Data Analysis for Atomic Shapes in Nuclear Science

Mehmet Cagri Kaymak[1], Hasan Metin Aktulga[1], Ron Fox[2], and Sean N. Liddick[2,3]

[1] Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA
[2] National Superconducting Cyclotron Laboratory, Michigan State University, East Lansing, MI 48824, USA
[3] Department of Chemistry, Michigan State University, East Lansing, MI 48824, USA

Abstract. We consider the problem of detecting a unique experimental signature in time-series data recorded in nuclear physics experiments aimed at understanding the shape of atomic nuclei. The current method involves fitting each sample in the dataset to a given parameterized model function. However, this procedure is computationally expensive due to the nature of the nonlinear curve fitting problem. Since data is skewed towards non-unique signatures, we offer a way to filter out the majority of the uninteresting samples from the dataset by using machine learning methods. By doing so, we decrease the computational costs for detection of the unique experimental signatures in the time-series data. Also, we present a way to generate synthetic training data by estimating the distribution of the underlying parameters of the model function with Kernel Density Estimation. The new workflow that leverages machine learned classifiers trained on the synthetic data are shown to significantly outperform the current procedures used in actual datasets.

Keywords: Machine Learning · Density Estimation · Nuclear Physics.

1 Introduction

Although many people assume that the atomic nucleus adopts a spherical shape, the actual nuclear shapes could vary [10]. Deformed nuclei are quite common at certain regions of proton and neutron numbers. Further, some regions of proton and neutron number are categorized as exhibiting signs of shape-coexistence phenomena in which different nuclear states can be interpreted as having different mean square charge radii. A key experimental indicator of the phenomena is the presence of strong transitions between states with identical angular momentum and parity. Such transitions between spin 0 states proceed through the emission of an atomic electron and when the excited spin-0 state is at a low enough energy, the state will live for a longer period of time before decaying. We describe the workflow and associated challenges through a recent experiment to explore the spin-0 to spin-0 transitions observed in the mass 32 region of the nuclear chart.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_16

The experiment starts with a beam of rare isotopes produced at the National Superconducting Cyclotron Laboratory (NSCL), which in this case is a beam containing $_{32,33}$Na isotopes and numerous neighboring mass nuclei. The $_{32,33}$Na nuclei moving at relativistic speeds are brought to rest inside a large volume active detector. Since the $_{32,33}$Na isotopes are radioactive, they will eventually "beta decay" (a process which changes a neutron into a proton) into an isotope of $_{32,33}$Mg. In rare cases, the decay will leave the resulting isotope with an excess of energy which it will shed within a few hundred nanoseconds through the emission of an energetic atomic electron from the spin-0 to spin-0 transition.

The rare isotopes are brought to rest in a large volume, unsegmented $CeBr_3$ detector. The unsegmented $CeBr_3$ scintillator is readout using a Hamamatsu 13700 position sensitive photomultiplier tube (PSPMT) which is segmented into an array of 16 x 16 pixels for a total of 256 pixels. Each pixel and a signal corresponding to the sum over the entire detector is connected to individual digitizing electronics channels running between 250 and 500 MSPS. An onboard firmware controls when data will be recorded to the onboard memory. For these experiments, information related to the energy of the signal includes time, the channel that recorded it, and a waveform which records the history of the detector voltage as a function of time. An example of a recorded waveform (*i.e.*, time-series data of detector voltage) from the detector is shown in Fig. 1. What is sought is essentially a unique signature, *i.e.*, double pulses (to the right of the figure corresponding to the science case in which the first pulse represents the beta-decay electron and the second pulse represents the transition of interest), which are extremely rare in the presence of other "uninteresting" signal shapes, *i.e.*, single pulses (to the left).

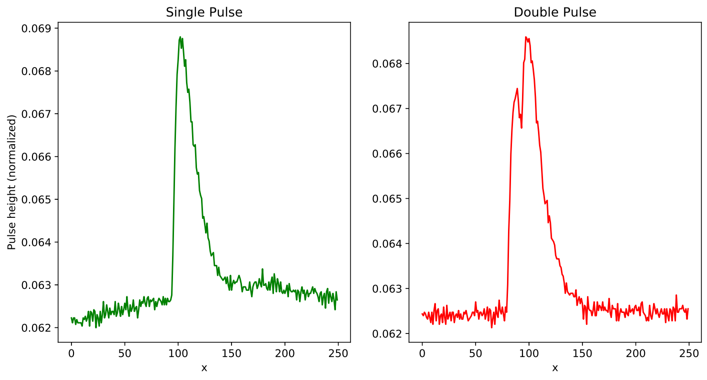

Fig. 1. Different pulses from the dataset.

Whenever a double pulse is detected, the amplitude of both pulses and the time difference between them must be calculated for further physical analysis, which is accomplished by fitting a known functional form to the observed signal, called `lmfit`, as explained in further detail below. The `lmfit` procedure is computationally demanding, and in the absence of a better alternative, it is used for detecting the double pulses among the plethora of uninteresting signals.

Consequently, the existing practice cannot enable real-time analysis of the vast number of traces obtained during experiments. This is highly problematic because only a single experimental session can typically be secured at NSCL per year, and during an experiment scientists need to be able to determine whether they have been able to capture a sufficient number of the rare double pulse signals. It should be noted that since each signal is independent, their processing can easily be parallelized across a number of processors which would potentially ensure real-time processing guarantees with moderate resource investments, but NSCL is soon to be upgraded to the Facility for Rare Isotope Beams (FRIB), which will increase the data collection rates by two to three orders of magnitude. As such, it is critical to have a highly accurate yet inexpensive computational tool to detect double pulses, and this constitutes our main goal in this paper.

In what follows, we first briefly describe the time-series dataset from nuclear physics experiments in Section 2. The proposed machine learning based framework is presented in Section 3 which explains how realistic-looking data is generated synthetically, preprocessing of the data and classification methods used to detect unique signatures. Finally, the framework is evaluated according to its recall, precision and speedup rates for different scenarios in Section 4, and we review the related work in Section 5.

2 Dataset

The dataset used in this study includes 14,985,016 samples. The snapshots recorded by the data acquisition system consist of 250 2-byte unsigned integers, denoting the energy intensities captured by the detector. With the exception of any noise (which can cause significant distortions), pulses are expected to exhibit patterns given by the following equations:

$$G(x) = \frac{A_1 e^{-k_1(x-T_1)}}{1+e^{-k_2(x-T_1)}} + O, \tag{1}$$

$$F(x) = \frac{A_1 e^{-k_1(x-T_1)}}{1+e^{-k_2(x-T_1)}} + \frac{A_2 e^{-k_3(x-T_2)}}{1+e^{-k_4(x-T_2)}} + O, \tag{2}$$

where G and F serve as models for the single pulse and double pulse data, respectively. Effectively, F replicates the pulse term in G, albeit with a different set of parameters. Parameters involved in these models are explained in Table 1. In case of a double pulse, rise rates (K_1, K_3) and amplitudes (A_1, A_2) of both signals can be significantly different from each other, whereas decay rates (K_2, K_4) are almost identical across all samples because decay rate is a property of the detector itself. The time difference between the two signals ($T_2 - T_1$) has a distribution which favors short time differences, meaning the two signals are more likely to overlap with each other rather than being two separate pulses. As the time difference decreases, or the relative amplitudes between the two signal heights become significantly different, it becomes increasingly difficult to decide if a trace is a single pulse or a double pulse trace, underlining the main challenge faced by nuclear physicists.

Currently used `lmfit` software uses non-linear least-squares curve fitting method with the Levenberg-Maequardt algorithm [16] implemented in the GSL (GNU Scientific Library). The target is to minimize the summation of the residuals by adjusting the parameters of the model:

$$\hat{\beta} \equiv argmin_{\beta} \sum_{i=1}^{N} (y_i - f(x_i, \beta))^2 \tag{3}$$

The Levenberg-Maequardt algorithm is an iterative procedure and works by calculating the Jacobian matrix for each step which increases the computational complexity of the algorithm.

Table 1. Parameters

Parameter	Definition
O	Offset
T_1	Starting position of the first pulse
K_1	Rise rate of the first pulse
A_1	Amplitude of the first pulse
K_2	Decay rate of the first pulse
T_2	Starting position of the second pulse
K_3	Rise rate of the second pulse
A_2	Amplitude of the second pulse
K_4	Decay rate of the second pulse

As noted earlier, single pulses always constitute the majority of the captured snapshots in all experiments. Detailed information about the specific data set that we are working with is given in Table 2.

Table 2. Dataset

Type	Count	Percentage
Single Pulse	14968801	99.9%
Double Pulse	16215	0.1%

3 Realtime Beta Decay Detection Framework

To enable real time processing of traces obtained in beta decay experiments, we propose a novel framework based on machine learning techniques. Since there is no need for single pulse data to be analyzed using `lmfit`, our proposed framework acts as an inexpensive filter whose main duty is to reduce the number of traces to be analyzed by `lmfit` as much as possible without missing any of the rare beta decay events.

Arguably, an important precedent to a good machine learning algorithm is a high quality training dataset. The fact that our dataset is highly skewed towards single pulse data presents a challenge at this point. There are different ways to handle skewed distribution of the classes. Weights corresponding to classes could be adjusted [19] or data could be resampled to have a fair balance of the classes [4]. For this problem, since we have a good analytical model for the different trace types, we choose to estimate the underlying parameter distributions of the double/single pulses and generate synthetic pulses. Models in Eq. 1 and 2 are used to capture the underlying parameters with high fidelity and generate synthetic data by feeding these parameters back into the equations. To obtain higher quality synthetic data, we use Kernel Density Estimation (KDE) techniques.

A shortcoming in using the analytical models for pulse generation is that we only can produce clean traces without any noise. Since the synthetic dataset should be representative of the real dataset to have better generalization performance, noise must be modeled too. By adding the modeled noise into the generated traces, one can obtain more realistic synthetic data for training.

Figure 2 gives a visual overview of this framework. The first step is determining distribution of parameters in Eq. 2 using `lmfit` on a small sample dataset which indeed can be obtained by scientists during their preparatory phase before the actual experiments begin. These distributions are then fed into our analytical models in conjunction with the noise model and synthetic traces are generated. Synthetic data are used to enhance the small real dataset obtained during preparations, and machine learned classifiers are training using the enhanced dataset. These classifiers are then used the filter out the uninteresting samples from the dataset, significantly accelerating the overall procedure. We note that classifiers may emit false positives, which can then be eliminated by `lmfit` in the final analysis stage.

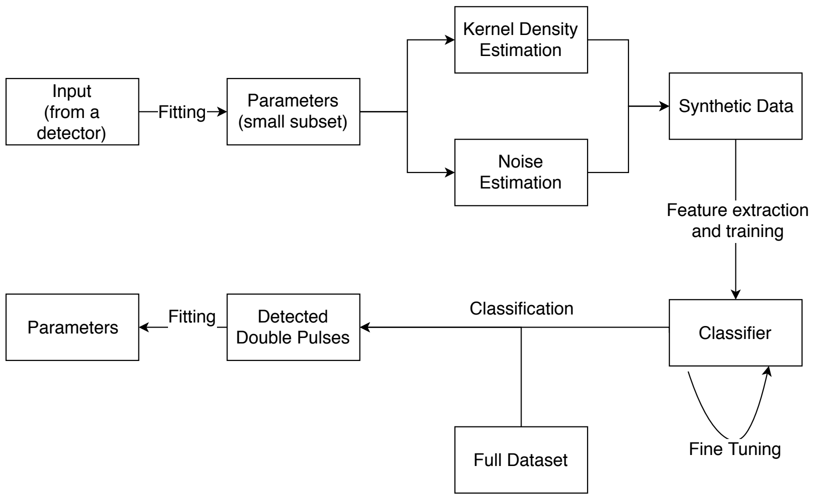

Fig. 2. Proposed method.

3.1 Synthetic Data Generation

Generating synthetic data to train a classifier is used in many domains including computer vision [18] and speech recognition [12]. We have chosen this approach due to the skewed distribution of classes in actual datasets. Note that there is no need for generating single pulse signals as the majority of traces collected are already traces with a single pulse. Consequently, we draw the single pulse data from the actual dataset itself, and synthetically generate the double pulse traces by approximating the underlying parameter distribution of this rare event.

Kernel Density Estimation (KDE) Histograms of individual parameters describing traces with double pulses are given in Fig. 3. As can be seen, it is hard to describe these distributions analytically. Also, due to the nature of the detectors and experiments, distributions are expected to change. Therefore, we use non-parametric kernel density estimation methods, as they work without requiring a parametric representation and can adapt to different distributions well without making any assumptions. With KDE, the formula for a set of observations X at a point x is given by:

$$f_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right), \tag{4}$$

where n is the sample size, K is the kernel function (smoothing function) and h is the window size or bandwidth. By changing the bandwidth h or the kernel function, smoothness of the estimated probability distribution function f can be adjusted.

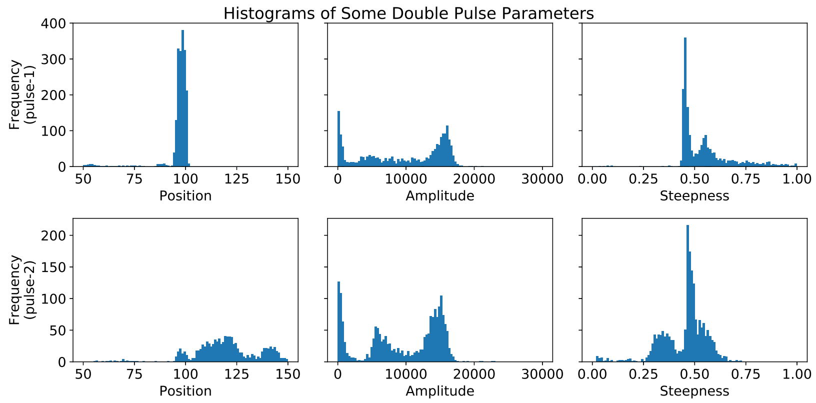

Fig. 3. Histograms of the selected parameters. The top row for the parameters corresponding to the first pulse of double pulse samples and the bottom row for the second pulse of double pulse samples

Since our aim is to use the synthetic data to train classifiers, we select the kernel and the bandwidth based on a grid search using a validation set created from real data samples. Other widely used methods for bandwidth selection are

Scott estimate [21] and Silverman estimate [22]. Based on earlier experiments, we have concluded that selecting the parameters based on the performance on the validation set yields better performance.

Using the KDE technique described above, new samples are generated to create a more balanced training dataset for which we generate 50,000 samples for the double pulse class. However, parameters require strict bounding. Position variable of the pulses should be in the $[0, 250]$ range. Also, decay time and steepness cannot be less than 0. To impose these boundary conditions, samples drawn from the KDE are checked for these conditions and the ones which violate any of the conditions are redrawn till all the conditions are met.

Noise Estimation Noise is a significant component of the traces obtained from the detectors. Depending on the detector, noise levels can actually be high compared to the pulse(s). To create a high quality synthetic dataset, we modeled the noise as well. While it is hard to separate the noise from the signals, we observe that the part of the trace before the rise of the first peak gives a good representation of the noise in the rest of the signal. Since this initial part of the trace before the pulse is expected to be flat, it is straightforward to detect the noise there. For the intensity values in this initial segment, we calculate the differences from the mean value in that range; create a histogram by aggregating the differences across all traces in the training data. As seen in Figure 4, for the particular dataset that we work with, the noise could be modeled as an additive Gaussian distribution:

$$\begin{aligned} Z_i &\sim \mathcal{N}(0, \sigma^2) \\ Y_i &= X_i + Z_i \end{aligned} \tag{5}$$

where Y_i is the output, X_i is the underlying true value defined by Eq. (2) and Z_i is drawn from the given distribution. The assumptions here are that Z_i is independent and identically distributed, and Z_i is not correlated with X_i. Basically, Eq. (5) implies that the detector adds some noise coming from a fixed distribution to the true values coming from the event.

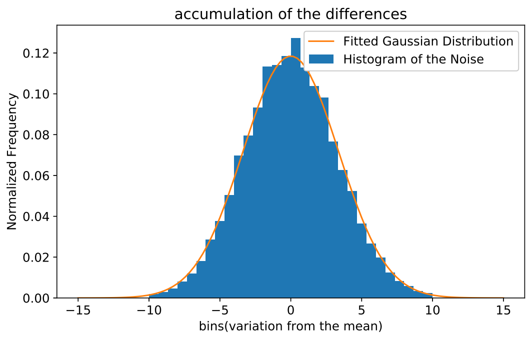

Fig. 4. Noise estimated for the experimental data

Based on the analysis via the given method, we found out that the noise for the particular experiment could be modeled as $\mathcal{N}(0, 11.34)$. We note that in cases where the noise does not fit into a simple analytical form such as the Gaussian distribution, one can use the KDE technique described above for accurately modeling the noise as well.

3.2 Preprocessing of the Data

Normalization Since the offset O and amplitude variables A_1 and A_2 cause variations between traces that are otherwise structurally similar, data needs to be normalized to minimize the effects of these variations. By doing so, classifiers are expected to become less susceptible to non-structural variations. We have chosen to use the L2-norm normalization for this purpose.

Dimensionality Reduction In the dataset, each sample consists of 250 points. As can be seen in Figure 1, majority of these points do not present any information about the class which the sample belongs to. Including unnecessary points increase the complexity of the classification process. We investigated the use of Principal Component Analysis (PCA) which is useful to represent the d-dimensional data in a lower dimensional space while keeping the maximum amount of variance. PCA is defined as an orthogonal linear transformation. By lowering the dimensionality, our expectation is that the classification process would get faster and it would require less space. To not have data leakage, we calculated the covariance matrix required for PCA by only using the training data. As will be discussed below, for the classifiers that use synthetic data, the covariance matrix is calculated on synthetic data only.

3.3 Classification Methods

Recently, neural network methods such as Convolutional Neural Networks (CNN) or different variations of Recurrent Neural Networks (RNN) have gained immense popularity for classification tasks. They are both proven to be successful on tasks related to time series analysis such as speech recognition ([17], [8]), and human activity recognition ([25]). However, they are computationally expensive since they operate on raw data and require high number of layers for accuracy.

As discussed before, speed is crucial for our purposes due to the high data acquisition rate and high number of signals that need to be classified and analyzed. Also, the system could tolerate a reasonable number of false positives because the `lmfit` method will be used to further clean unwanted traces and analyze all traces that are detected to have double pulses. Under these circumstances, we decided to test Random Forest (RF) [14], Support Vector Machines (SVM) [23] and Gradient Boosted Classifiers (GBC) [5]. For the RF, Classification and Regression Tree (CART) algorithm is used. It basically splits the nodes based on Gini index which is a way to calculate impurity [3]. Since a single decision tree may not be robust to outliers and noise, we trained a Random Forest classifier which is an ensemble learning method. Multiple decision trees are grown together and after the training, the classification is done by voting between the trained trees. GBC is another ensamble learning method. For GBC, Friedman Mean Squared Error is used as the splitting criteria [6]. The training procedure for GBC is sequential unlike RF where the trees could be grown in parallel. Each newly added tree is generated to correct the errors caused by the previous one. SVM tries to separate the dataset in order to classify the data into two groups by passing a linearly separable hyper-plane through the data space. However, most of the datasets are complex and not easily separable by a hyperplane. To solve that kernel trick could be used. The idea is that data could become linearly separable after projecting the input vectors to a higher dimensional feature space. Finally, all classifiers (RF, GBC and SVM) are tuned with grid search and the tuning is done based on precision and recall scores for double pulses.

4 Evaluation of the Proposed Framework

We present results from different classifiers tested in our proposed framework. We basically test three different scenarios to show how the framework performs under different conditions. In all scenarios, it is assumed that experimental non-unique signatures (single pulses) are available. For the first two cases, we assume that we have double pulse samples available as well, but in different quantities. For the last case, only single pulses and their parameter distributions are used to explore the scenario where there is no double pulse data available. We used two different splits for our evaluations for the first two scenarios. The first one inspects cases where there is a limited amount of real data for training purposes, such as during the brief preparation period leading up to the actual experiments, and the second one inspects cases with abundant real data, for instance, while repeating an experiment or on-the-fly tuning of ML models during the experiment which typically takes a week or so. For the first split, 10% of the available dataset is used for training (or generating synthetic data), another 10% is used for validation (tuning the parameters) and the remaining is used for testing. For the second split, these percentages are 60-20-20, respectively. In the third case where there is no experimental double pulse data available beforehand, synthetic double pulses are generated for training purposes by drawing the parameters from various uniform distributions. Since T_1, K_1,K_2,K_3 and K_4 depend on the properties of the detector but not the actual experiment, the parameter distributions of the single pulses for T_1, K_1 and K_2 (1) are used to model the listed parameters. T_2, A_1 and A_2 are drawn from the uniform distributions of $\mathrm{unif}(T_1, 250)$, $\mathrm{unif}(50, 16383)$ and $\mathrm{unif}(50, 16383)$, respectively, as determined by the experimental setup (16383 is the largest possible amplitude value that the detector could record). Using uniform distributions allows us to avoid any bias in the models when we assume there is no actual double pulse data available.

For each split, two different evaluations are performed. The first evaluation finds parameters of the double pulses in the training data, then estimates the distribution of the parameters and the noise. Then new parameters are drawn from the fitted distribution and synthetic data is generated by feeding the parameter to the functional for the double pulse traces. For the second evaluation, the training data is used directly without adding any synthetic data. Note that for the synthetic data based evaluation, only double pulse samples are generated with the described method since it is easy to acquire single pulse samples and using actual single pulses yield better results than synthetically generated ones. For the third case, as we assume there is no experimental data available for the double pulses, only single pulse parameters from Eq. (1) are used to model the detector dependent parameters and the rest are drawn from uniform distributions.

All computations have been performed on Laconia, a cluster with over 400 compute nodes at Michigan State University's High Performance Computing Center. Each of the base compute nodes on Laconia has 28 cores, located on two fourteen-core Intel Xeon E5-2680v4 Broadwell 2.4 GHz processors, and has 128 GB DDR3 2133 MHz ECC memory. Each core possesses a 64 KB L1 cache (32 KB instruction, 32 KB data), a 256 KB L2 cache. Between fourteen cores of a single "Broadwell" processor, a 35 MB L3 cache is shared. At the time of the experiments, the Laconia nodes ran CentOS version 6.8 distribution of GNU/linux for x86_64 architectures, kernel version 2.6.32-696.20.1, and glibc version 2.12-1.192. The software was built using the GNU Compiler Collection version 6.2 with the `-O3` flag. For all experiments, we restricted our computations to a single CPU core on a Laconia node. For the `lmfit`, GSL library is used and for the classification procedures sklearn package is used. Since both inference and `lmfit` steps are highly parallelizable due to the independence of the samples, single core speed is a good indicator for performance.

4.1 Classification Results

As mentioned above, two important metrics for our purposes are recall rate and speedup on the real data compared to the baseline `lmfit` method. Since our main goal is to create a filtering mechanism for the real data, we present results for the testing dataset coming from the real data. As the data is extremely skewed, we cannot use accuracy as a comparison metric. If we use a classifier which classifies everything as a single pulse, based on Table 2, the accuracy would be around 99.999% without doing any meaningful work. Therefore, we choose to compare the classifiers based on recall rates which shows how accurately a classifier detects double pulse traces. We present results from SVM, Random Forest and Gradient Boosting Classifier based classifiers trained on real data and synthetic data created with Kernel Density Estimation. The parameters of the KDE was selected based on the performance of the classifiers trained with it. When only 10% of the data (limited data) is used to generate synthetic data, 0.1 and Gaussian are chosen as KDE bandwidth and kernel respectively. When 40% of the data (abundant data) is used to generate synthetic data, the chosen parameters are 0.001 and Gaussian kernel. Also, the paremeters of the classifiers and number of components for PCA are chosen based on the performance in the validation set. For cases where there is experimental data available for double pulses, projecting the data onto the space created with 15 principal components yields better performance for the majority classifiers. As such, we fixed the corresponding parameter to 15 to limit the search space for PCA. On the other hand, for the last case, since the overall variance is high in the training data due to the uniform distributions, the first 50 components are used. Other hyperparameters required for the classifier and KDE have chosen based on their performance on the validation set.

Results for different datasets are reported in Table 3. Speedups with respect to the baseline `lmfit` method are calculated according to the following formula:

$$speedup = \frac{1}{T_c} + \frac{1}{T_{LMFit}} * (P_{double}/PR_c) \tag{6}$$

where T_c and is the throughput of the classifier, T_{LMFit} is the throughput for the `lmfit` method, P_{double} is the percantage of the double pulses in the dataset and PR_c is the precision rate of the classifier. Note that this speedup formula accounts for the cost of eliminating the false positives produced by our proposed classifiers using the `lmfit` method at the end.

Table 3. Classification Results

Training Data	Model	Recall	Precision	Speedup
Synthetic data with uniformly distributed T_2, A_1 & A_2	RF	0.9406	0.4315	37.8
	SVM	0.8448	0.6970	52.2
	GBC	0.9174	0.4944	44.0
Synthetic data generated from limited data	RF	0.9993	0.4566	40.0
	SVM	0.9914	0.1156	10.2
	GBC	0.9945	0.4667	41.5
Synthetic data generated from abundant data	RF	0.9995	0.4514	39.5
	SVM	0.9935	0.1012	8.9
	GBC	0.9951	0.5019	44.6
Limited data	RF	0.9843	0.1266	11.3
	SVM	0.9476	0.0323	2.9
	GBC	0.9741	0.0992	8.9
Abundant data	RF	0.9995	0.3266	28.8
	SVM	0.9943	0.0917	8.1
	GBC	0.9989	0.3708	33.0

As seen in these results, Random Forest method performs better in terms of recall rate and gives a reasonable speedup. Even though Gradient Boosting Classifier is a lot faster than Random Forest for inference, their speedups are relatively close. The reason is that GBC has lower precision than Random Forest and triggers the `lmfit` method more often to eliminate false positives. SVM has the lowest precision and the lowest throughput compared to the other methods. Based on these results, Random Forest method is a reasonable candidate to use for classification part of the proposed framework. The random forest models trained on abundant data synthetic data generated from abundant data have the highest recall rate which is 0.9995%.

When we further examine the results, it can be seen that in the absence of sufficient data, generating synthetic data for training purposes helps classifiers to learn more about the unique signature. When 60% of the data is used for the training, it results in better or comparable recall rate for the classifiers. For precision, classifiers trained on real data still lack behind the ones trained on synthetic data. One reason might be that in the 60% of the data, there are less than 10,000 double pulse samples. However, for the synthetic data, we are generating 50,000 double pulse samples, which is more than the number of double pulses in the whole dataset. When there is no double pulse available for the training and the given uniform distributions are used to generate double pulse samples, the recall rate of the classifiers are relatively lower unsurprisingly but the precision rates are high potentially related to the low recall rates. This causes the speedups to be high since precison and speedup are correlated based on Eq. 6. Also, since the actual data is skewed towards single pulses and the hyperparameters are chosen to maximize the recall rate as the classifiers primary purpose is to filter out non-unique signatures while keeping almost all of the double pulses, the precision rates vary a lot. This situation renders majority of the SVM models useless as low precision rates decrease their speedups.

5 Related Work

Different pulse shape classification (PSD) and discrimination methods using machine learning based approaches have been proposed in the literature [20, 1, 11]. More broadly, in nuclear physics use of machine learning techniques have been explored for other problems such as track classification [13] and jet classification [9]. In [20], it is shown that the support vector machine (SVM) method yields better results than the charge-integration PSD method that originated in analog systems. However, in our dataset, as shown above SVM performs worse compared to the ensemble tree based methods in terms of recall rate. In [11], a neural network architecture called auto-encoders is used to learn how to represent a given trace in a smaller space. Finally, another network is trained to classify transformed traces. We note that we also experimented with neural networks in this study and although neural networks were highly accurate in the classification task, they were significantly slower than the methods we presented above.

To be able to deploy a supervised machine learning method, sufficient amount of training data should be available beforehand. To overcome that issue, simulated data is used for track classification in [13] where the authors show that even though simulated data help training supervised classifiers, results are more promising when experimental data is used. Our results support the same conclusion and we offer a novel way to increase performance when simulated data is used for training.

Finally, we note that unlike the typical classification tasks, it is not practical to compare the performance of the classifiers reported in this study to others, because the detector, its setup and the way experiments are conducted render a classifier developed for a given context ineffective for another context.

6 Conclusion

We proposed a novel framework to enable realtime analysis of data from beta decay experiments performed at MSU's National Superconducting Cyclotron Laboratory. The framework uses computationally inexpensive machine learning techniques to accelerate the classification bottlenecks with use of the existing `lmfit` method. We also developed a synthetic training set generation tool to address the issue of limited (or non-existent) double pulse samples observed in actual training datasets beforehand. In conclusion, we observe that generating synthetic data significantly improves the recall rates for the classifiers used, if the real dataset is limited in size and/or the unique signature we are looking for is extremely rare in the dataset. By modeling the distribution of the underlying parameters via KDE and generating new samples by feeding parameters drawn from KDE to the functional form of the wanted pattern, we have been able to develop a high-performance filtering mechanism to speedup the overall data analysis process. For the classification schemes with highest recall rates, *i.e.,* RF and GBC, we observe speedups up to $44x$ compared to the current method. Coupled with use of parallelism on a server with moderate computing power, the proposed framework can easily provide real-time analysis guarantees.

As future work, in terms of data generation, a Generative Adversarial Network [7] could be used. It could generate more realistic samples without even providing a functional form for the signature pattern and estimating the distribution of its parameters. In terms of further performance improvements, to ensure real-time analysis with the rate of data collection that will be possible at the upcoming Facility for Rare Isotope Beams, an \$850M facility funded by the US Department of Energy, our proposed framework can be ported to FPGAs and GPUs. There are several examples in the literature which demonstrate significant performance boosts by the use of these accelerators, see for instance [24], [15] and [2].

References

1. Balmer, M.J., Gamage, K.A., Taylor, G.C.: Comparative analysis of pulse shape discrimination methods in a 6li loaded plastic scintillator. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **788**, 146–153 (2015)
2. Bauer, S., Köhler, S., Doll, K., Brunsmann, U.: Fpga-gpu architecture for kernel svm pedestrian detection. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops. pp. 61–68. IEEE (2010)
3. Breiman, L., Friedman, J., Olshen, R.: Stone cj. classification and regression trees. wadsworth wadsworth statistics/probability series (1984)
4. Chawla, N.V.: C4. 5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In: Proceedings of the ICML. vol. 3, p. 66 (2003)
5. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
6. Friedman, J.H.: Stochastic gradient boosting. Computational Statistics & Data Analysis **38**(4), 367–378 (2002)
7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
8. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: Acoustics, speech and signal processing (icassp), 2013 ieee international conference on. pp. 6645–6649. IEEE (2013)

9. Guest, D., Cranmer, K., Whiteson, D.: Deep learning and its application to lhc physics. Annual Review of Nuclear and Particle Science **68**, 161–181 (2018)
10. Heyde, K., Wood, J.L.: Shape coexistence in atomic nuclei. Reviews of Modern Physics **83**(4), 1467 (2011)
11. Holl, P., Hauertmann, L., Majorovits, B., Schulz, O., Schuster, M., Zsigmond, A.: Deep learning based pulse shape discrimination for germanium detectors. arXiv preprint arXiv:1903.01462 (2019)
12. Ko, T., Peddinti, V., Povey, D., Seltzer, M.L., Khudanpur, S.: A study on data augmentation of reverberant speech for robust speech recognition. In: Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on. IEEE. pp. 5220–5224 (2017)
13. Kuchera, M.P., Ramanujan, R., Taylor, J.Z., Strauss, R.R., Bazin, D., Bradt, J., Chen, R.: Machine learning methods for track classification in the at-tpc. arXiv preprint arXiv:1810.10350 (2018)
14. Liaw, A., Wiener, M., et al.: Classification and regression by randomforest. R news **2**(3), 18–22 (2002)
15. Mishina, Y., Murata, R., Yamauchi, Y., Yamashita, T., Fujiyoshi, H.: Boosted random forest. IEICE Transactions on Information and systems **98**(9), 1630–1636 (2015)
16. Moré, J.J.: The levenberg-marquardt algorithm: implementation and theory. In: Numerical analysis, pp. 105–116. Springer (1978)
17. Palaz, D., Magimai-Doss, M., Collobert, R.: Analysis of cnn-based speech recognition system using raw speech as input. In: Sixteenth Annual Conference of the International Speech Communication Association (2015)
18. Richardson, E., Sela, M., Kimmel, R.: 3d face reconstruction by learning from synthetic data. In: 3D Vision (3DV), 2016 Fourth International Conference on. pp. 460–469. IEEE (2016)
19. Rosenberg, A.: Classifying skewed data: Importance weighting to optimize average recall. In: Thirteenth Annual Conference of the International Speech Communication Association (2012)
20. Sanderson, T., Scott, C., Flaska, M., Polack, J., Pozzi, S.: Machine learning for digital pulse shape discrimination. In: 2012 IEEE Nuclear Science Symposium and Medical Imaging Conference Record (NSS/MIC). pp. 199–202. IEEE (2012)
21. Scott, D.W.: On optimal and data-based histograms. Biometrika **66**(3), 605–610 (1979)
22. Silverman, B.W.: Density estimation for statistics and data analysis. Routledge (2018)
23. Suykens, J.A., Vandewalle, J.: Least squares support vector machine classifiers. Neural processing letters **9**(3), 293–300 (1999)
24. Van Essen, B., Macaraeg, C., Gokhale, M., Prenger, R.: Accelerating a random forest classifier: Multi-core, gp-gpu, or fpga? In: 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines. pp. 232–239. IEEE (2012)
25. Yang, J., Nguyen, M.N., San, P.P., Li, X., Krishnaswamy, S.: Deep convolutional neural networks on multichannel time series for human activity recognition. In: Ijcai. vol. 15, pp. 3995–4001 (2015)