Mateusz Kocot
Krzysztof Misan
Valentina Avati
Edoardo Bossini
Leszek Grzanka
Nicola Minafra

USING DEEP NEURAL NETWORKS TO IMPROVE THE PRECISION OF FAST-SAMPLED PARTICLE TIMING DETECTORS

Abstract

Measurements from particle timing detectors are often affected by the time walk effect caused by statistical fluctuations in the charge deposited by passing particles. The constant fraction discriminator (CFD) algorithm is frequently used to mitigate this effect both in test setups and in running experiments, such as the CMS-PPS system at the CERN's LHC. The CFD is simple and effective but does not leverage all voltage samples in a time series. Its performance could be enhanced with deep neural networks, which are commonly used for time series analysis, including computing the particle arrival time. We evaluated various neural network architectures using data acquired at the test beam facility in the DESY-II synchrotron, where a precise MCP (MicroChannel Plate) detector was installed in addition to PPS diamond timing detectors. MCP measurements were used as a reference to train the networks and compare the results with the standard CFD method. Ultimately, we improved the timing precision by 8% to 23%, depending on the detector's readout channel. The best results were obtained using a UNet-based model, which outperformed classical convolutional networks and the multilayer perceptron.

1. Introduction

Precise time measurements of particles are crucial in many fields, including nuclear medicine and high-energy physics. Designing the most efficient method can be challenging, especially when the required precision is of the order of nanoseconds or picoseconds. Our research focuses on the detectors of the Precision Proton Spectrometer (PPS) [1], which is a subsystem of the Compact Muon Solenoid (CMS) [23] detector at CERN's Large Hadron Collider (LHC) [12]. At the LHC, protons and ions are accelerated to high energies and are then collided in dedicated interaction points. PPS detects and measures the kinematics of so-called forward protons, which are scattered to small angles after the interaction. Accurate calculation of the particle position and arrival time allows for the precise reconstruction of the particle trajectory and estimation of the interaction position which is crucial for the CMS-PPS subsystem.

The CMS-PPS system uses detectors installed on both sides of CMS, at a distance of approximately 220 m, to perform precise time measurements [7]. Each detector contains four detection planes with scCVD (single crystal Chemical Vapour Deposition) diamond sensors. When a charged particle passes through a sensor, it generates an electric analogue signal that is later amplified and digitised. In LHC Run 3[1], one of the digitisation techniques uses SAMPIC [11], a fast sampling ASIC (Application-Specific Integrated Circuit), on which we focus in our work. The chip samples and digitises the signal every 156.25 ps. Proper online or offline[2] analysis of these data including a multi-step preprocessing and filtering procedure can be used to compute the particle arrival time with high precision. The data flow during the analysis is depicted in Figure 1.
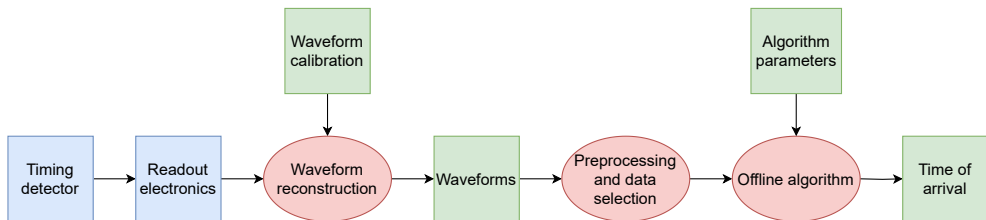

Figure 1. Data flow diagram for the analysis of the timing data (blue: electronics, green: data, red: digital algorithm). Multiple algorithms can be used to retrieve the time of arrival from the waveform data. In this research we focus on digital algorithms working in the offline mode.

The accuracy of the arrival time measurement is impacted by two main factors: jitter and the time walk effect. In our work, we focus on minimising the impact

---

[1]The operating period of the LHC, which started in 2022.

[2]The word 'offline' in this article is used to describe an algorithm working some time after data acquisition, contrary to online algorithms which work in the software or analogue pipeline straight after the data have been acquired.

of these components on the data from PPS sensors. Jitter is caused by the noise from the signal amplifier. Time walk is the dependence of the measured time on the signal amplitude. It is caused by statistical fluctuations of the charge released in a sensor by a passing particle. This leads to detecting signals with variable amplitudes. Signals with larger amplitudes cross a given threshold earlier than signals with smaller amplitudes. An example of measurements affected by the time walk effect is provided in Figure 2.
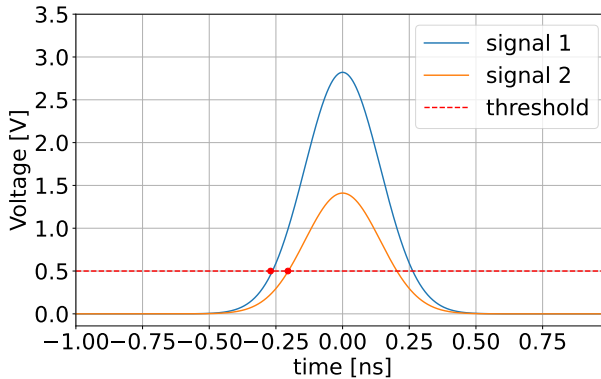

Figure 2. Time walk error illustrated as a difference in the threshold crossing times between two signals with the same shapes but different amplitudes

The constant fraction discriminator (CFD) is currently used at CMS-PPS to reduce the impact of the time walk effect and extract the time of arrival timestamps. The CFD is an analytical algorithm and does not use all available samples in a time series. Furthermore, the quality of its results is substantially reduced by the presence of noise and waveform irregularities. To address this issue, we propose a solution that utilises a deep neural network. This network can predict the arrival time of a particle from a sampled time series by using all available samples in a waveform.

2. State of the art

Various digital techniques are used to obtain the time of arrival. The classical approach is to use one of the multiple analytical methods. Following the recent trends, machine learning techniques are gaining popularity in this domain, too. This section outlines both of these strategies.

2.1. Analytical approaches

The simplest analytical method is the fixed threshold, which extracts the timestamp as the time of crossing a threshold fixed at a specific voltage. The main flaw of the fixed threshold is not taking the time walk effect into consideration at all.

The most common method used to mitigate the time walk effect is the normalised threshold algorithm, often referred to as the constant fraction discriminator (CFD). This algorithm normalises the waveform amplitude and then applies a fixed threshold. Other techniques include using the signal maximum as the timestamp or extracting only two timestamps and using the time over threshold (TOT) method [5, 9].

Due to its simplicity and relatively high performance, the constant fraction discriminator is considered the most reliable choice [6]. It is used in both test setups and running experiments, such as the CMS-PPS system at the LHC. Originally, the CFD was devised as an analogue device. However, we use it as an offline algorithmic solution to measure the arrival time of a particle given a very fast electrical pulse. By mitigating the error introduced by the time walk effect, the CFD allows for very accurate timing measurements. Given the excellent properties of the CFD and its common usage in the field, we selected this algorithm as the baseline for our numerical experiments.

2.2. Machine learning methods

Machine learning techniques are widely used in high energy physics. Common use cases include monitoring data quality by identifying outliers [2] and particle track reconstruction [19]. The short execution time of machine learning methods makes them useful for the high level trigger reconstruction task [18].

Although some supervised machine learning techniques, specifically deep neural networks, show promising results in time series analysis and timestamp prediction [21], they are seldom used to predict the time of arrival and have never been utilised for this purpose in the CMS-PPS subsystem.

The most extensive tests of neural networks in the domain of computing the time of arrival have been performed for MRPC (Multigap Resistive Plate Chamber) detectors. The research showed that multilayer perceptrons, LSTM (Long Short-Term Memory) recurrent neural networks or their combinations can be successfully used with the signals from a particle timing detector [26, 27]. The interest is high in medical applications, too. Various convolutional architectures, mainly UNet-based, are used to calculate the time of flight in PET detectors [4] and to tag ECG diagrams [20, 28]. While these problems are different from the one discussed in this paper, they still require time series tagging, which is at the core of our problem.

3. Dataset

This section describes the data source and preprocessing steps required to construct the dataset used in this work. We also provide a detailed description of the version of the CFD algorithm, which is used during the data preprocessing procedure.

3.1. Data source

We constructed a dataset using the data acquired at the test beam facility in the DESY-II synchrotron in 2020 [8]. The facility hosted the PPS diamond timing detectors, as well as an MCP-PMT (Microchannel Plate Photomultiplier Tube) detector. The sensors were connected to the SAMPIC readout chip. The voltage time series sampled by SAMPIC had a fixed length of 64 samples within a 10 ns time window. Typically, the time window was long enough to capture the entire MCP signal. However, with diamond sensors, the signal is typically longer, with a wider trailing edge compared to the leading edge. As a result, in most cases, 10 ns was enough to fully capture only the leading edge of a signal.

The expected precision of the PPS diamond sensors was 50–100 ps, while the MCP timing precision was measured to be around 10 ps [8]. Considering its performance, MCP readouts were a perfect source of ground-truth information for our experiments. We present examples of waveforms acquired using the MCP and a diamond sensor in Figure 3.
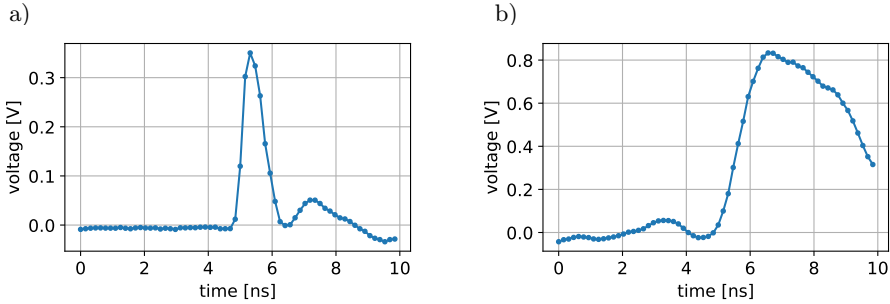

Figure 3. Example waveforms from the MCP (a) and a diamond detector (b)

3.2. Preprocessing

Multiple preprocessing steps were required for the data acquired from the DESY beam. Firstly, the samples were inverted to compensate for negative signals. This operation resulted in a rising edge in a signal indicating a particle.

The next preprocessing step was to eliminate noisy and so-called saturated events. Noisy signals are too weak to analyse, while saturated events occur when the voltage exceeds the amplifier's dynamic range, resulting in the signal being capped at a certain level. Examples of such events are shown in Figure 4. This means that the true amplitude cannot be easily retrieved from the signal, and the CFD cannot produce accurate reference values. We excluded noisy and saturated events from the analysis to focus solely on the comparison between deep learning models and the CFD.

The data filtering was done mainly using amplitude histograms. Amplitude (i.e. maximum voltage) was plotted on a histogram for each event. Typically, noisy events appear on the left side of the histogram, while saturated events appear on the right

side. Therefore, it is simple to filter out such events by selecting minimum and maximum amplitudes and discarding any events that fall outside this range. The minimum and maximum cuts were determined manually by analysing the histogram shapes. Figure 5 shows the maximum voltage histograms for the MCP and two selected diamond detectors. Due to its excellent waveform quality and low ratio of saturated events (visible as a small peak on the right side of its histogram), MCP did not require filtering.
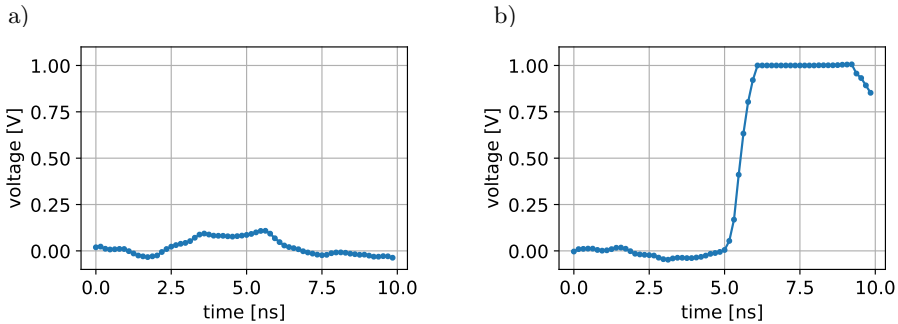
a)   b)

Figure 4. Examples of noisy (a) and saturated (b) events acquired
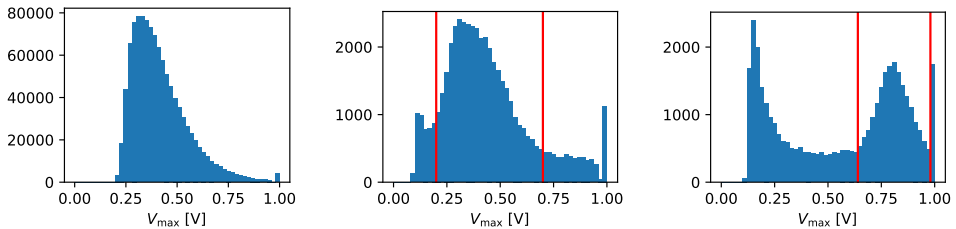using the diamond sensor

Figure 5. Maximum voltage histograms for MCP (left) and two selected diamond detectors (middle and right). The red lines on the histograms of the diamond detectors indicate the minimum and maximum amplitude cuts. It is important to note that the histograms look different for each detector, and therefore, it was necessary to find the amplitude cut values separately for each of them.

In the original dataset, the distribution of signal rising edge timestamps was centred around a single value in the middle of the time window. Using such a dataset would make it highly likely for the networks to overfit and collapse the predictions to the same timestamp for any input data. To avoid this, the signals were trimmed from 64 to 48 samples by removing 16 samples from the edges of the window. Specifically, up to 16 first samples were cut from each signal, and only the next 48 samples were kept. The number of first samples to drop was chosen randomly to smear the timestamp distribution.

Another issue was the fact that the waveforms had varying baselines and amplitudes. They had to be normalised in order to be properly processed by the neural networks. At first, the baseline was calculated as the mean value of the first 20 samples. Then, it was subtracted from the voltage values in the time series. Afterwards, the waveforms were divided by their maximums to normalise the amplitudes. The same steps are used in our version of the CFD algorithm and are visualised in Figure 7a.

3.3. Final dataset

In order to train the neural networks, we needed both time series and ground-truth (reference) time. Therefore, we constructed a dataset that only included events with corresponding readouts from both the MCP and a diamond sensor. The ground-truth timestamps were obtained from the MCP signals using the CFD method explained below. Due to the high quality of the MCP measurements, this approach was sufficient and provided the necessary timing precision for the training process. Figure 6 presents a normalised waveform from the final dataset, along with a corresponding MCP signal and the reference timestamp. The slight difference between the reference time shown on the MCP and diamond detector waveforms is due to the relative difference in the times of the first samples of both signals. This difference must be taken into account in the calculations to avoid bias.
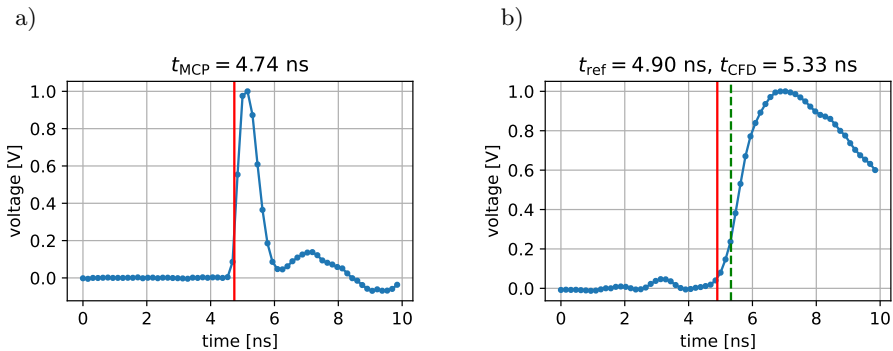
Figure 6. Example from the final dataset: a) waveform from MCP. t_{MCP} (red line) is computed using the CFD; b) waveform from a diamond detector. t_{ref} (red line) is the neural network's reference time computed using the CFD for the MCP waveform. The green dashed line represents the time computed using the CFD with the waveform from the diamond detector (t_{CFD}). It is not included in the dataset and is shown only for visualisation purposes

The final dataset consisted of approximately 500,000 waveform entries and their corresponding reference timestamps. After removing irrelevant information, performing data filtering and preprocessing, the size of the dataset was reduced from around 5.5 GB to 126 MB.

3.4. Constant fraction discriminator

The CFD algorithm used in this research is a modified version of the normalised threshold algorithm.

First, we normalise a time series using the same strategy as during data preprocessing, which involves baseline subtraction and division by the amplitude. Next, we calculate the time of arrival as the moment when the series crosses a chosen voltage threshold. To determine the exact timestamp, we apply a linear interpolation between the point before and the point after the threshold crossing. The chosen threshold is a fraction of the normalised amplitude, which ensures that the crossing point's dependence on the amplitude is removed. Figure 7 illustrates these steps.
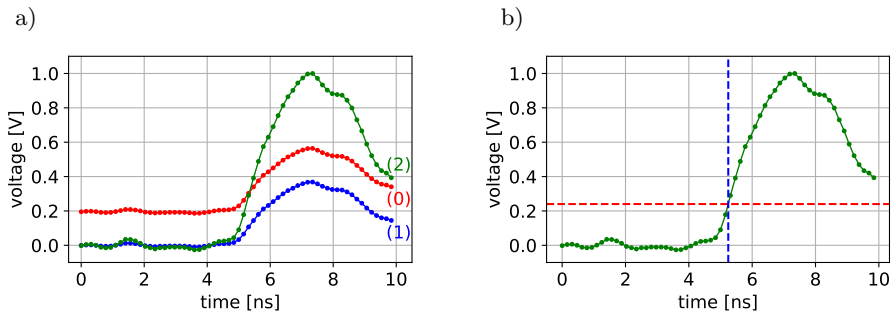

Figure 7. Depiction of the CFD algorithm: a) (0) before normalisation, (1) baseline subtraction, (2) division by maximum; b) when the normalisation is done, the timestamp can be found using the fixed threshold algorithm. The threshold chosen for this example is arbitrary

4. Network architectures

Our goal was to improve the timing precision using deep neural networks. We aimed to demonstrate that our methods achieve better results in terms of particle arrival time precision than the CFD. We started from a multilayer perceptron (MLP) and progressively increased the overall complexity of the network structure by using regular convolutional architectures and UNet-based [17] networks. We ran a hyperparameter tuning algorithm for each network type and selected the best candidates. Below, we briefly describe the training configuration, tuning method and hyperparameter options. We also depict the best-performing models.

4.1. Training configuration

We selected a single detector readout channel, i.e. a single diamond detector, for our primary tests. After preprocessing, we obtained 15,675 and 3,919 entries in the training and test sets, respectively. However, at this point, we left the test set for the final performance assessment. The models were trained using the Adam [15] optimiser with an adaptable learning rate which was reduced on learning curve plateaus.

As the output of the MCP and regular convolutional models was just a single number (the predicted timestamp), their metric was the squared error between the predicted and ground-truth values. In the case of the UNet-based model, it was trained to output a heatmap, so its error was calculated as the mean squared error between the predicted heatmap and the ground-truth one. The ground-truth heatmap was generated as a Gaussian with the mean at the true timestamp and a small standard deviation of one sampling step, i.e. 156.25 ps, following [25]. The final UNet timestamp could be retrieved as the mean of a Gaussian fitted to the output vector. The training process stopped when no improvements in the loss function were observed, following the early stopping method. This aimed to minimise the impact of overfitting [10].

4.2. Hyperparameter tuning procedure

Our hyperparameter tuning procedure consisted of two steps. First, we used a tuner algorithm to select the most promising models. Then, we performed cross-validation to determine the best one. The entire procedure utilised only the training set, with the same train-validation splits for every network type. We reserved the test set for final performance estimation.

For the first step, we chose one of the most common hyperparameter tuners, KerasTuner [16]. It is capable of selecting the top-N best models given an optimisation algorithm. We chose the Bayesian optimiser, which is an improved version of the grid search and random search algorithms. It estimates the loss function versus the hyperparameter values, and samples the hyperparameter sets according to that distribution. For each network type, we ran 40 iterations of KerasTuner, testing 40 different hyperparameter sets. Each set was trained using 80% of the original training set, while the remaining 20% was used for validation. To improve the quality of the results and filter out unstable models, we used two executions per trial, meaning that the result of a model was calculated as an average of the loss values from these two, separate executions of training and validation.

In the second phase, we used the top 5 models outputted by the tuner and performed 5-fold cross-validation using only the training set. The folds were consistent across all network types, and each fold value was an average of three trials. The final model for a given network type was chosen based on the mean and standard deviation of the cross-validation results.

The hyperparameter tuning procedure was run on the High-Performance Computing GPU cluster. The computations took from one to four hours, depending on the network architecture.

4.3. Multilayer perceptron

The main goal of tuning the multilayer perceptron (MLP) architecture was to find the optimal number of hidden layers and neurons. To avoid using a separate hyperparameter for the number of neurons in each layer, we assumed that either the same

number was used or that the number was divided by 1.5 or 2 for every consecutive dense layer. The final layer always had only one neuron, since it was the output of the model and represented the predicted timestamp.

In addition to these parameters, we also tested the effects of adding batch normalisation [14] and dropout [22]. Batch normalisation could be added either after every dense layer or not at all, and could also be added independently after the input layer. Finally, dropout was set to either 0, 0.2, or 0.5. An activation function was applied after every dense layer, except for the last one, and was fixed to ReLU [13].

The best models returned by the tuning procedure almost always included batch normalisation after every dense layer and the input but did not use dropout. There were no clear patterns for the rest of the hyperparameters. The final MLP architecture is shown in Figure 8.
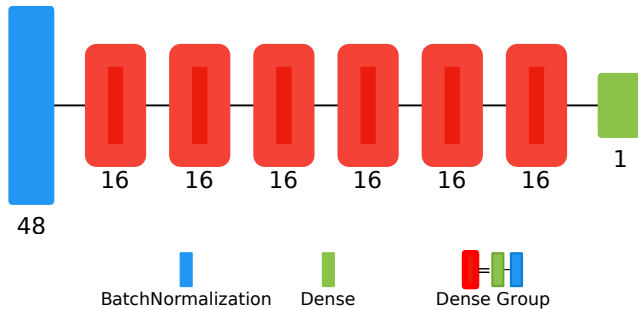

Figure 8. Optimal MLP model [3]

4.4. Convolutional network

The convolutional neural network (CNN) consisted mainly of convolutional layers, which are commonly used in image and time series processing tasks. Unlike dense layers, convolutional layers can learn the relations between neighbours, such as time series samples located next to each other.

Typically, the number of filters increases with each consecutive convolutional layer. In our case, it was multiplied by 2. However, up to three convolutional layers could be used in sequence before increasing the filter count. We refer to this group as a convolutional block. The number of blocks was another hyperparameter, ranging from 1 to 4. Only the number of filters in the first block was optimised, as the numbers in the following blocks could be inferred from the number in the first block. All convolutional layers had kernels of small size: 3. A single dense layer was placed at the end of the network so that the network could output a single number. A small MLP could be inserted between the convolutional part of the network and the final dense layer, parametrised similarly to the full MLP architecture. However, its depth was limited to 3.

In addition to the core layers, batch normalisation was parametrised similarly to the MLP. Dropout could be applied after each dense layer in the MLP part. For convolutional layers, we used spatial dropout [24] instead of regular dropout. Its rate could be set to 0.0, 0.1, or 0.2.

Batch normalisation was used in all of the models returned by the tuner, while the MLP dropout was always set to 0.0. No visible patterns were observed for other hyperparameters. The final convolutional architecture is shown in Figure 9.
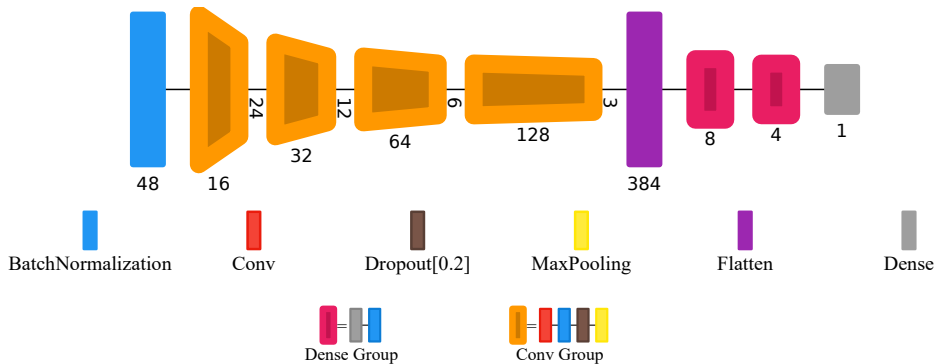

Figure 9. Optimal CNN model [3]

4.5. UNet

The last architecture we used, UNet, is characterised by the U shape of its architecture. It is composed of an encoder and a decoder. The encoder extracts relevant features from the input, while the decoder uses those features to build a vector of the input shape and highlight relevant spots, such as predicted timestamps in time series processing. The UNet architecture takes advantage of skip connections to amplify the importance of initial features in the decoder. Thanks to the segmentation and noise reduction capabilities of UNet, we expected it to be a good candidate for our task.

The main hyperparameter for our model of the network depth, measured in UNet blocks. The encoder and decoder were symmetrical and contained the same number of blocks. An encoder block consisted of one to three convolutional layers followed by a max pooling layer. A decoder block started with deconvolution, which we implemented through upsampling and a convolutional layer with a kernel size of one (all other convolutional layers had a kernel size of 3). The output of a decoder block was concatenated with the output from the corresponding block in the encoder through a skip connection. Finally, one to three convolutional layers were used. As before, batch normalisation and spatial dropout could be added after every convolutional layer with a kernel size of 3. Batch normalisation could also be used after input. The final UNet architecture is depicted in Figure 10.
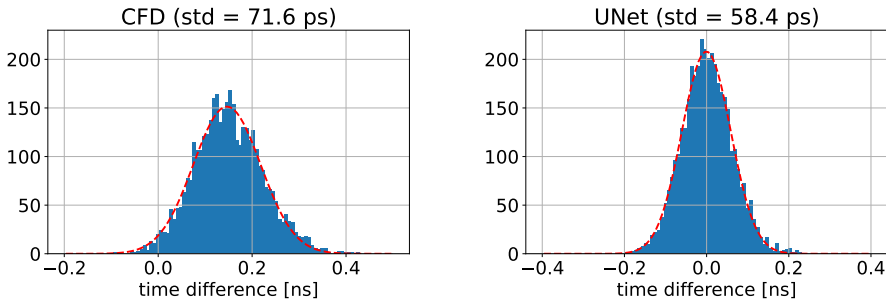
Figure 10. Optimal UNet model [3]

5. Results

This section begins with a description of the method used to assess the timing precision of either the CFD or a deep neural network. Then, we compare the results obtained with the CFD to those obtained with the best neural network using data from a single readout channel of the detector. We also provide a description of other performance tests we performed, including the tests on other readout channels.

5.1. Precision assessment method

To measure the time precision of a detector, a typical method is to compare its measurements with those of a 'reference' detector that has much better time precision. The reference detector is placed on the same beam line to detect the same particles. The mean of the differences between the tested detector and the reference detector represents a constant offset. Although neural networks can learn to have a mean close to zero, the CFD mean is expected to be shifted due to the method's inability to adjust to inconsistent signal characteristics. The precision of the time measurement is represented by the standard deviation of the differences.

In our case, MCP served the role of the reference detector. What is more, to reduce tail effects, we fitted a Gaussian curve to the histogram of the time differences and used the standard deviation of the Gaussian as the precision measure (as shown in Figure 11).

Figure 11. Difference histograms for the CFD and our best-performing, UNet-based model

5.2. The optimal architecture

We performed cross-validation on the best models, one from each network type. Instead of computing loss values, we used the evaluation method described above to compute the results. The computations were performed only on the training portion of the dataset. The results are shown in Table 1. As expected, UNet had the smallest (the best) average precision value. It was also the most complex model with the biggest number of parameters. Interestingly, the model had more parameters than the number of training samples. This is typical for neural networks as they often use more parameters than the minimum required number. While this can make the network susceptible to overfitting, with proper training, overfitting can be avoided. The early stopping method we employed is the best approach to address this issue. Additionally, spatial dropout was applied to improve performance at the expense of further increasing the number of parameters. Surprisingly, the MLP model was the most stable, with the smallest standard deviation of results for each fold.

Table 1

Comparison of the precisions achieved by the optimal models in the cross-validation procedure. In addition to the cross-validation scores, the number of parameters used by each network is reported, too

Architecture	Mean [ps]	Std [ps]	Parameter count
MLP	63.90	0.85	2737
CNN	62.83	1.34	36,865
UNet	60.71	1.19	456,965

To evaluate the final performance of our solution, we used the test set which was composed of data from the same readout channel as the training set. Figure 11 shows the difference histograms for the CFD and our best-performing neural network. The network's histogram is visibly narrower, indicating better precision.

5.3. Adjusting the data to the LHC conditions

Due to limited available bandwidth in the PPS setup at the LHC, the SAMPIC time series consist of only 24 samples. To validate the networks under these conditions, we trimmed the original time series from 64 to 24 samples. First, we smeared the timestamp distribution by randomly removing up to 10 samples from the beginning of the series, retaining only the next 56 samples. We then selected the 24 samples from the middle to ensure that the most important part of the signal was preserved. We made slight adjustments to the network architectures to accommodate the smaller input size (24 instead of 48). We performed the same tests as before and obtained similar results. We were able to improve the precision from $73.3\,\mathrm{ps}$ to $62.1\,\mathrm{ps}$ standing for 15%, which is promising for using deep learning with LHC data.

5.4. Tests on many channels

In the previous sections, we only used data from a single detector channel. However, we also tested data from other channels. We first trained the optimal network on each channel separately and then tested it on the same channel which was used for training. We achieved precision improvements ranging from 8% to 23% compared to the CFD. We also investigated if the network could be trained on one channel and tested on another, or even trained on all available data while maintaining the train-test split. We present the results for selected, representative channels in Table 2.

Table 2
Precision improvements with respect to the CFD obtained with many detector channels involved. We selected seven, representative channels and highlighted the best precision for each one

Training channel	Test channel						
	10	16	17	22	25	27	31
10	**13%**	10%	13%	7%	-1%	-23%	-7%
16	6%	**23%**	16%	9%	-22%	-9%	3%
17	7%	17%	**19%**	9%	-3%	8%	-6%
22	4%	14%	-4%	11%	-84%	-51%	7%
25	4%	4%	7%	4%	**12%**	8%	-4%
27	-13%	-10%	4%	-16%	4%	**19%**	-17%
31	2%	13%	10%	7%	-7%	-7	**16%**
all	8%	22%	14%	**12%**	9%	17%	14%

These results show that, typically, in order to achieve the highest precision for a given channel, the network needs to be trained using data from that channel specifically. If trained on one channel and tested on another, the network might perform worse than the CFD, resulting in negative entries in Table 2. This shows that even though the same sensor is used, the collected data differ significantly between channels. Channel 22 is the only channel for which the network trained on all channels

was able to achieve slightly better precision than the network trained on that specific channel. This may be due to an unfavourable train-test split for the channel 22 data. We present the improvements for all the channels we explored using the networks trained on the particular channels in Figure 12. The data were available for channels from 8 to 31. We do not report the improvements for channels 12 and 15, as the waveforms were too noisy.
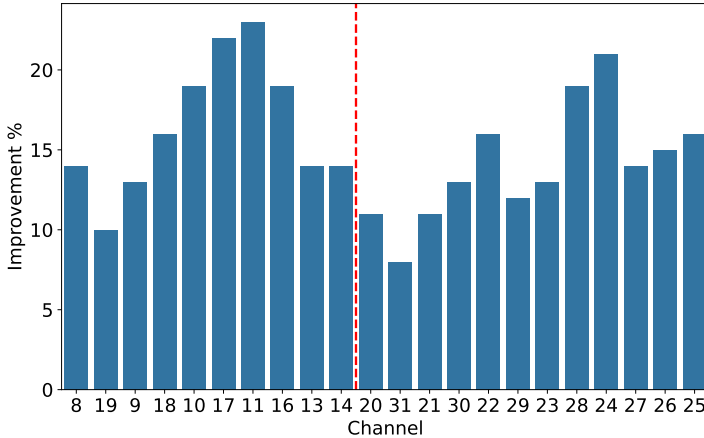

Figure 12. Improvements with respect to the CFD for the channels we explored. The channel order follows the physical set-up of the test beam experiment. The red dashed line divides the channels from two separate planes of diamond sensors used in the experiment

It is worth noting that we did not test the network on data from all available channels at once. This would be difficult due to the different mean values of the difference histograms in various channels. Instead of a single Gaussian, we would have a group of smeared Gaussians, which would make it impossible to retrieve the true detector precision. Therefore, even when the network was trained using data from all channels, we tested it separately for each channel.

6. Conclusion

We demonstrated that deep neural networks can be used to compute the time of arrival of particles taking samples voltage signals at input. In fact, these networks can improve timing precision compared to the most commonly used algorithm, the CFD. In the base numerical experiment, we were able to improve precision by 17%. It is a significant value considering that we did not make any modifications the detector setup, but just used a different algorithm for computing the time of arrival. Other readout channels also showed improvements ranging from 8% to 23%. We found that networks based on the UNet architecture yield the best results among the models we investigated. However, we did not test recurrent networks, which are also expected to

perform well in this kind of problem. We leave that for further research. Nonetheless, even the simplest network architecture we tested – MLP – enabled us to calculate the arrival time with noticeably better precision than the CFD.

Neural networks have a wide range of applications in high energy physics. For example, some types of neural networks are used to evaluate the quality of waveform-like experimental data, thereby improving the detection of outliers and bad data [2]. Additionally, the resolution of pattern recognition algorithms is improved for detectors with complex geometries [19]. This work contributes to the evaluation of neural networks in high-energy physics. The method proposed in this manuscript broadens their applicability and increases the precision of timing detectors. Deep neural networks have been tested in similar applications. It has been shown that they can be used in the prediction of arrival time [26, 27], or more generally in the annotation of time series [20]. This proves that our findings are not just a coincidence. However, it is worth noting that neural networks can achieve high accuracy only on data similar to the training dataset.

The method described in [26] differs from our approach in terms of the detector architecture and the usage of simulated data in network training. In contrast, our work relies entirely on experimental data and a more precise source of reference data in the form of an MCP-PMT detector.

It is important to note that our reference data were not flawless. The MCP precision was assessed to be 10 ps, which is much better than that of diamond sensors (about 50–100 ps). Nevertheless, MCP signals are not perfect and introduce a small degree of uncertainty. Using the CFD further worsens the reference precision. The resulting error is random and can cause some events to contradict each other during neural network training. For instance, reference timestamps may vary for waveforms that look identical. As a result, the final precision of the neural network was negatively impacted.

In addition to the time of arrival study, the procedure developed in this research using KerasTuner to tune hyperparameters and find the optimal network architecture has been successfully applied in ongoing studies to improve the precision of timing computations in the CMS-PPS subsystem at the LHC.

The Large Hadron Collider (LHC) restarted in 2022 and is producing data in a format similar to that discussed in this article. The SAMPIC readout board saves the full waveform in the raw data stream, which can be subject to further analysis using the method presented in this manuscript. Although the CFD is still commonly used for timing measurements, it may be replaced by neural networks. Lack of the MCP in the LHC setup poses a problem in terms of the reference data acquisition, but the work on finding another way is ongoing.

The inference from the neural network has a very low demand for the CPU time (order of 10 milliseconds), making it well-suited for online processing, such as in the high-level trigger reconstruction chain. The data can be processed in batches, enabling efficient parallel processing of a large number of events.

Acknowledgements

References

[1] Albrow M., Arneodo M., Avati V., Baechler J., Cartiglia N., Deile M., Gallinaro M., *et al.*: CMS-TOTEM Precision Proton Spectrometer, Technical design report CMS:13, Technical design report TOTEM:3 13, 2014. https://cds.cern.ch/record/1753795/.

[2] Azzolin V., Andrews M., Cerminara G., Dev N., Jessop C., Marinelli N., Mudholkar T., *et al.*: Improving data quality monitoring via a partnership of technologies and resources between the cms experiment at cern and industry. In: *EPJ Web of Conferences*, vol. 214, 01007, 2019. doi: 10.1051/epjconf/201921401007.

[3] Bäuerle A., Onzenoodt van C., Ropinski T.: Net2Vis – A Visual Grammar for Automatically Generating Publication-Tailored CNN Architecture Visualizations, *IEEE Transactions on Visualization and Computer Graphics*, vol. 27(6), pp. 2980–2991, 2021. doi: 10.1109/TVCG.2021.3057483.

[4] Berg E., Cherry S.R.: Using convolutional neural networks to estimate time-of-flight from PET detector waveforms, *Physics in Medicine & Biology*, vol. 63(2), 02LT01, 2018.

[5] Berretti M., Bossini E., Minafra N.: Timing performances of diamond detectors with Charge Sensitive Amplifier readout, Technical report, CERN-TOTEM-NOTE-2015-003, 2015. https://cds.cern.ch/record/2055747.

[6] Bossini E.: *Development of a Time Of Flight diamond detector and readout system for the TOTEM experiment at CERN*, Ph.D. thesis, INFN, Siena, 2016. https://cds.cern.ch/record/2227688. CERN-THESIS-2016-137.

[7] Bossini E.: The CMS Precision Proton Spectrometer timing system: performance in Run 2, future upgrades and sensor radiation hardness studies, *Journal of Instrumentation*, vol. 15(05), C05054, 2020. doi: 10.1088/1748-0221/15/05/C05054.

[8] Bossini E., Figueiredo D.M., Forthomme L.M., Garcia Fuentes F.I.: Test beam results of irradiated single-crystal CVD diamond detectors at DESY-II, Technical reports, CMS-NOTE-2020-007, CERN-CMS-NOTE-2020-007, 2020.

[9] Breton D., De Cacqueray V., Delagnes E., Grabas H., Maalmi J., Minafra N., Royon C., Saimpert M.: Measurements of timing resolution of ultra-fast silicon detectors with the SAMPIC waveform digitizer, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 835, pp. 51–60, 2016. doi: 10.1016/j.nima.2016.08.019.

[10] Caruana R., Lawrence S., Giles C.: Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping, *Advances in Neural Information Processing Systems*, vol. 13, 2000.

[11] Delagnes E., Breton D., Grabas H., Maalmi J., Rusquart P., Saimpert M.: The SAMPIC waveform and time to digital converter. In: *2014 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, pp. 1–9, IEEE, 2014. doi: 10.1109/nssmic.2014.7431231.

[12] Evans L., Bryant P.: LHC machine, *Journal of Instrumentation*, vol. 3(08), S08001, 2008. doi: 10.1088/1748-0221/3/08/S08001.

[13] Glorot X., Bordes A., Bengio Y.: Deep sparse rectifier neural networks. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTAT)*, pp. 315–323, JMLR Workshop and Conference Proceedings, 2011.

[14] Ioffe S., Szegedy C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pp. 448–456, 2015.

[15] Kingma D.P., Ba J.: Adam: A method for stochastic optimization, *arXiv preprint arXiv:14126980*, 2014. doi: 10.48550/arXiv.1412.6980.

[16] O'Malley T., Bursztein E., Long J., Chollet F., Jin H., Invernizzi L., *et al.*: KerasTuner, https://github.com/keras-team/keras-tuner, 2019.

[17] Ronneberger O., Fischer P., Brox T.: U-net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015. doi: 10.1007/978-3-319-24574-4_28.

[18] Schefer M.M.: Machine Learning Techniques for selecting Forward Electrons $(2.5 < |\eta| < 3.2)$ with the ATLAS High Level Trigger, Technical report, ATL-DAQ-PROC-2023-001, 2023. https://cds.cern.ch/record/2851302.

[19] Shlomi J., Battaglia P., Vlimant J.R.: Graph neural networks in particle physics, *Machine Learning: Science and Technology*, vol. 2(2), 021001, 2020. doi: 10.1088/2632-2153/abbf9a.

[20] Sodmann P., Vollmer M., Nath N., Kaderali L.: A convolutional neural network for ECG annotation as the basis for classification of cardiac rhythms, *Physiological Measurement*, vol. 39(10), 104005, 2018.

[21] Song X., Liu Y., Xue L., Wang J., Zhang J., Wang J., Jiang L., Cheng Z.: Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model, *Journal of Petroleum Science and Engineering*, vol. 186, 106682, 2020.

[22] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R.: Dropout: a simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research*, vol. 15(1), pp. 1929–1958, 2014.

[23] The CMS Collaboration *et al*, *Journal of Instrumentation*, vol. 3, S08004, 2008. doi: 10.1088/1748-0221/3/08/S08004.

[24] Tompson J., Goroshin R., Jain A., LeCun Y., Bregler C.: Efficient object localization using convolutional networks. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 648–656, 2015. doi: 10.1109/cvpr.2015.7298664.

[25] Tompson J.J., Jain A., LeCun Y., Bregler C.: Joint training of a convolutional network and a graphical model for human pose estimation, *Advances in Neural Information Processing Systems*, vol. 27, 2014. doi: 10.48550/arXiv.1406.2984.

[26] Wang F., Han D., Wang Y.: Improving the time resolution of the MRPC detector using deep-learning algorithms, *Journal of Instrumentation*, vol. 15(09), C09033, 2020. doi: 10.1088/1748-0221/15/09/C09033.

[27] Wang F., Han D., Wang Y., Yu Y., Lyu P., Guo B.: The study of a new time reconstruction method for MRPC read out by waveform digitizer, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 954, 161224, 2020. doi: 10.1016/j.nima.2018.09.059.

[28] Zahid M.U., Kiranyaz S., Ince T., Devecioglu O.C., Chowdhury M.E., Khandakar A., Tahir A., Gabbouj M.: Robust R-Peak Detection in Low-Quality Holter ECGs Using 1D Convolutional Neural Network, *IEEE Transactions on Biomedical Engineering*, vol. 69(1), pp. 119–128, 2022. doi: 10.1109/tbme.2021.3088218.

Affiliations

Mateusz Kocot
AGH University of Krakow, Krakow, Poland, mateusz.kocot@cern.ch

Krzysztof Misan
AGH University of Krakow, Krakow, Poland, krzysztof.misan@cern.ch

Valentina Avati
AGH University of Krakow, Krakow, Poland, valentina.avati@cern.ch

Edoardo Bossini
INFN Sezione di Pisa, Pisa, Italy, edoardo.bossini@pi.infn.it

Leszek Grzanka
AGH University of Krakow, Krakow, Poland, grzanka@agh.edu.pl

Nicola Minafra
University of Kansas, Department of Physics and Astronomy, Lawrence, KS, United States, nicola.minafra@cern.ch