Contents lists available at ScienceDirect

Nuclear Inst. and Methods in Physics Research, A

journal homepage: www.elsevier.com/locate/nima

Using LSTM recurrent neural networks for monitoring the LHC superconducting magnets

Maciej Wielgosz [a,*], Andrzej Skoczeń [b,c], Matej Mertik [d]

[a] Faculty of Computer Science, Electronics and Telecommunications, AGH University of Science and Technology, Kraków, Poland
[b] Faculty of Physics and Applied Computer Science, AGH University of Science and Technology, Kraków, Poland
[c] The European Organization for Nuclear Research — CERN, CH-1211 Geneva 23, Switzerland
[d] Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia

A R T I C L E I N F O

Keywords:
LHC
Recurrent neural networks
LSTM
Deep learning
Modeling

A B S T R A C T

The superconducting LHC magnets are coupled with an electronic monitoring system which records and analyzes voltage time series reflecting their performance. A currently used system is based on a range of preprogrammed triggers which launches protection procedures when a misbehavior of the magnets is detected. All the procedures used in the protection equipment were designed and implemented according to known working scenarios of the system and are updated and monitored by human operators.

This paper proposes a novel approach to monitoring and fault protection of the Large Hadron Collider (LHC) superconducting magnets which employs state-of-the-art Deep Learning algorithms. Consequently, the authors of the paper decided to examine the performance of LSTM recurrent neural networks for modeling of voltage time series of the magnets. In order to address this challenging task different network architectures and hyper-parameters were used to achieve the best possible performance of the solution. The regression results were measured in terms of RMSE for different number of future steps and history length taken into account for the prediction. The best result of RMSE = 0.00104 was obtained for a network of 128 LSTM cells within the internal layer and 16 steps history buffer.

1. Introduction

The Large Hadron Collider (LHC) is the largest and the most powerful particle collider ever built. It was designed and constructed as a joint effort of the international scientific collaboration of the European Organization for Nuclear Research (CERN) [1,2]. The whole architecture of the LHC is unique and most of its components were custom manufactured specifically for this particular application. Consequently, malfunctions and failures of the components usually result in long and costly repairs. This, in turn, affects the availability of the particle beams for physics experiments carried out at the LHC. Therefore, maintenance and faults prevention is critical and dedicated solution named Machine Protection System (MPS) was created. The MPS system comprises many subsystems, including beam and equipment monitoring, a system to safely stop beam operation and an interlock system providing the glue between these systems. The goal is to ensure a safe operation to the accelerator and to maximize a time when particle beams are delivered to interaction points.

One of the most crucial components of the LHC is a set of superconducting magnets which keep the bunches of protons in a right trajectory inside the vacuum beam pipes in the 27 km long accelerator tunnel [1,2]. A voltage on each of the superconducting magnets in the LHC is measured by dedicated digital voltmeter [3] and sent to the central database. The generated stream of the voltage data is used to monitor performance and detect anomalies in the behavior of superconducting elements.

One of the most dangerous phenomenon, which can take place at any time in a superconducting electrical circuit, is a quench. It occurs when a part of the superconducting cable becomes normally-conducting [4]. The quench may happen at any time randomly and may occur for many reasons. Usually, it is due to a mechanical event inside a superconducting cable or coil, related to the release of stresses generated during production, transportation, and assembly of a magnet. Another phenomenon which may lead to a quench is a deposition of energy of particles which escaped from the beam (so called beam losses). When

[*] Corresponding author.
E-mail addresses: wielgosz@agh.edu.pl (M. Wielgosz), skoczen@fis.agh.edu.pl (A. Skoczeń), matej.mertik@um.si (M. Mertik).

http://dx.doi.org/10.1016/j.nima.2017.06.020
Received 21 March 2017; Received in revised form 3 June 2017; Accepted 13 June 2017
Available online 17 June 2017

the Quench Protection System (QPS) detects an increased resistance, the huge amount of energy stored in the magnet chain is extracted and dumped into a specially designed resistor.

Currently the QPS is the highly dependable system specifically designed for the LHC. The instruments of this system perform acquisition of total voltage across superconducting elements (magnet coils, bus bars, current leads) and extract resistive component of this voltage. The system [3,5] requires a number of settings. Two the most important settings are:

- resistive voltage threshold at which actuators are triggered when the quench event occurs,
- discrimination time by which the threshold must be exceeded to recognize the quench event.

The values of these parameters are chosen based on a prior analysis of the magnets and the power supply behavior. The current approach is very useful and has proven its high effectiveness to successfully protect the LHC against severe consequences of the quench events.

The purpose of this article is to present an approach of modeling the resistive voltage of the LHC superconducting magnets by means of using Recurrent Neural Networks (RNN). The model comprises also such severe phenomena as quenches. The existence of so called quench precursors was already observed in voltage signal acquired from superconducting coil [6,7]. In the authors' opinion, it is likely that a part of the quench causes may gradually unfold in time which ultimately leads the magnet to leave its operating point. If this is the case, they can be modeled and predicted using RNN.

Data for training and testing was taken from logging database [8]. The low time resolution of this data severely limits the possibility to infer a thesis about the effectiveness of quench prediction. However, the quench data is only used as an example of anomalies occurring during hardware operation - a case study, for which the data was available. The article demonstrates that even for very low resolution data (one sample for 400 ms) the proposed neural network structures could model behavior of the magnets.

At so early stage of investigation, there is no chance to answer how fast the network can generate a response. The implementation of neural network in FPGA or ASIC is necessary in order to check a timing in comparison to a quench time scale. However, a literature review was conducted in order to estimate the possible system reaction time.

The article contains the following four main contributions:

- analysis of LSTM applicability to modeling voltage time series of LHC superconducting magnets,
- experimental verification of a range of the LSTM models with real LHC data,
- development of a procedure for data extraction and the model training and testing,
- development of a custom designed prototype application of LSTM-based model for conducting experiments.

The rest of the article is structured as follows. Section 2 provides the background and related work of the LHC, superconducting magnets and quench protection. Section 3 contains theory and mathematical formalism for recurrent neural networks. Section 4 presents an idea of a visualization environment for the results of the experiments. Section 5 describes the architecture of the custom designed system used for the experiments as well as for data acquisition and provides the results of the experiments. Section 6 provides a discussion about possible uses of the proposed solution, as well as the performance of comparable systems. Finally, the conclusions of our research are presented in Section 7.

2. Large Hadron Collider

Large Hadron Collider (LHC) is currently the most powerful scientific instrument ever built. The main objective of this huge enterprise is a pure desire for exploring the deepest structure of matter. The project was launched in 1994 and it is managed by the European Organization for Nuclear Research (CERN). After overcoming many technical challenges, the LHC started the operation in 2010 [2]. Currently the second run of the LHC is in progress.

Table 1
The nominal conditions in the main dipole circuits of the LHC at the beginning and at the end of ramping up [2].

Parameter	Injection	Collision	Unit
Proton energy	0.450	7	TeV
Magnetic field	0.535	8.33	T
Supply current	763	11 850	A

One of the ideas used in high energy physics experiments is essentially based on a concept of colliding two high energy particles which travel in opposite directions. This allows to look deeply into the structure of the matter which constitutes our universe. The particles used in case of the LHC are protons or lead ions. The products of protons collision are carefully analyzed by huge systems of particle detectors. There are four main independent detection systems built at the LHC, namely: ATLAS, CMS, LHCb and ALICE.

The main goal of the LHC is confirmation or refutation of the theories in the field of elementary particle physics. One of the most crucial questions which LHC was intended to address is related to the existence of the postulated Higgs boson which was ultimately discovered in the course of the experiments conducted independently by two collaborations at CERN in 2012 [9,10].

For the most part, the LHC is located in the tunnel lying between 45 and 170 m below the earth's surface near Geneva lake. The tunnel is of a circular shape with the circumference of approx. 27 km. There are many superconducting magnets located in the tunnel which provide a magnetic field necessary to lead the proton beams around a circular trajectory. The LHC tunnel is divided into eight different sectors. The particles are injected into the LHC with energy of 450 GeV. They are prepared within the smaller accelerator (called SPS) and injected into the LHC in bunches. A single bunch contains nominally 1.15×10^{11} protons. The operation of gradual delivery of proton bunches to the LHC is denoted as "filling the machine". It takes 2808 bunches altogether to fill up the LHC. The time between bunches is 25 ns.

It is worth noting that all the bunches traveling along the LHC circle are accelerated in one dedicated place. The remaining sections of the circle guide particles to the accelerating cavities and during each revolution the energy of particles is raised. In order to maintain a stable trajectory of the particles the uniform dipole magnetic field has to be raised synchronously with the raising particle energy. This in turn results in a ramp up of a current in the superconducting dipoles. The described process, denoted as "ramping up the machine", allows to achieve a particle energy of 7 TeV after multiple iterations. Table 1 shows initial and final levels of proton's energy, magnetic field and current supply.

When desired energy is achieved, the beams collide at four points around the circle which are surrounded by the four detection systems. There is a huge amount of data produced by the detectors since every 25 ns two bunches collide giving a number of individual proton–proton collisions. The tracks of particles produced in each individual collision are recorded by detection system. The data gathered by the system is processed by a reconstruction algorithm.

2.1. *Superconducting magnets*

The superconducting magnets are the critical components of the LHC which store huge amount of magnetic energy. This imposes a series of challenges related to powering the whole system. The cables used to wind the magnet coils and to deliver a current (bus bars, current leads) to the coils must conduct the current at the level of ≈12 000 A in the magnetic field of ≈8.5 T (Table 1). Consequently, the designers

Table 2
The general overview of the circuits powering the superconducting magnets of the LHC [12,13]. The number of quenches is reported on 10 October 2016.

LHC Circuit	No of circuits	No of magnets in one circuit	No of quenches
RB	8	154	1270
RQ	16	47	64
IT	8	4	18
IPQ	78	2	323
IPD	16	1	53
600 A EE	202	*m*	
600 A EEc	136	1 or 2	425
600 A	72	1	
80÷120 A	284	1	116
60 A	752	1	44

RB — Main Dipole; RQ — Main Quadrupole; IT — Inner Triplet; IPQ — Individually Powered Quadrupole; IPD — Individually Powered Dipole; EE — Energy Extraction; EEc — Energy Extraction by crowbar; *m* — number of magnets in circuits is not constant in this class of circuits.

decided to take advantage of superconducting materials which meet all the electrical and magnetic requirements of the LHC. The instruments built with superconductors are also small enough to fit in the tunnel.

The superconducting cables are not cryostable and therefore a random and local temperature change can lead to a sudden transition to a normal conduction state [4]. This phenomenon is known as quench. During assembly and operation, a superconducting coil is always subjected to stresses resulting from pre-loading at assembly, from differential thermal contractions at cool-down and from the electromagnetic forces during its regular operation. The release of the mechanical energy happens locally, through micro-slips constrained by friction, vibration, or local cracking. The amount of energy generated within the process can be enough to elevate temperature locally above a critical value. Consequently, the resistive place in the cable generates enough heat to damage the cable.

Beam losses are a very important problem for superconducting accelerators. Protons which escape from a bunch in the direction perpendicular to the beam hit the wall of the vacuum beam pipe. Cascades of particles are produced and a radiated energy is deposited in the surrounding materials, in particular in the superconducting windings. This energy can locally heat the coil above critical temperature, causing a quench [11].

Quenches may occur in various circumstances but some of the most common ones take place during a so-called magnet training. At the first powering during ramping up a current, a magnet losses superconducting state long before reaching the expected critical current. At the next attempt of powering, the current that could be reached before quench is higher. The process continues over all the next attempts, and the maximum current that could be reached increases quench after quench, slowly approaching a plateau.

A circular particle accelerator requires a dipole magnetic field to maintain the particle beam within its trajectory. Furthermore, several other kinds of magnets are required for shaping and guiding the beam. In the case of the LHC, most of them are superconducting magnets supplied with constant current by means of power converters. A summary of superconducting circuits is presented in Table 2.

2.2. Quench protection

A need for a system of an active magnet protection originates from the nature of the superconducting cables used to build the magnets. Most of the high-current superconducting magnets used in the LHC are not self-protected and would be damaged or destroyed if they were not protected during the quench. Therefore, the quench protection system was introduced [3,5]. The LHC Machine Protection System (MPS) comprises many subsystems. One of the subsystems is a Quench Protection System (QPS). This system consists of a Quench Detection System (QDS) and actuators which are activated once a quench is detected.

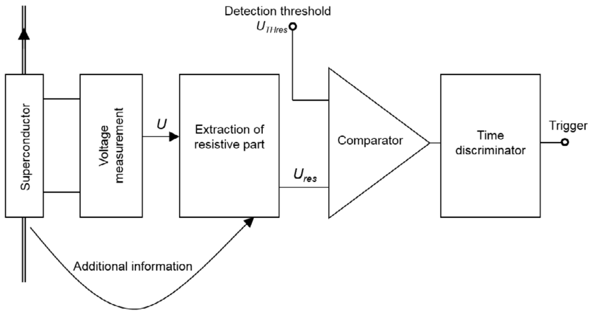

Fig. 1. The general architecture of the quench detector.

A superconducting magnet has zero resistance and a relatively large inductance which is equal ≈100 mH in the case of main LHC dipole. When a constant current flows through the magnet, the total voltage across it, is zero. When the magnet loses its superconducting state (quench) the resistance becomes non-zero, hence, a voltage develops over the resistive part. This voltage is used to detect the quench. However, during normal operation (ramp up or down, fast power abort) a current change in the magnet generates an inductive voltage which might be well above the resistive voltage detection threshold. Therefore, the inductive voltage must be compensated in order to prevent the QDS from spurious triggering. Consequently, the most important part of the quench detector is an electronic module for extracting the resistive part of the total voltage. It is shown in Fig. 1.

A quench detector is an electronic device with the following functions (Fig. 1):

- Monitoring of the voltage of superconducting elements,
- Extraction of the resistive part of the voltage U_{res},
- Generation of trigger signals in case the resistive voltage exceeds the threshold.

The triggers are transmitted to other protection devices via current loops to initiate a safe shutdown of the electric circuits supplying the magnets.

The method for compensation of the inductive voltage is simple in the case of a differential magnet circuit where two very similar inductances are connected in series in one circuit. However, in some LHC corrector magnet circuits, there are no reference elements available, hence the compensation of the inductive voltage by simple subtraction cannot be implemented. In such a case, Kirchhoff's voltage law for the circuit must be solved. To satisfy timing requirements, the solution must be performed numerically (online) by means of using digital logic.

Next, a quench candidate is validated as a real quench or noise. This is carried out by means of a time discriminator shown in Fig. 1. The voltage resistive component U_{res} must be higher than a threshold for the time interval longer than a validation time Δt_{val} in order to be classified as a quench. This condition is depicted in Fig. 2.

The trigger signal has two important functions: the release of energy to quench heaters and an opening for an interlock loop. The goal of the quench heater is an acceleration of the propagation of the quench along a cable. It prevents local overheating (or even melting) of the quenching cable. The opening of the interlock loop is a method for an immediate transferring of the request for the termination of an operation of other LHC components.

Voltage time series measured and extracted by the QPS system are sent over to two different storage systems. The system called the CERN Accelerator Logging Service (CALS) contains low resolution data [8]. The second system, called POST_MORTEM, is dedicated to store data delivered by any equipment in the LHC whenever a trigger occurs [14].

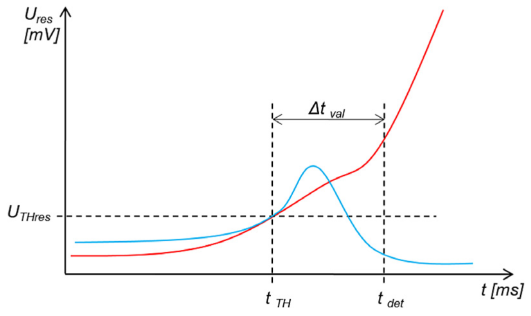

Fig. 2. The principle of a quench validation by means of using a time discriminator.

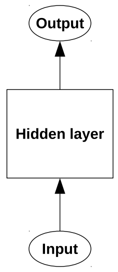

Fig. 3. The architecture of standard feed-forward neural network.

3. Recurrent neural networks

Recent years have witnessed a huge growth of deep learning applications and algorithms. They are powerful learning models, which achieve great successes in many fields and win multiple competitions [15]. The neural nets are capable of capturing latent content of the modeled objects in large hierarchies [16–19]. Two main branches of the neural networks are feed-forward and recurrent models. The members of the first one, of which Convolutional Neural Networks (CNN) is now the most prominent example, are usually used for processing data belonging to a spatial domain, where data occurrence in time is not important and not taken into account [16,20,21]. Opposed to that there are algorithms working in temporal domain, in which the information about the order of data is critical.

Since magnets behavior modeling involves temporal dependencies of the examined signals we decided to focus on the neural network models that are capable of sequence processing, namely RNN, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) [17,22–24].

Currently, the LSTM is considered to be the best model [25] and it is also the most often used in applications. Therefore, we have decided to use it in our experiments.

Unlike traditional models that are mostly based on hand-crafted features deep learning neural networks can operate directly on raw data. This makes them especially useful in applications where extracting features is very hard and even sometimes impossible. It turns out that there are many fields of applications where no experts exist who can handle feature extraction or the area is simply uncharted and we do not know whether the data contains latent patterns worth exploring [26–28].

Foundations of the most neural network architectures currently used were laid down between 1950 and 1990. For almost the last two decades researches were not able to take full advantage of these powerful models. But the whole machine learning landscape changed in early 2010, when deep learning algorithms started to achieve state-of-the-art results in a wide range of learning tasks. The breakthrough was brought about by several factors, among which computing power, huge amount of widely available data and affordable storage are considered to be the critical ones. It is worth noting that in the presence of large amount of data, the conventional linear models tend to under-fit or under-utilize computing resources.

CNNs and feed-forward networks rely on the assumption of the independence of data within training and testing set as presented in Fig. 3. This means that after each training item is presented to the model, the current state of the network is lost i.e. temporal information is not taken into account in training a model.

In the case of independent data, it is not an issue. But for data which contain crucial time or space relationships, it may lead to the loss of the majority of the information which is located in between steps. Additionally, feed-forward models expect a fixed length of training vectors which is not always the case, especially when dealing with time domain data.

Recurrent neural networks (RNNs) are models with the ability to process sequential data one element at a time. Thus they can simultaneously model sequential and time dependencies on multiple scales. Unfortunately, a range of practical applications of standard RNN architectures is quite limited. This is caused by the influence of a given input on hidden and output layers during the training of the network. It either decays or blows up exponentially as it moves across recurrent connections. This effect is described as the *vanishing or exploding gradient problem* [17]. There had been many unsuccessful attempts to address this problem before LSTM was eventually introduced by Hochreiter and Schmidhuber [25] which ultimately solved it.

Recurrent neural networks may be visualized as looped-back architectures of interconnected neurons. This was presented in Fig. 4. Originally RNNs were meant to be used with single variable signals but they have also been adapted for multiple stream inputs [17].

It is a common practice to use feed-forward network on top of recurrent layers together in order to map outputs from RNN or LSTM to the result space as presented in Fig. 5.

3.1. RNN

Architecture of standard neural networks is presented in Fig. 6. The nodes of the network receive input from the current data point $x^{(t)}$ as well as the hidden state values of the hidden layer in the previous state $h^{(t-1)}$. Thus, inputs at time t have impact on the outputs of the network to come in the future by the recurrent connections.

There are two fundamental equations (1) and (2), which characterize computations of a recurrent neural network as presented in Fig. 6.

$$h(t) = Q(W_{(hx)}x^{(t)} + W_{(hh)}h^{(t-1)} + b_h), \tag{1}$$

$$\hat{y}^{(t)} = \sigma(W_{(yh)}h^{(t)} + b_y) \tag{2}$$

where: Q is an activation function. $W_{(hx)}$, $W_{(yh)}$ and $W_{(hh)}$ are weights matrices of input-hidden layer, hidden-output layer and recurrent connections respectively. b_h and b_y are vectors of biases.

Standard neural networks are trained across multiple time steps using the algorithm called *backpropagation through time* [29].

3.2. LSTM

In practical applications, the Long Short-Term Memory (LSTM) model has shown extraordinary ability to learn long-range dependencies as compared to standard RNNs. Therefore, most of state-of-the-art applications use the LSTM model [29].

The LSTM internal structure is based on a set of connected cells. The structure of a cell is presented in Fig. 7, it contains feedback connection storing the temporal state of the cell. Additionally, the LSTM cell contains three gates and two nodes which serve as an interface for information propagation within the network.

There are three different gates in each LSTM cell:

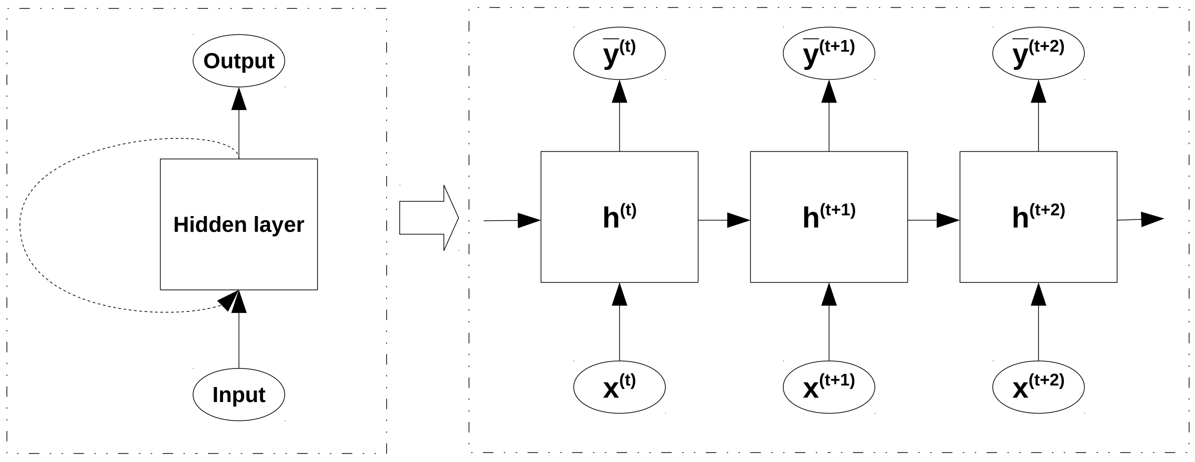

Fig. 4. The general overview of recurrent neural networks.

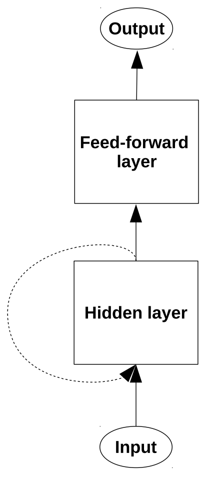

Fig. 5. The mapping of outputs of a recurrent neural network.

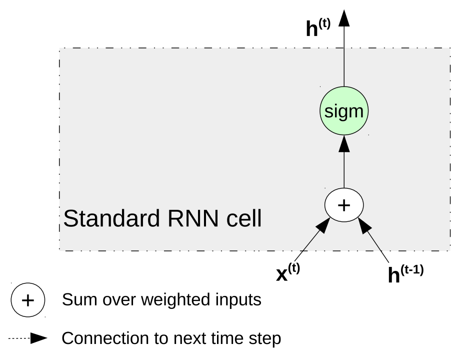

Fig. 6. A cell of a standard recurrent neural network.

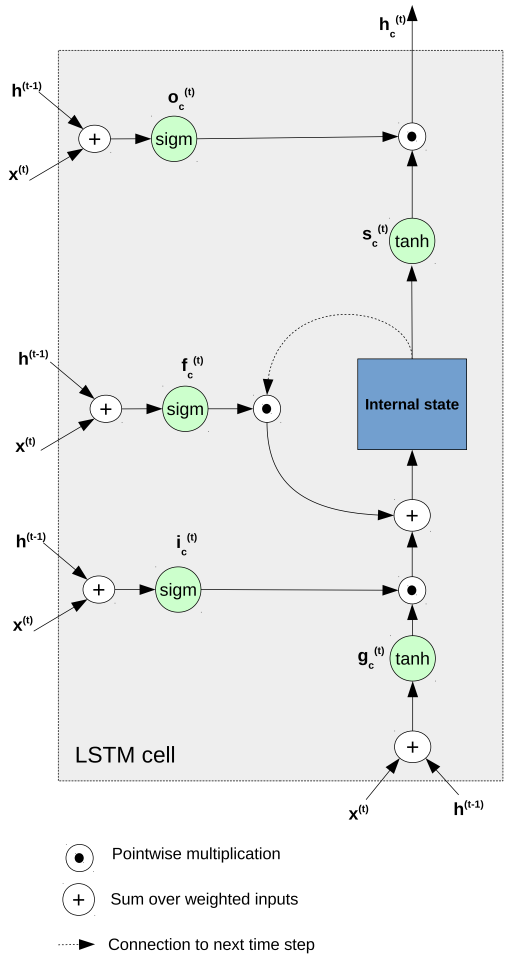

Fig. 7. An architecture of the LSTM cell.

- *input* gate $i_c^{(t)}$ which controls input activations into the memory element,
- *output* gate $o_c^{(t)}$ controls cell outflow of activations into the rest of the network,
- *forget* gate $f_c^{(t)}$ scales the internal state of the cell before summing it with the input through the self-recurrent connection of the cell. This enables gradual forgetting in the cell memory.

In addition, the LSTM cell also comprises an input node $g_c^{(t)}$ and an internal state node $s_c^{(t)}$.

Modern LSTM architectures may also contain *peephole connections* [30]. Since they are not used in the experiment, they were neither depicted in Fig. 7 nor addressed in this description.

The output of a set of LSTM cells is calculated according to the following set of vector equations:

$$g^{(t)} = \phi(W_{gx}x^{(t)} + W_{gh}h^{(t-1)} + b_g), \quad (3)$$

$$i^{(t)} = \sigma(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + b_i), \tag{4}$$

$$f^{(t)} = \sigma(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + b_f), \tag{5}$$

$$o^{(t)} = \sigma(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + b_o) \tag{6}$$

$$s^{(t)} = g^{(t)} \odot i^{(t)} + s^{(t-1)} \odot f^{(t)}, \tag{7}$$

$$h^{(t)} = \phi(s^{(t)}) \odot o^{(t)}. \tag{8}$$

While examining Eqs. (3)–(8), it may be noticed that instances for a current and previous time step are used for the value of the output vector of hidden layer h as well as for the internal state vector s. Consequently, $h^{(t)}$ denotes a value of an output vector at the current time step, where as $h^{(t-1)}$ refers to the previous step. It is also worth noting that the equations contain vector notation which means that they address the whole set of LSTM cells. In order to address a single cell a subscript c is used as it is presented in Fig. 7, where for instance $h_c^{(t)}$ refers to a scalar value of an output of this particular cell.

The LSTM network learns when to let an activation into the internal states of its cells and when to let an activation of the outputs. In this gating mechanism all the gates are considered as separate components of the LSTM cell with their own learning capability. This means that the cells adapt during training process to preserve a proper information flow throughout the network as separate units. Therefore, when the gates are closed, the internal cell state is not affected. In order to make this possible a hard sigmoid function σ was used, which can output 0 and 1 as given by Eq. (9). As a result the gates can be fully opened or fully closed.

$$\sigma(x) = \begin{cases} 0 \text{ if } x \le t_l, \\ ax + b \text{ if } x \in (t_l, t_h), \\ 1 \text{ if } x \ge t_h. \end{cases} \tag{9}$$

In terms of the backward pass, so-called constant error carousel enables the gradient to propagate back through many time steps [25,29].

4. Visualization framework

The model is intended to be integrated within visualization environment for the experiments. Python framework based on Django (storing and managing experiments setup data) [31] and Bokeh (interactive Python library for visualization of data) [32] will be used for the development of web application for quench prediction. Described LSTM model will be integrated to building blocks of an ELQA data analysis framework [33] developed at Machine Protection and Electrical Integrity group (TE-MPE) in order to prototype a web based quench prediction application for use at CERN.

In the ELQA framework, an access to the data is addressed with the Object-Relational Mapping. The Django framework handles this mapping and provides full functionality of the Structure Query Language (SQL). The architecture of the Django is organized with three layers as follows:

- the bottom layer which is a database, followed by
- an access library that is responsible for a communication between Python and the database by means of SQL statements and
- a specific Django database back-end.

As the access library to CERN Oracle from Django a Python library `cx_oracle` [34] is used. For results capturing and maintaining appropriate data model is defined, which can be created by means of the tool called `inspectdb` available inside Django. Information necessary for this process is taken from database tables, however relationship between the tables should be separately defined. Dashboard for an application will be designed with widgets and plots available within ELQA framework and Bokeh library.

5. Experiments

This section presents the results of the experiments which were conducted in order to validate performance of the LSTM network in an voltage time series modeling task. All the experiments required several steps of preparation which were mostly related to a data preprocessing.

5.1. Setup description

Data acquired from all the CERN accelerators are kept in the global database called Logging Service (LS) [8]. Despite the very low time resolution (one sample for 400 ms) of this data, it is possible to examine the feasibility of using LSTM for modeling the behavior of the magnets. A generic Java GUI called TIMBER [35] and a dedicated Python wrapper [36] are provided as tools to visualize and extract logged data.

The logging database stores a record of many years of the magnets activity. This is a huge amount of data with relatively few quench events. Since part of the planned research focuses on quench prediction and/or detection it was important to include such a data in constructed data sets. It is worth noting that one day-long record of single voltage time series for a single magnet occupies roughly 100 MB. There are several voltage time series associated with a single magnet [35] but ultimately authors decided to use U_{res} in the experiments. The origin and the meaning of the resistive voltage U_{res} were discussed in the Section 2.2.

There are various kinds of magnets located in the LHC tunnel and they generate different number of quench events (Table 2, Fig. 8). It is beneficial to choose a group of magnets for which the largest possible number of quenches was recorded. The longest history of quenches was provided for 600 A magnets in LS database. Therefore, we decided to focus our initial research on 600 A magnets. Unfortunately, the 600 A magnets data stored in a database is very large i.e. an order of several gigabytes. However, as it was mentioned before, the activity record of superconducting magnets during operational time of the LHC is composed mostly of sections of normal operation and only sometimes quench events happen. Furthermore, the logging database does not enable automated quench periods extraction, despite having many useful features for a data preprocessing and information extraction.

It would be a tedious work to manually extract all the quenches. Therefore, a quench extraction application (presented in Fig. 9) was developed, which automates the process of fetching the voltage time series from the LS database. It is composed of a set of Python scripts which generate appropriately prepared queries to the LS database. The queries are built based on the quench list [13] and data extraction parameters configuration files. Once the data has been fetched from the LS database it is normalized to the range from 0 to 1 and split into training and testing set: 70 % of the data is used for training and 30 % for testing.

Different lengths of time window frame before and after the quench events were considered. Ultimately, we chose in our view a reasonable trade-off between the amount of data and their representativeness for the model i.e. 24 h long time window before a quench event. We extracted days on which quenches occurred between the years 2008 and 2016, which amounted to 425 in total for 600 A magnets (Table 2).

A training of deep learning models takes long time even when fast GPUs are employed for the calculations. Therefore, it is important what kind of and how large data sets are used for training and testing the models. Furthermore, it is important to preliminary adjust hyper-parameters of the model using relatively small data set when a single iteration time is short. Thereafter, tiny updates are done using the largest data set, when each training routine of the network consumes substantial amount of time. Thus, we have created three different data sets: small, medium and the large ones as presented in Table 3.

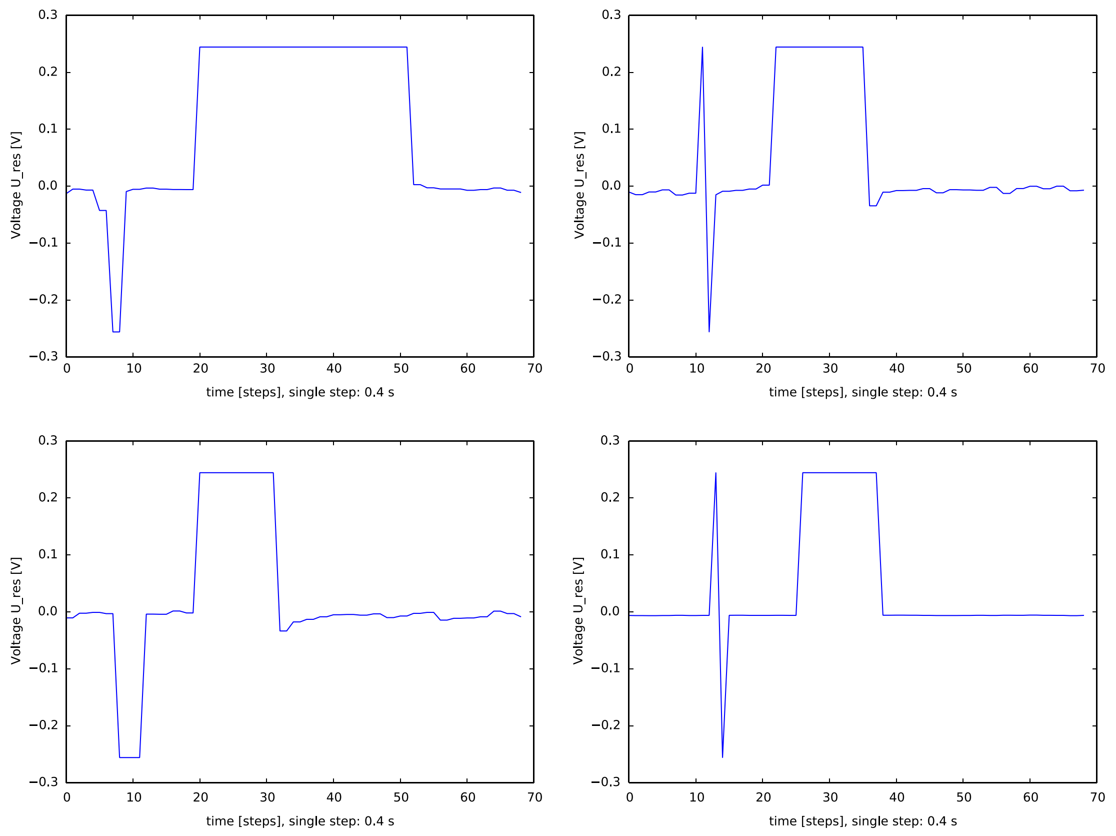

Fig. 8. The selected sample anomalies of 600 A magnets extracted from the LS database.

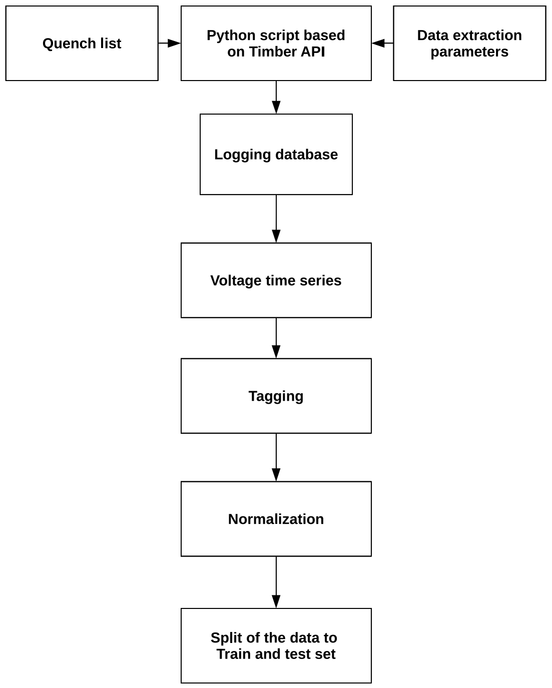
Quench list
Python script based on Timber API
Data extraction parameters
Logging database
Voltage time series
Tagging
Normalization
Split of the data to Train and test set

Fig. 9. The procedure for extraction of voltage time series with anomalies from the LS database.

5.2. Results and analysis

The designed LSTM model was examined with a respect to its ability of anticipating few voltage values forward. The core architecture of the module used for the experiments is presented in Fig. 10, but we trained and tested various models with wide range of parameters such as number of neurons, layers and inputs. The primary goal was to find a core set of the parameters which enabled modeling of the U_{res} voltage data.

Table 3
The data sets used for training and testing the model.

Data set	Size [MB]
Small	22
Medium	111
Large	5000

The core architecture of the network module is composed of seven layers all together: an input layer, four LSTM hidden layers, one feed-forward hidden layer and an output layer. It is worth noting that dropout operations are also classified as separate layers. Every second layer among layers of the LSTM type has a dropout with a value of 20%. Furthermore, a number of LSTM cells in the middle layer were changed from 32 to 128 and then to 512 in order to examine the performance of the model as a function of the number of neurons. The module was implemented in Keras [37] with the Theano backend [38].

Fig. 11 shows both real voltage signal and its prediction. Visual similarity analysis is neither efficient nor recommended for validation of regression models therefore authors have decided to use more reliable measures such as Root Mean Square Error (RMSE) and Mean Percentage Error (MPE).

The measures are given by the following equations:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y^{(t)} - \hat{y}^{(t)})^2} \tag{10}$$

$$MPE = \frac{100\%}{N}\sum_{t=1}^{N}\frac{y^{(t)} - \hat{y}^{(t)}}{y^{(t)}} \tag{11}$$

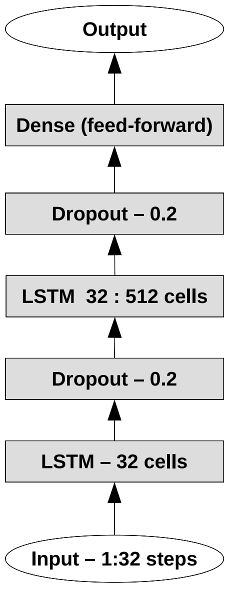

Fig. 10. The LSTM-based network used for the experiments.

where: $y^{(t)}$ and $\hat{y}^{(t)}$ is a voltage time series and its predicted counterpart, respectively. Both Eqs. (10) and (11) are calculated for N data points which in turn depend on the size of data set that is used to train and test the model.

Fig. 12–14 present the prediction results in terms of RMSE for 32 future steps. The model was trained using the medium data corpus. Two quantities were used as parameters:

L	- a number of previous time steps to use as input variables to predict the next time period,
B	- a size of training batch.

The experiments were conducted for three different L values: 1, 16 and 32 which also affected the model input size. The more steps back in time are taken into account in model training and testing processes the wider input should be used. The size of the model input is equal to a number of steps back L in time which are taken for building the LSTM model. Furthermore, four different batch sizes B values were tested: 32, 128, 512 and 2048. The batch size B has two-fold effect on the performance of the model. On the one hand it affects a range of the voltage series which is processed by the model. On the other hand, the larger batches are computed faster on GPUs because matrix calculation optimization measures may be applied.

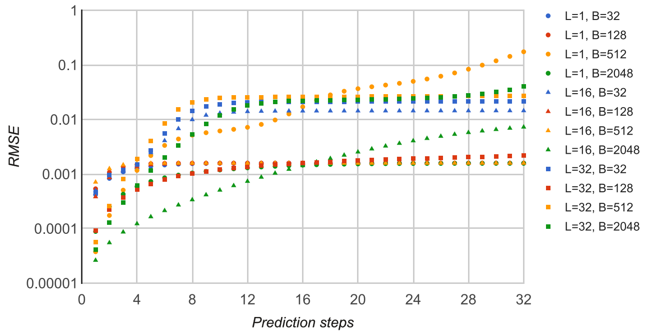

Fig. 12. The value of RMSE as a function of prediction steps for different batch size B and number of previous time steps L values with 32 neurons in the middle LSTM layer.

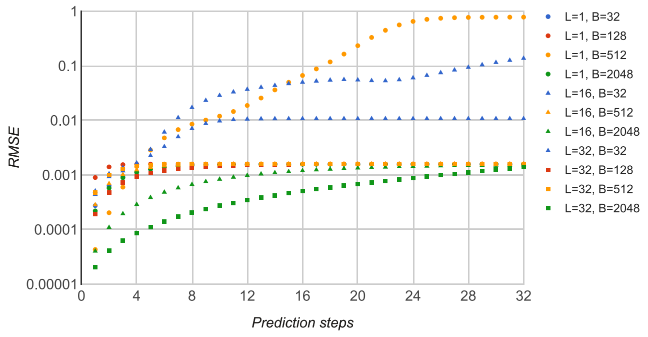

Fig. 13. The value of RMSE as a function of prediction steps for different batch size B and number of previous time steps L values with 128 neurons in the middle LSTM layer.

According to Fig. 12, the best result (in terms of mean RMSE) in the experiment with 32 neurons used in the middle LSTM layer was obtained for $L = 16$ and batch size $B = 2048$. In the case of the experiments with 128 neurons, the parameters combination of $L = 32$ and batch size $B = 2048$ resulted in the best result in terms of RMSE as it is presented in Fig. 13.

The MPE values, according to Eq. (11), were computed for a selected quench fragments in parallel to the RMSE calculation of the whole voltage time series. The results of the MPE calculation generally follow the RMSE trend, therefore we decided not to include them. However, a sample MPE plot is presented in Fig. 15. It is worth noting that a prediction quality is lower for more steps ahead. This is due to the fact that a prediction of t_{n+1} time step is based on a previous t_n step.

Table 4 presents the results in terms of the mean value of the RMSE. The results were obtained by averaging the RMSE error over all the steps in the future. It is worth emphasizing that the lowest RMSE error is

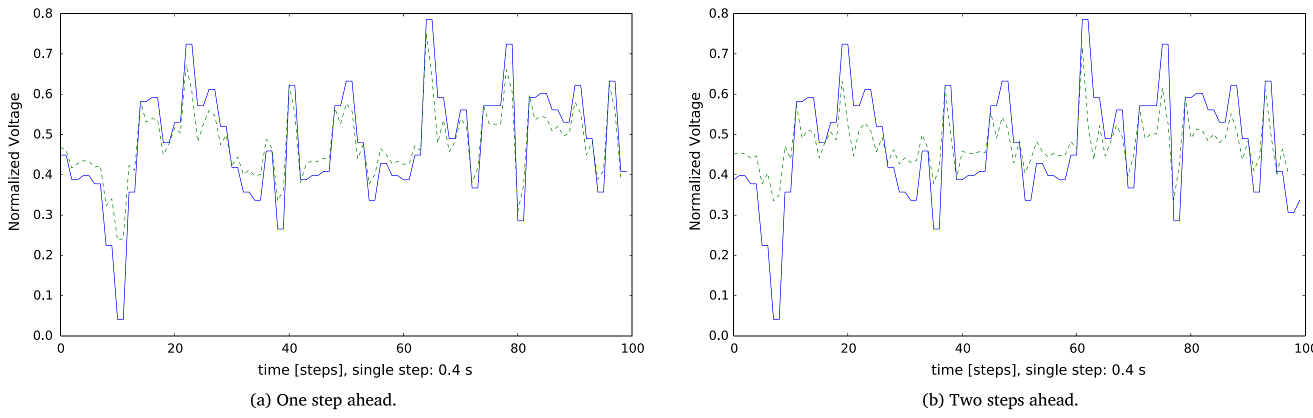

Fig. 11. Two examples of prediction for one and two steps ahead in time. Predicted signal is plotted in a green broken line.

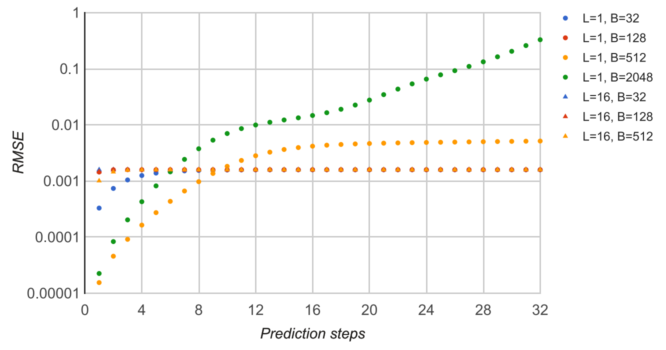

Fig. 14. The value of RMSE as a function of prediction steps for different batch size B and number of previous time steps L values with 512 neurons in the middle LSTM layer.

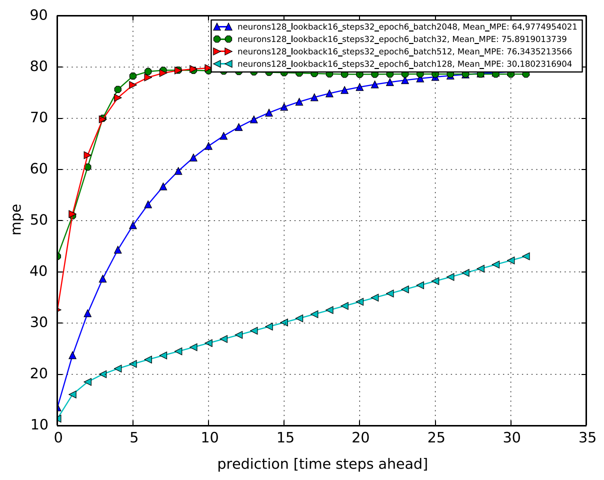

Fig. 15. The example of the MPE plot as a function of prediction steps for different batch size B and number of previous time steps L values.

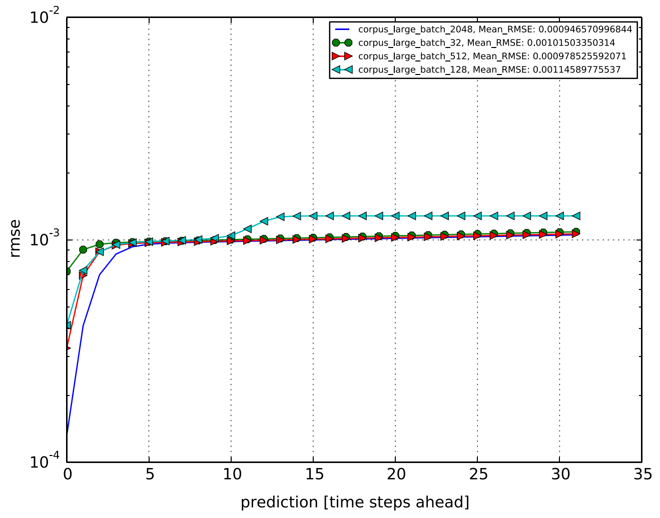

Fig. 16. The example of the RMSE plot as a function of prediction steps for the large corpus (Table 3).

achieved for predictions with a small number of forward steps. The more steps to the future are predicted, the worse results are obtained which is reflected in the rising RMSE value. Therefore, the results provided by Table 4 should be considered as an approximate performance of the LSTM model. Nevertheless, it is noticeable that the best two results are achieved for batch size of 2048.

Table 4
The best results obtained for the medium corpus (Table 3).

LSTM cells	L	B	Mean RMSE
128	16	2048	0.001 04
32	1	2048	0.001 25
128	32	128	0.001 40
32	1	32	0.001 48

Table 5
The parameters of the LSTM network used to the experiments.

Parameter	Value
Number of layers	5
Number of epochs	6
Total number of the network parameters	21 025
Dropout	0.2
Max. number of steps ahead	32

All the tests presented in this section were performed on Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz with 32 GB DDR3 1600 MHz memory. The processing time was long — it took over two weeks to compute the presented results. The biggest contribution was the training time of the LSTM model, which was significantly higher for the architectures with more neurons. This was the main reason why the model was trained over only six epochs as presented in Table 5.

A test for the large corpus (Table 3) of 5 GB was also conducted and an example of the results is presented in Fig. 16. Because of the long computation time we were not able to conduct the same set of experiments as we did for the medium size corpus. Nevertheless, the results we managed to gather show that the RMSE value converges to a value on the level of 0.001. This was expected since much more data was used for the experiment with the large corpus.

6. Discussion

6.1. Solution applicability

Safety of the systems used in LHC is of a high importance. As a result, any neural-network-based application is best thought of as an addition to or enhancement of the highly dependable current system.

There is also a broad range of possible applications of RNN-based solutions in the other CERN control systems, such as cryogenics, vacuum, machine protection and power converters [39,40]. Examples include:

- anomaly detection on beam screen,
- faulty cryogenics valve detection, and
- Future Circular Collider (FCC) Reliability, Availability, Maintainability and Safety (RAMS) studies.

6.2. Anomalies classification

It is worth emphasizing that in order to classify anomalies it is essential to map regression results to classification task. In other words, it would be necessary to express RMSE in terms of F1 score. Unfortunately, this would require well defined threshold of RMSE value, which would discriminate positive and negative classification results [41]. At the current stage of the research, such a threshold value has not been determined and requires further investigation. It may also be possible to adopt different approaches to the conversion to a classification task, especially ones not requiring data of anomalous behavior during model training [42–44]. This is a subject of the ongoing research.

6.3. System reaction time

The authors are currently implementing RNN LSTM module on FPGA to be used for anomaly detection. However, it is worth noting that networks of the similar size as the one which is to be used for the

Table 6
Performance of various approaches to LSTM hardware implementation (data from [45–47]).

Setup	Platform	Computation time [μs]
2 layers (128 cells), 32/16 bit	Xilinx Zynq 7020 (142 MHz), external memory — DDR3	~932
Compressed LSTM (20x), 1024 cells	Xilinx XCKU060 Kintex (200 MHz), external memory — DDR3	82.7
2 layers (30, 256 cells), 6-bit quantization	Xilinx Zynq XC7Z045 (100 MHz) 2.18 MB on-chip memory max, all in the internal memory	15.96

system described in the paper were already implemented and described in the following papers [45–47]. The performance of the LSTM module implemented on FPGA strictly depends on three main factors:

- memory footprint of the network,
- localization of the network weights (external or internal memory),
- degree to which the network is compressed.

It is worth emphasizing that the network weights (coefficients) are used for every single iteration of the computations. Therefore keeping them in the internal memory of the processing unit (FPGA) is highly beneficial for the performance of the module.

Table 6 shows results gathered from three papers which adopt various approaches to illustrate discrepancies in performance across them.

The first approach [45] shows uncompressed network (with high data representation precision) and the weights are stored in the external memory. It takes roughly 1 ms to process a single LSTM iteration.

The second approach [46] takes advantage of weights compression which affects significantly the performance of the module. The weights were reduced to 12 bits which resulted in a drop of an amount data which is fetched from the memory for the computations in each iteration of the algorithm. Consequently, the module is capable of achieving ~83 μs for the single LSTM iteration processing time.

The third approach [47] adopts both coefficients compression and optimal weights localization. All the coefficients are kept in the internal BRAM memory. This is possible due to the high compression ratio of the weights. Huge gain in the performance may be noticed compared to the two previously presented approaches (single iteration takes ~16 μs).

It is worth noting that the further performance improvement is possible by moving coefficients from BRAM memories to the distributed ones.

The total acquisition time i.e. input signal to FPGA delivery depends on PCB (Printed Circuit Board) implementation as well as the components used in a given application such as ADC (Analog to Digital Converter). The time is similar to any FPGA-based project with custom-designed PCB. The total time estimation and analysis is out of scope of this paper, which is primarily focused on a feasibility of using LSTM for the LHC magnets monitoring.

7. Conclusions and future work

The data, including quench events as sample anomalies, acquired from the logging database were used to verify LSTM recurrent neural networks ability to model voltage time series of LHC superconducting magnets. It has been proved that LSTM-based setup performs well, with RMSE value approaching 0.001 for the largest data set used. As it was expected, prediction results for more steps ahead are inferior to the short time prediction in terms of accuracy expressed in RMSE.

As a future work, the Post Mortem data of much bigger resolution will be used for model training and the algorithm for anomalies detection and/or prediction will be developed and verified. Authors are also going to implement the prediction stage of the algorithm in FPGA or ASIC to evaluate its performance in real-time applications.

References

[1] A. Wright, R. Webb, The Large Hardron Collider, Nature 448 (7151) (2007) 269-269. http://dx.doi.org/10.1038/448269a.

[2] L. Evans, P. Bryant, LHC machine, J. Instrum. 3 (08) (2008) S08001. http://dx.doi.org/10.1088/1748-0221/3/08/S08001.

[3] R. Denz, Electronic systems for the protection of superconducting elements in the LHC, IEEE Trans. Appl. Supercond. (ISSN: 1051-8223) 16 (2) (2006) 1725–1728. http://dx.doi.org/10.1109/TASC.2005.864258.

[4] L. Bottura, Cable stability, in: Proceedings of the CAS-CERN Accelerator School: Superconductivity for Accelerators, Erice, Italy, CERN-2014-005, 2014, pp. 401–451. http://dx.doi.org/10.5170/CERN-2014-005.401. arXiv:1412.5373.

[5] J. Steckert, A. Skoczen, Design of FPGA-based radiation tolerant quench detectors for LHC, J. Instrum. 12 (04) (2017) T04005. http://dx.doi.org/10.1088/1748-0221/12/04/T04005.

[6] M. Calvi, L. Agrisani, L. Bottura, A. Masi, A. Siemko, On the use of wavelet transform for quench precursors characterization in the LHC superconducting dipole magnets, IEEE Trans. Appl. Supercond. (ISSN: 1051-8223) 16 (2) (2006) 1811–1814. http://dx.doi.org/10.1109/TASC.2006.871308.

[7] C. Lorin, A. Siemko, E. Todesco, A. Verweij, Predicting the quench behavior of the LHC dipoles during commissioning, IEEE Trans. Appl. Supercond. (ISSN: 1051-8223) 20 (3) (2010) 135–139. http://dx.doi.org/10.1109/TASC.2010.2043076.

[8] C. Roderick, G. Kruk, L. Burdzanowski, The CERN accelerator logging service-10 years in operation: a look at the past, present and future, in: Proceedings of ICALEPCS2013, 2013, pp. 612–614. URL http://cds.cern.ch/record/1611082.

[9] ATLAS Collaboration, Observation of a new particle in the search for the standard model Higgs boson with the ATLAS detector at the LHC, Phys. Lett. B (ISSN: 0370-2693) 716 (1) (2012) 1–29. http://dx.doi.org/10.1016/j.physletb.2012.08.020. arXiv:1207.7214.

[10] CMS Collaboration, Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC, Phys. Lett. B (ISSN: 0370-2693) 716 (1) (2012) 30–61. http://dx.doi.org/10.1016/j.physletb.2012.08.021.

[11] L. Burnod, D. Leroy, Influence of Beam Losses on the LHC Magnets, Tech. Rep. LHC-NOTE-65, CERN, 1987. http://cds.cern.ch/record/1017593.

[12] Layout Database, https://layout.web.cern.ch/.

[13] Quench Database, https://twiki.cern.ch/twiki/bin/viewauth/MP3/QuenchDatabase.

[14] E. Ciapala, F. Rodríguez-Mateos, R. Schmidt, J. Wenninger, The LHC Post-mortem System, Tech. Rep. LHC-PROJECT-NOTE-303, CERN, Geneva, 2002. http://cds.cern.ch/record/691828.

[15] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105 URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[16] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature (ISSN: 0028-0836) 521 (7553) (2015) 436–444. http://dx.doi.org/10.1038/nature14539. Insight.

[17] A. Graves, Neural networks, in: Supervised Sequence Labelling with Recurrent Neural Networks, Springer, Berlin, Heidelberg, ISBN: 978-3-642-24797-2, 2012. http://dx.doi.org/10.1007/978-3-642-24797-2.

[18] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Netw. (ISSN: 0893-6080) 61 (2015) 85–117. http://dx.doi.org/10.1016/j.neunet.2014.09.003. arXiv:1404.7828.

[19] L. Deng, D. Yu, Deep Learning: Methods and Applications, Tech. Rep. MSR-TR-2014-21, NOW Publishers, 2014 URL https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/.

[20] C. Farabet, C. Couprie, L. Najman, Y. LeCun, Learning hierarchical features for scene labelling, IEEE Trans. Pattern Anal. Mach. Intell. (ISSN: 0162-8828) 35 (8) (2013) 1915–1929. http://dx.doi.org/10.1109/TPAMI.2012.231.

[21] Y. LeCun, Deep learning of convolutional networks, in: 2015 IEEE Hot Chips 27 Symposium, HCS, 2015, pp. 1–95. http://dx.doi.org/10.1109/HOTCHIPS.2015.7477328.

[22] J. Morton, T.A. Wheeler, M.J. Kochenderfer, Analysis of recurrent neural networks for probabilistic modelling of driver behaviour, IEEE Trans. Intel. Trans. Syst. (ISSN: 1524-9050) PP (99) (2016) 1–10. http://dx.doi.org/10.1109/TITS.2016.2603007.

[23] F. Pouladi, H. Salehinejad, A.M. Gilani, Recurrent Neural Networks for Sequential Phenotype Prediction in Genomics, in: 2015 International Conference on Developments of E-Systems Engineering, DeSE, 2015, pp. 225–230. http://dx.doi.org/10.1109/DeSE.2015.52.

[24] X. Chen, X. Liu, Y. Wang, M. J.F. Gales, P.C. Woodland, Efficient training and evaluation of recurrent neural network language models for automatic speech recognition, IEEE/ACM Trans. Audio, Speech, Lang. Process. (ISSN: 2329-9290) 24 (11) (2016) 2146–2157. http://dx.doi.org/10.1109/TASLP.2016.2598304.
[25] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Comput. (ISSN: 0899-7667) 9 (8) (1997) 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.
[26] C.Y. Chang, J.J. Li, Application of deep learning for recognizing infant cries, in: 2016 IEEE International Conference on Consumer Electronics-Taiwan, ICCE-TW, 2016, pp. 1–2, http://dx.doi.org/10.1109/ICCE-TW.2016.7520947.
[27] M. Cai, Y. Shi, J. Liu, Deep maxout neural networks for speech recognition, in: Automatic Speech Recognition and Understanding, ASRU, 2013 IEEE Workshop on, 2013, pp. 291–296., http://dx.doi.org/10.1109/ASRU.2013.6707745.
[28] L. Tóth, Phone recognition with deep sparse rectifier neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, ISSN 1520-6149, 2013, pp. 6985–6989, http://dx.doi.org/10.1109/ICASSP.2013.6639016.
[29] Z.C. Lipton, J. Berkowitz, C. Elkan, A Critical Review of Recurrent Neural Networks for Sequence Learning, 2015. arXiv:1506.00019.
[30] K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, J. Schmidhuber, LSTM: A Search Space Odyssey, 2015, arXiv:1503.04069.
[31] Django Software Foundation, Django documentation, 2015, https://docs.djangoproject.com/en/1.8/ref/databases/.
[32] Bokeh Development Team, Bokeh: Python library for interactive visualization, 2014, http://www.bokeh.pydata.org.
[33] L. Barnard, M. Mertik, Usability of visualization libraries for web browsers for use in scientific analysis, Int. J. Comput. Appl. 121 (1) (2015).
[34] cx_Oracle, 2015, http://sourceforge.net/projects/cx-oracle/.
[35] Logging Database TIMBER, http://lhc-logging.web.cern.ch/lhc-logging/timber/.
[36] R.D. Maria, Python API to Timber database - pytimber, 2016, https://github.com/rdemaria/pytimber.
[37] F. Chollet, Keras, GitHub, 2015. https://github.com/fchollet/keras.
[38] Theano Development Team, Theano: A Python Framework for Fast Computation of Mathematical Expressions, 2016, arXiv:1605.02688.
[39] A.R. Marín, M.M. Marquez, Fault Detection using Advanced Analytics at CERN's Large Hardron Collider, 2017, URL https://indico.cern.ch/event/515136/attachments/1249768/1842142/20160126-BIWA-ORE-Final.pdf, cited 27.04.2017.
[40] L. Tsan, CERN Big Data Exploration for FCC, 2016, https://blogs.oracle.com/infrastructure/cern-big-data-exploration-for-fcc, cited 27.04.2017.
[41] A. Nanduri, L. Sherry, Anomaly detection in aircraft data using Recurrent Neural Networks (RNN), in: 2016 Integrated Communications Navigation and Surveillance, ICNS, 2016, pp. 5C2-1-5C2-8, http://dx.doi.org/10.1109/ICNSURV.2016.7486356.
[42] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, Long Short Term Memory Networks for Anomaly Detection in Time Series, in: 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2015, Bruges (Belgium), 2015, Proceedings, Presses universitaires de Louvain, 2015, pp. 89–94 URL https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2015-56.pdf.
[43] L. Bontemps, V.L. Cao, J. McDermott, N.-A. Le-Khac, Collective Anomaly Detection Based on Long Short-Term Memory Recurrent Neural Networks, in: T.K. Dang, R. Wagner, J. Küng, N. Thoai, M. Takizawa, E. Neuhold (Eds.), Future Data and Security Engineering: Third International Conference, FDSE 2016, Can Tho City, Vietnam, November 23–25, 2016, Proceedings, Springer International Publishing, Cham, ISBN: 978-3-319-48057-2, 2016, pp. 141–152. http://dx.doi.org/10.1007/978-3-319-48057-2_9.
[44] T.J. O'Shea, T.C. Clancy, R.W. McGwier, Recurrent Neural Radio Anomaly Detection, 2016. arXiv:1611.00301.
[45] A.X.M. Chang, B. Martini, E. Culurciello, Recurrent Neural Networks Hardware Implementation on FPGA, 2015. arXiv:1511.05552.
[46] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang, H. Yang, W.B.J. Dally, ESE: Efficient Speech Recognition Engine with Sparse LSTM on FPGA, in: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '17, 2017, pp. 75–84. http://dx.doi.org/10.1145/3020078.3021745. arXiv:1612.00694.
[47] M. Lee, K. Hwang, J. Park, S. Choi, S. Shin, W. Sung, FPGA-Based Low-Power Speech Recognition with Recurrent Neural Networks, 2016, arXiv:1610.00552.