however the precision is lower than that with proton beams (of 96% section 2.5).

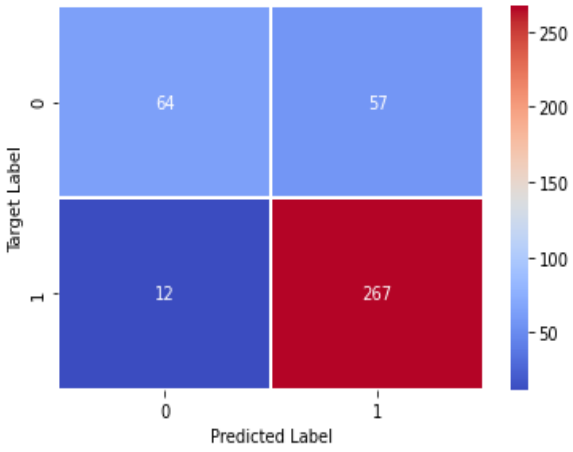


Figure 7.2: Confusion matrix

Table 7.1: Classification report

Label	Precision
0	84.2%
1	82.4%
Average	83.2%

7.3 Can we predict the decay length?

In this analysis will be studied if the decay length of ion alignment spikes can be predicted from other features as it has been done in subsection 4.2.2.

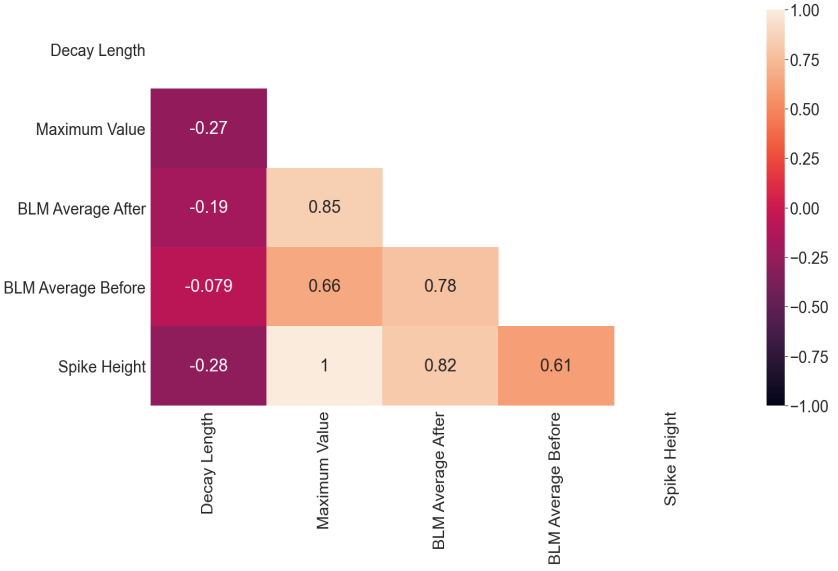


Figure 7.3: Ions (Collision Stage)

Figure 7.3 shows that the decay length of the ion alignment spike is not correlated with any feature. While the features BLM average before the spike, BLM average after the spike and spike height are correlated with the maximum value of the spike. Spike height and mean before the spike are also correlated with the mean after the spike. The summary of this analysis is that the decay length cannot be predicted from others features.

7.4 Can LSTM be used to classify ion beam alignment spike?

In this section will be studied if it is beneficial to use the new deep learning framework presented in chapter 5 to classify alignments spikes of ion beam.

7.4.1 Decay analysis

The second analysis conducted is the decay computation with the use of the half-life and mean-lifetime (Equations 4.5 and 4.8). As the signals were too noisy, a smoothing filter has been applied. This allowed for extracting the exponential fit more easily and then calculate the estimated decay time for each signal. The smoothing is applied at each BLM value after the threshold is exceeded by taking an average of 7 samples of the signal at the time and repeating this for the whole time series analyzed.

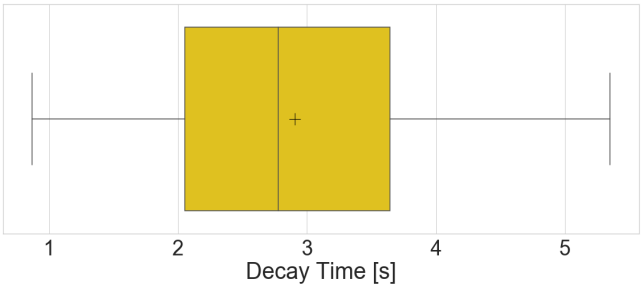
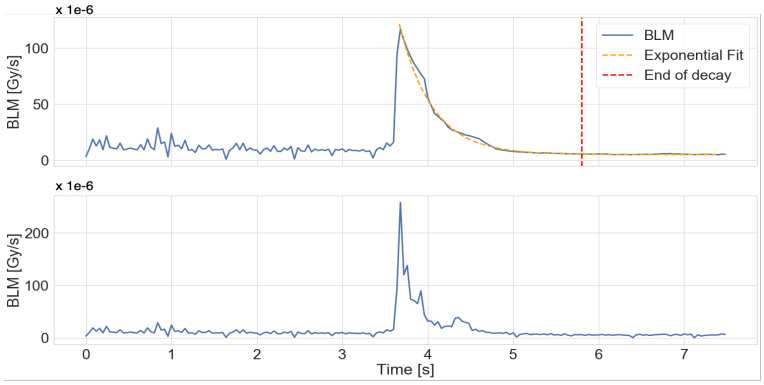
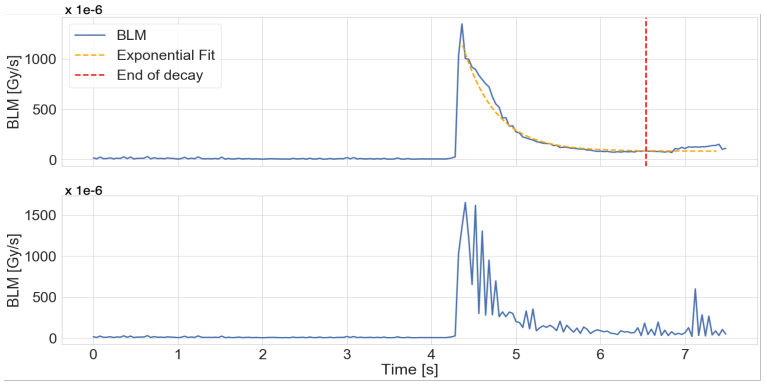


Figure 7.4: Decay time ions



(a) Example of short decay



(b) Example of short decay

Figure 7.5: Examples of two alignment spike with short decay (the upper plot represents the signal with the smoothing filter applied, the lower represents the original BLM signal)

As it can be seen from Figure 7.4 the average decay time for ion alignments spike in collision is 2.90 s ('+' symbol). Moreover from Figure 7.5 is shown that the decay can have variable length. This analysis confirmed that also in this case can be beneficial to use the classification model developed in chapter 5 for the same reason explained in subsection 4.2.1.

7.4.2 LSTM test on Ion Alignments

LSTM classification model developed and trained with BLM time series of proton beams was tested in order to verify if it can be used to classify ion beam spikes. The outcome of this test is depicted in the confusion matrix in Figure 7.6 and listed in Table 7.2, where it can be seen that only 21 time series have been miscassified as false positive. The average precision between the two classes scored by the LSTM on a total of 400 unseen time series is of 85.1% confirming the reliability of the new deep learning framework.

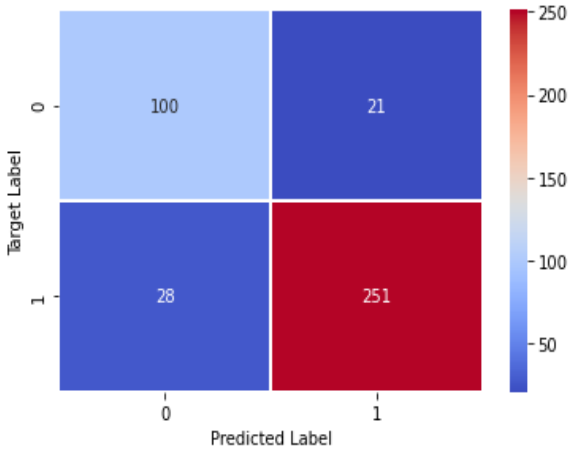


Figure 7.6: Confusion matrix

Table 7.2: Classification report

Label	Precision
0	78.1%
1	92.2%
Average	85.1%

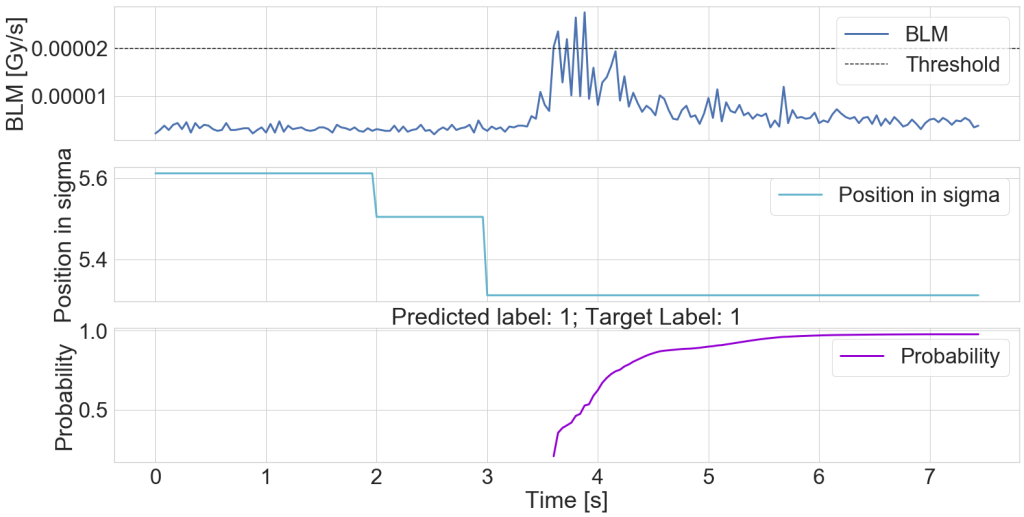


Figure 7.7: Ion alignment spike correctly classified

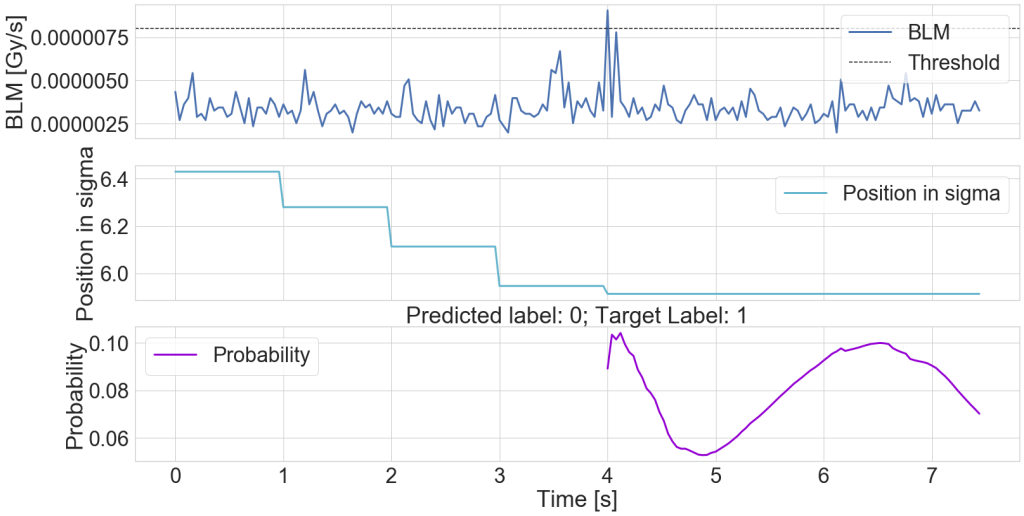


Figure 7.8: Ion spurious signal exceeding threshold correctly classified

In Figures 7.7 and 7.8 are shown respectively an alignment spike and a spurious signal exceeding threshold both correctly classified. In particular, from Figure 7.7 it is possible to notice the high noise of the BLM signal, despite all the LSTM has recognized that is a spike.

7.5 Conclusions

In this chapter two analyses have been conducted. The first covered the test of the classification model currently in use, the second covered the test of the new LSTM model developed. Both tests were done on ion BLM signals labelled by collimators experts. The results of these tests proved that both models can be used to classify ion alignment spikes with a precision loss of about 7% for the LSTM compared to proton time series classifications.

Chapter 8

Conclusions

This thesis presented a research whose main purpose is to speed up the automatic beam-based alignment of the LHC collimators. Collimator alignment is a critical phase of the commissioning because the system includes more than 100 collimators that have to be set around the circulating beams with precision of a few tens of micrometers to form a multi-stage hierarchy which cleans the halo particles in the high-energy LHC beams and protects the machine against beam losses. In particular, aligning a collimator involves automatically classifying, time series. In the present implementation, this process uses machine learning techniques. The time series are signals coming from the Beam Loss Monitors (BLM) installed downstream each collimator. In order to align a collimator its jaws are moved towards the beam until the losses detected by the BLM are not exceeding a predefined threshold and a spike is observed.

A detailed analysis of the data collected in previous alignment campaigns was carried out, this indicating a possible way to improve the alignment algorithms. The initial implementation is based on a fixed time to observe the beam loss signal following the interaction of the collimator jaws with the beam. The analysis consisted in calculating the decay time for each alignment spike with the use of the half-life and showed that shorter observations times could have been used in most of the observed signals.

Following this analysis, a new deep learning framework that makes use of Long Short-Term Memory Recurrent Neural Network for time series classification was proposed. The data set used in this work consists in a total of 2980 BLM time series collected during 2016 and 2018 collimator alignments. The developed model can classify signals of varying lengths, overcoming the constraint to fix an observation time. The LSTM has been tested on BLM signals with proton and ion beams, reaching a precision of 92.8% and 85.1%, respectively. Therefore, this model proved to be reliable to be used during LHC operation.

In addition, an algorithm that incorporates the new deep learning framework into the automatic beam based alignment was developed. The latter classifies beam loss monitors signals in real time from the moment when the BLM threshold is exceeded. This will allow to decrease the time needed to align the LHC collimators while maintaining a very high precision thus

improving the efficiency of the fully-automatic alignment.

In addition, as a part of this thesis work, a new analysis was carried out on previous data collected during the alignment campaigns for heavy-ion beams at the LHC. The currently used supervised learning models have been cross-checked, to study the possibility to apply machine learning approach also in such LHC working point. The ensemble of the classification models obtained a precision of 83.2% and proved that machine learning can be used to classify ion alignment spikes.

When aligning a collimator, in most cases, a second alignment spike is required for each jaw, to ensure that the collimator has touched the beam. In order to further improve the fully-automatic collimator alignment, a possible future development can be to prove with a suitable analysis the observations on the LSTM probability made in Chapter 6. In particular, provide evidence that in a clear alignment spike:

- the LSTM probability is over 80% after one second from the moment when the BLM threshold is exceeded;
- the LSTM probability is over 90% after two seconds from the moment when the BLM threshold is exceeded.

This may reduce the number of spikes per jaw from two to one and it would allow the new classification model to drastically reduce the time needed to align all LHC collimators.

In view of the LHC Run 3 in 2022 a further step is to test the thesis results with beams.

Bibliography

[1] Large Hadron Collider, 2021 (accessed 4 January 2021). *https://en.wikipedia.org/wiki/Large_Hadron_Collider#cite_note-bbc20100330-4.*

[2] Jean-Luc Caron. LHC layout, 1997 (accessed 22 December 2020). *https://cds.cern.ch/record/841573.*

[3] R. W. Aßmann: Requirements for the LHC collimation system. *Technical report, CERN, 2002.*

[4] G. Valentino, R. W. Assmann, R. Bruce, N. Sammut. Classification of LHC Beam Loss Spikes using Support Vector Machines. *IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMI), 2012.*

[5] Courtesy of G. Azzopardi, CERN

[6] Gianluca Valentino, Ralph Aßmann, Roderik Bruce, Stefano Redaelli, Adriana Rossi, Nicholas Sammut, Daniel Wollmann. Semiautomatic beam-based LHC collimator alignment. *Physical Review Special Topics-Accelerators And Beams 15, 051002, 2012.*

[7] G. Azzopardi. Automation of the LHC Collimator Beam-Based Alignment Procedure for Nominal Operation. *PhD Thesis, University of Malta, 2019.*

[8] G. Azzopardi, G. Valentino, B. M. Salvachua Ferrando, S. Redaelli, A. Muscat. Beam Loss Threshold Selection for Automatic LHC Collimator Alignment. *Proceedings of ICALEPCS2019, New York, NY, USA, 2019.*

[9] G. Azzopardi, G. Valentino, A. Muscat, B. M. Salvachua Ferrando. Automatic spike detection in beam loss signals for LHC collimator alignment. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 934, 2019.*

[10] Gianluca Valentino, Ralph Aßmann, Stefano Redaelli, Nicholas Sammut. Simulator for beam-based LHC collimator alignment. *Physical Review Special Topics-Accelerators And Beams 17, 021003, 2014.*

[11] Christopher Olah. Understanding LSTM Networks, 2015 (accessed 3 November, 2020). *https://colah.github.io/posts/2015-08-Understanding-LSTMs/.*

[12] Deep Dive into Bidirectional LSTM, 2019 (accessed 3 November, 2020). *https://www.i2tutorials.com/deep-dive-into-bidirectional-lstm*

[13] Usman Malik. Solving Sequence Problems with LSTM in Keras, (accessed 3 November 2020). *https://stackabuse.com/solving-sequence-problems-with-lstm-in-keras.*

[14] Peilu Wang, Yao Qian, Frank K. Soong, Lei He, Hai Zhao: A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding. *ArXiv, abs/1511.00215, 2015.*

[15] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov: Improving neural networks by preventing co-adaptation of feature detectors. *CoRR, http://arxiv.org/abs/1207.0580, 2012.*

[16] Dense layer, 2020, (accessed 10 January, 2021). *https://keras.io/api/layers/core_layers/dense/dense-layer.*

[17] Deval Shah. What is the Sigmoid Function?, (accessed 10 January, 2021). *https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function.*

[18] Ravindra Parmar. Common Loss functions in machine learning, 2018 (accessed 10 January, 2021). *https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23.*

[19] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization. *Conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.*

[20] Lutz Prechelt. Early Stopping — But When? *Orr, G.B. and Müller, K.-R. (Eds.): LNCS 1524, ISBN 978-3-540-65311-0, 1998.*

[21] Dejan Jovanovic. Deep Learning — Overfitting, 2018 (accessed 10 January 2021). *https://medium.com/newcryptoblock/deep-learning-overfitting-ed677a403147.*

[22] Ramya Vidiyala. Normalization vs Standardization, 2020 (accessed 10 January 2021). *https://towardsdatascience.com/normalization-vs-standardization-cb8fe15082eb.*

[23] Marco Altmann, Peter Ott, Nicolaj C. Stache, Christian Waldschmidt. Learning Dynamic Processes from a Range-Doppler Map Time Series with LSTM Networks. *16th European Radar Conference (EuRAD), 2019.*

[24] Andrew A. Fingelkurts, Alexander A. Fingelkurts and Carlos F.H. Neves. Natural World Physical, Brain Operational, and Mind Phenomenal Space-Time. *Physics of Life Reviews 7(2):195-249, 2010.*

[25] Kusumika Krori Dutta. Multi-class time series classification of EEG signals with Recurrent Neural Networks. *9th International Conference on Cloud Computing, Data Science Engineering (Confluence), 2019.*

[26] McGraw-Hill Encyclopedia of Science & Technology.

[27] Mike Run. File:Exponential-decay-half-life.svg, 2020 (accessed 3 March 2021). *https://commons.wikimedia.org/wiki/File:Exponential-decay-half-life.svg.*

[28] Spearman's Rank-Order Correlation, (accessed 15 January 2021). *https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php.*

[29] Chollet, F. & others, 2015. Keras. Available at: *https://github.com/fchollet/keras.*

[30] Abadi, Martín and Barham, Paul and Chen, Jianmin and Chen, Zhifeng and Davis, Andy and Dean, Jeffrey and Devin, Matthieu and Ghemawat, Sanjay and Irving, Geoffrey and Isard, Michael and others. Tensorflow: A system for large-scale machine learning. *12th Symposium on Operating Systems Design and Implementation, 2016.*

[31] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J. Leon Zhao. Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks. *Web-Age Information Management (WAIM). Lecture Notes in Computer Science, vol 8485. Springer, Cham. 2014.*

[32] N. Fuster-Martínez, R. Bruce, F. Cerutti, R. De Maria, P. Hermes, A. Lechner, A. Mereghetti, J. Molson, S. Redaelli and E. Skordis, *et al.*. Simulations of heavy-ion halo collimation at the CERN Large Hadron Collider: Benchmark with measurements and cleaning performance evaluation. *Phys. Rev. Accel. Beams **23** no.11, 111002doi:10.1103/PhysRevAccelBeams.23.111002 [arXiv:2008.03234 [physics.acc-ph]], 2020.*

[33] Brüning Oliver Sim, Collier Paul, Lebrun P, Myers Stephen, Ostojic Ranko, Poole John, Proudlock Paul: LHC Design Report. *Geneva : CERN, 548 p. 2004.*