ISBN: 978-3-95450-241-7 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2022-WEP42

# APPLICATION OF MACHINE LEARNING TOWARDS PARTICLE COUNTING AND IDENTIFICATION

S. Engel[1,*] R. Singh[2,†], P. Boutachkov[2]
[1] University of Essex, Colchester, England
[2] GSI Helmholtz Centre for Heavy Ion Research, Darmstadt, Germany

*Abstract*

An exploration into the application of three machine learning (ML) approaches to identify and separate events in the detectors used for particle counting at the GSI Helmholtz Centre for Heavy Ion Research was performed. A shape-based template matching algorithm (STMF), Peak Property-based Counting Algorithm (PPCA) and convolutional neural network (CNN) were tested for counting the number of particles accurately without domain-specific knowledge required to run the currently used algorithm. The three domain-agnostic ML algorithms are based on data from scintillation counters commonly used in beam instrumentation and represent proof-of-principle for an automated particle counting system. The algorithms were trained on a labelled set of over 150 000 experimental particle data. The results of the three classification approaches were compared to find a solution that best mitigates the effects of particle pile-ups. The two best-achieving algorithms were PPCA and CNN, achieving an accuracy of over 99%.

## INTRODUCTION

As charged particle beams pass through a scintillation counter, it induces the excitations in the scintillation material which relaxed with emission of photons. These photons are amplified using photo multiplier tubes (PMT) which outputs a voltage pulse measured over time. The shortest pulse lengths are typically of the order of few ns given by the scintillation process itself. The transportation of these pulses from the radiation environment to the detection electronics is about 50-100 m, which further increases the pulse length due to cable dispersion. As the rate of particles arriving at the scintillator increases so does the probability for signal pile-ups, a overlap of sequential pulses. Due to complex slow extraction process from the particle accelerators and sub Poissonion particle distribution [1], non-negligible percentage of signal pile-ups occur at most particle arrival rates. These complex, hard-to-count, peak shapes change in a non-linear fashion. The interference causes difficulties in using conventional computing methods to accurately count particles, although strategies to mitigate them using hardware [2] and software solutions [3, 4] have been explored. Notably several ML approaches have been developed with some success by relying on domain-specific knowledge and feature engineering specific to the detector [3–5].

Other more general ML approaches rely on 2D data and/or data with multiple features [6], demonstrating domain-specific knowledge is not necessary to provide satisfactory results. Specifically, CNN's prove to be the ideal conventional ML solution because of their robustness and the ability to create a domain agnostic, end-to-end discriminating model which eliminates unnecessary domain-specific preprocessing [7].

## DATA COLLECTION AND TRANSFORMATION

The major portion of the training and testing data consisted of high-resolution experimental data of $1.5 \times 10^5$ peaks collected at the SIS18 synchrotron at GSI with a sampling rate of 2.5 GSa/s. The data was labelled by fine-tuning the present particle counting algorithm. Therefore, only labeled data with a low extraction rate of up to $3 \times 10^5$ particles per second could be used.

Low-resolution data was generated by downsampling the high-resolution data collected at $(2.5 \times 10^9)$ samples per second as seen in Fig. 1. Data was downsampled by a factor of $s$ to test the accuracy at various resolutions.

The high-resolution data was bootstrapped by combining an offset copy of a time series with the original to create more complex shapes representative of data with a higher rate of particles.

The validation data of 416 peaks was experimentally collected by firing a laser at the scintillation counter at precisely set intervals to selectively generate various pile-up shapes.

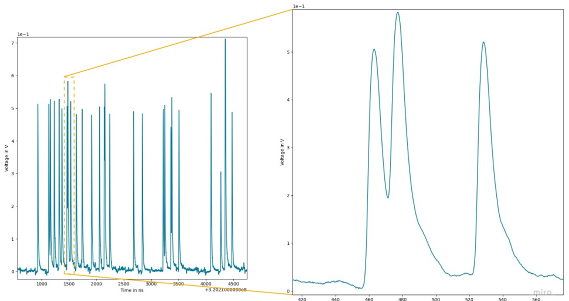
Figure 1: Graph showing a section of a time series with a close-up of a peak

## TIME SERIES CLASSIFICATION USING SUPERVISED MACHINE LEARNING

1. Shape-based Template Matching Framework (STMF)
2. Peak Property-based Counting Algorithm (PPCA)
3. Convolutional neural network (CNN)

* samuel-engel@outlook.com
† r.singh@gsi.de

### *Shape-based Template Matching Framework*

The shape-based template matching framework is an implementation based on a study exploring efficient matching of templates with sections of time series [8]. The training data time series is segmented into peak groups of $1, 2, 3 \ldots$ particles and a template is constructed for each one. A time series segment is classified by comparing it to every template to determine the best fit. The approach mimics the way humans approach the problem.

The templates were constructed using an averaging scheme which determines how each labelled peak group time series segment is combined. The cubic-spline dynamic time warping (CDTW) averaging function was iteratively applied to the two most similar time series. The averaged time series replaces the two original ones. Eventually, the result is a single time series representative of the peak group which became the template for that peak group.

An implementation of the dynamic time warping (DTW) algorithm was used to determine the similarity between a given peak group time series and every generated template. Dynamic time warping is a method of comparing two time series of unequal lengths and finding their similarity as demonstrated in Fig. 2. It warps the time series to match points regardless of shifts or distortions in time, creating a warping path [9]. A DTW warping path is a map matching each corresponding point in the two time series where the matching can be one-to-one, one-to-many or many-to-one due to the time series having unequal lengths [10].

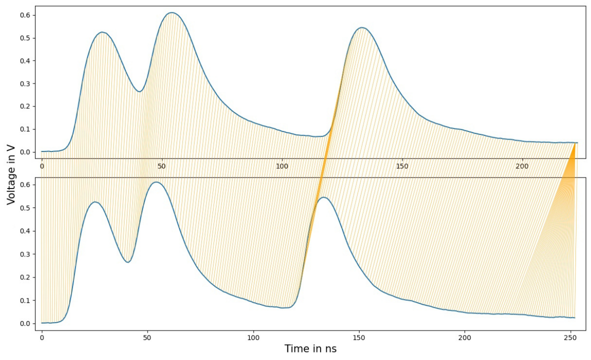

Figure 2: Graph showing a DTW mapping between two 3 peak time series segments.

### *Peak Property-based Counting Algorithm*

The peak property based counting algorithm (PPCA) works on the principle of peaks having certain mathematical properties which define them, namely the peak width, height, prominence and the distance between peaks. The algorithm uses labelled data to find the optimal attribute ranges such that all peaks are counted correctly. Establishing the peak properties of the labelled peaks allows the algorithm to learn the mathematical property ranges which contain all the peaks being counted.

The algorithm works with segments of arbitrary length given every peak in the segment is present in its entirety in order to calculate all the properties. Prominence $P$ is calculated as the vertical distance to the peak's lowest contour line $PC_l$. The peak's two contour lines are determined by the segment between the nearest peak on either side. The minimum point of each of the two segments are the peak bases. The highest peak base defines the lowest contour line $PC_l$. The height of a peak is the $y$ value of the data point in the peak's centre. The peak width is determined at an evaluation height $h_{\text{eval}}$ which is calculated first using $h_{\text{eval}} = h_{\text{peak}} - P \cdot 0.5$.

Half of the prominence $P$ of a peak is subtracted from the height of the peak $h_{\text{peak}}$ to find the height at which the width is measured. Half of the prominence is used to achieve the most accurate results across a variety of peak shapes. The width is calculated by getting an intersecting point with the data at the evaluation height on both sides. The distance between peaks is calculated as the horizontal distance between neighbouring peak centres. Lastly, a threshold is set to be just above the baseline in order to avoid any false positives arising from the noise in the baseline.

The training of the model weights relied on optimizing the parameters of the peak finding function. The labels of the training data contain the time at the centre of every peak. The first stage of the training involves extracting the properties of all labelled peaks which are visualized in Fig. 3.

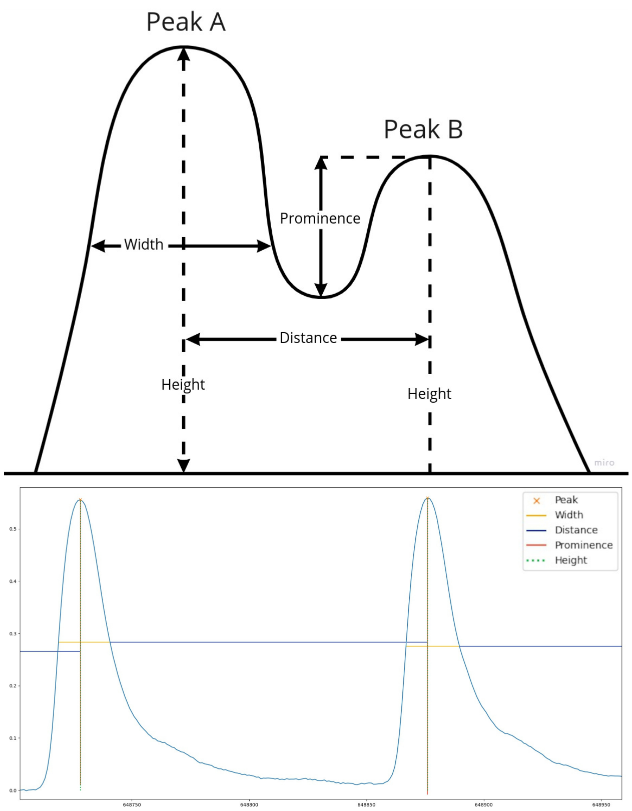

Figure 3: A visualization of the mathematical peak properties captured.

All the relevant peak data allows for several statistical methods to be applied to establish trends and find the most optimal combination of parameters for finding the peak's lower and upper bound respectively. The number of standard deviations the minimum and maximum are from the mean is used as a measure to ensure an outlier is not causing distortions in order to get a clean set of weights representing peak property bounds. Finally the peak finding function is run on the test data set with the weights as the parameters. Comparing the results with the labels gives the loss function consisting of the sum of false positives and negatives. This step is repeated iteratively while progressively changing the weights, one at a time. The weights are optimized in an order of importance determined by the peak finding function. The optimization order starts with the peak height followed by the threshold, distance, prominence and lastly the width.

The algorithm uses the weights learned in the training step to identify distinct peaks representing particles. Each weight represents the lower or upper bound of one of the parameters. The resulting combination of lower and upper bounds defines ranges into which all particle peaks fall. The order in which the properties are evaluated is important in terms of performance. The goal is to eliminate the greatest number of particle peaks which fall outside of the bounds using the least computationally demanding property evaluation method. That way, the number of potential particle peaks is reduced and only a fraction has to be evaluated using the more computationally demanding property evaluation methods.

### *Convolutional Neural Network*

The one-dimensional convolutional neural network (CNN) uses discriminative supervised learning to count particles. It learns features from raw time series data rather than through engineered features such as the peak shape or properties used for the previous two approaches. The input data is segmented into even-sized windows before the model is trained by learning the intrinsic features.

Every segment is labelled with the number of peak centres it contains meaning at least more than half of each peak's data points needs to be inside the window for it to be counted. Windows with only the baseline tend to vary very little from one another and their over-representation creates a redundancy which confuses the feature learning aspect of the CNN. A pruning technique had to be adapted to reduce redundant baseline data and decrease the time complexity of the algorithm. The technique settled on is based on separating windows labelled with 0 peaks into two categories. Baseline-only windows and windows with a fraction of a peak with less than 50% of the peak's data points. Just the baseline-only windows contain redundant information whereas the fractions of peaks are all different and contain key information. The baseline-only windows are reduced by a pruning factor which determines what proportion of baseline-only windows is kept.

The CNN model is developed using the Keras deep learning library which requires a 3-dimensional input for all its models. The design of the sequential model can be seen in Fig. 4. The input layer is fed data in the form of samples, time steps and features form the 3-dimensional input

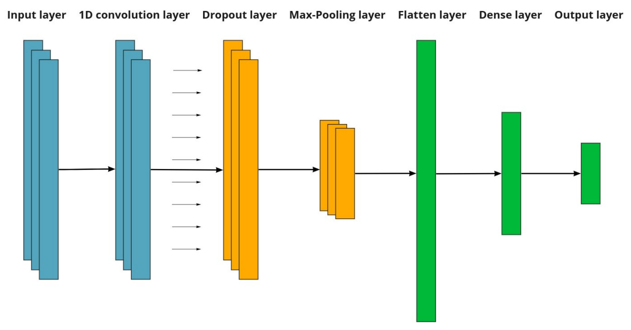

Figure 4: Diagram showing layers of the CNN model

required. The samples are the windows from the data segmentation step. Each sample or window has $S$ time steps which is one of the hyper-parameters of the model. The last dimension is the features although in this case, the only feature available was the voltage. The lack of features was one of the main challenges with managing the deep learning approach and caused difficulties in increasing the model's accuracy without overfitting the data. The model starts with a pair of two 1D convolutional layers, the first one being the input layer. The dropout layer is used for regularization by ignoring some of the features of the two convolutional layers to limit overfitting.

The max-pooling layer reduces the size of the learned features by $\frac{1}{4}$ by summarizing the features to the most essential ones. All of the features are then flattened into a large 1d vector before it is passed to the fully-connected layer towards the end of the sequential Keras model. The fully connected layer is a dense layer which serves as a buffer to consolidate the learned features before prediction takes place. The last layer is the output layer which is also a dense layer. However, its job is to simply make a prediction based on the learned features passed down to it from the previous layers. A window can be fed directly into the model in a stream to enable real-time data classification. The model outputs a list of probabilities of the window containing 1 peak, 2 peaks etc. based on the learned features.

## RESULTS & PERFORMANCE

The exploratory project is a proof-of-work showing machine learning can be used to successfully count particles. Such accuracy was achieved using a domain-agnostic approach which only takes into account the time series data and no specific information about the experiment or the instruments used. The comparison in terms of accuracy can be found in Table 1. However, assessing the accuracy of the 3 solutions solely based on the performance from the available data shows only one side of the picture. Considering the strengths and weaknesses of the three approaches allows predicting how the accuracy would change under different conditions. The PPCA not only has the highest accuracy out of the three algorithms but it is also strong in simplicity and observability. All of these qualities are highly desirable when it comes to conducting scientific analysis. However, the lack of complexity comes with a downside of a lack of robustness. It is likely the PPCA is the least adaptable solution to varying conditions or new trends in the data. Specifically, a higher rate of particles producing more complex shapes could result in peaks that physically cannot be described using parameter limits. On the other hand, the CNN is specifically designed to learn intrinsic features in the data so as to allow it to adapt to different trends in the data. The CNN design is much more adaptable and has the fewest limits out of the 3 approaches in terms of how complex the peak shapes can be. The PPCA seems to be the most suitable approach for the data used in this contribution. However given that the difference in CNN performance compared to the PPCA is minimal, and with higher adaptability of CNN, we believe it could be a promising approach with wider variety of data. Lastly, the STMF suffers from the same weaknesses as the PPCA in terms of being limited by the complexity of the peak shapes. The more peaks there are in a segment, the less representative a template becomes meaning the number of segments without a match increases as well. Adapting the STMF to train multiple templates per peak label could address this issue but the increased complexity would not be worth the minimal improvement.

Table 1: Accuracy and Space Performance of the 3 Algorithms

| | **Accuracy (%)** | **Storage Requirements** |
|---|---|---|
| PPCA | 99.97 | <1 kB |
| STMF | 96.71 | <20 kB |
| CNN | 99.84 | <1 MB |

Table 2: Time Performance for Classification

| | **Average time in seconds** | |
|---|---|---|
| | **Classify 1 peak** | **Classify 1000 peaks** |
| PPCA | 0.00023 | 0.23 |
| STMF | 0.024 | 24.5 |
| CNN | 0.063 | 62.7 |

## FUTURE WORK

The limitations of the labelled data restricted the algorithms to only be trained and tested at a low extraction rate up to $3 \times 10^5$ particles per second. Therefore, further development should focus on improving the chosen algorithm to enable classification at higher extraction rates without major losses in accuracy. The use of convolutional neural network has proven to be the best performing algorithm considering the combination of accuracy, adaptability and implementability. Therefore, a more in depth research into the application of a CNN for particle counting should be conducted. Exploring the ways in which the algorithm can be implemented as a part of the detector using an FPGA or a similar approach would also be a very productive area of future research.

## ACKNOWLEDGEMENT

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under GA No 101004730.

## REFERENCES

[1] R. Singh, P. Forck, and S. Sorge, "Reducing fluctuations in slow-extraction beam spill using transit-time-dependent tune modulation", *Phys. Rev. Applied*, vol. 13, p. 044 076, 4 2020. doi:10.1103/PhysRevApplied.13.044076

[2] W. H. Wong and H. Li, "A scintillation detector signal processing technique with active pileup prevention for extending scintillation count rates", *IEEE Transactions on Nuclear Science*, vol. 45, no. 3, pp. 838–842, 1998. doi:10.1109/23.682647

[3] P. J. Statham, "Pile-Up Correction for Improved Accuracy and Speed of X-Ray Analysis", *Microchimica Acta*, vol. 155, no. 1, pp. 289–294, 2006. doi:10.1007/s00604-006-0558-1

[4] D. J. R. Sevilla, "A Simple Pile-up Model for Time Series Analysis", vol. 843, no. 1, p. 44, 2017. doi:10.3847/1538-4357/aa72e8

[5] D. Bertolini, P. Harris, M. Low, and N. Tran, "Pileup Per Particle Identification", *Journal of High Energy Physics*, vol. 2014, no. 10, p. 59, 2014. doi:10.1007/JHEP10(2014)059

[6] P. Baldi, K. Bauer, C. Eng, P. Sadowski, and D. Whiteson, "Jet Substructure Classification in High-Energy Physics with Deep Neural Networks", *Physical Review D*, vol. 93, no. 9, p. 094 034, 27, 2016. doi:10.1103/PhysRevD.93.094034

[7] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: A review", 2018. doi:10.1007/s10618-019-00619-1

[8] V. Niennattrakul and C. A. Ratanamahatana, "Inaccuracies of Shape Averaging Method Using Dynamic Time Warping for Time Series Data", pp. 513–520, Proceedings of the 7th International Conference on Computational Science (ICCS'07).

[9] P. Senin, "Dynamic Time Warping Algorithm Review", pp. 1–23, 2008.

[10] C.-J. Hsu, K.-S. Huang, C.-B. Yang, and Y.-P. Guo, "Flexible Dynamic Time Warping for Time Series Classification", *Procedia Computer Science*, vol. 51, pp. 2838–2842, 1, 2015. doi:10.1016/j.procs.2015.05.444