Pulses Characterization from Raw Data for CDMS

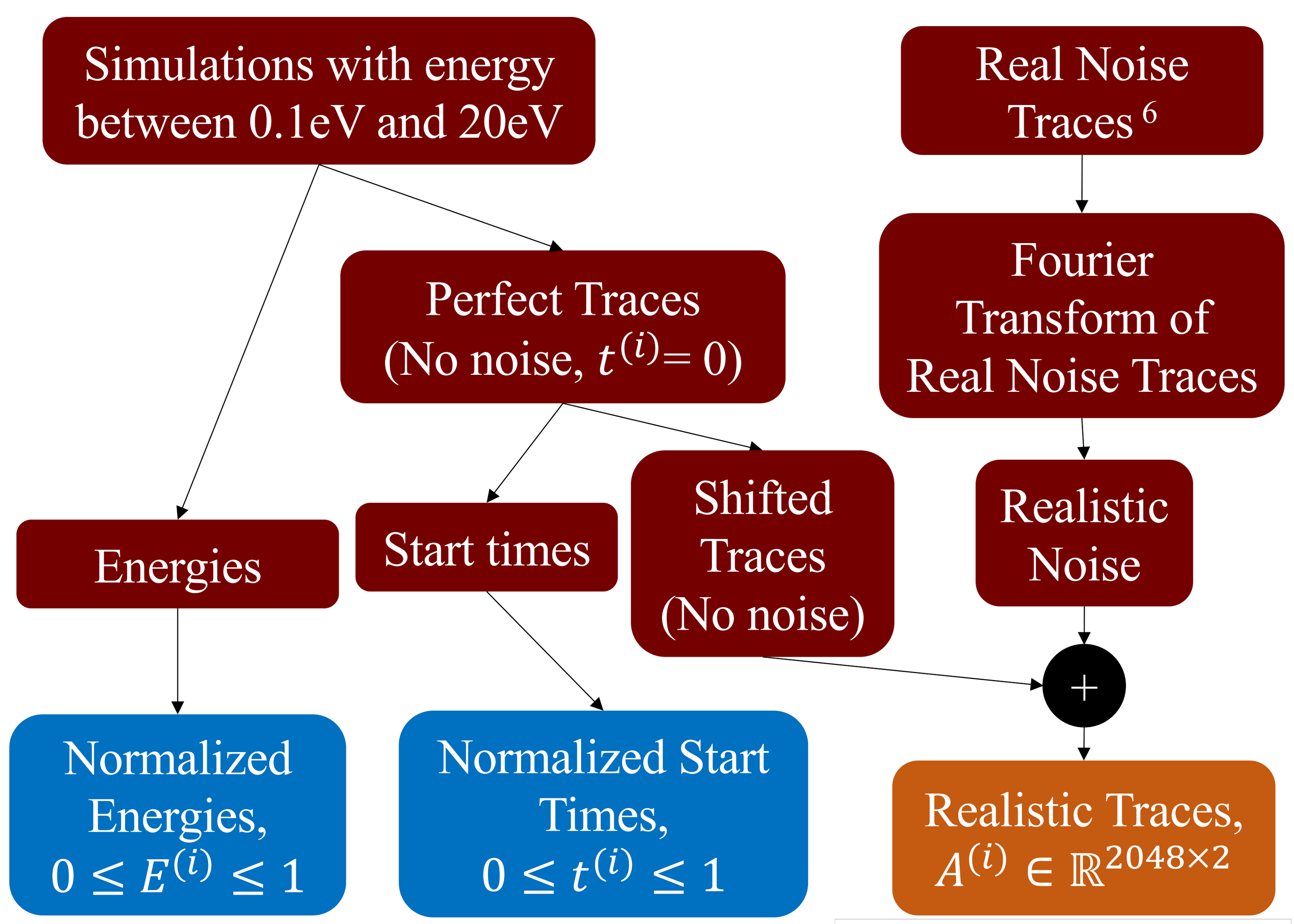
Francesco Insulla[1,2], Chris Stanford[1,3]
[1]Stanford University Physics Department, [2]franinsu@stanford.edu, [3]cstan4d@stanford.edu

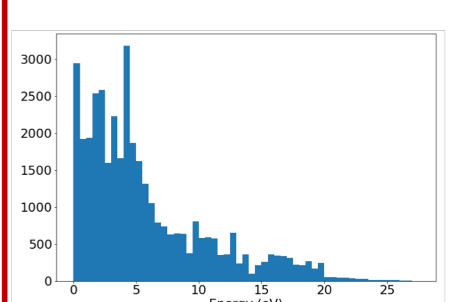
Introduction & Motivation

We seek to process signal pulses from a detector used in our physics lab[4] ($A^{(i)}$) and identify the start time ($t^{(i)}$) and energy ($E^{(i)}$) of the pulse. Currently, our techniques for registering a pulse are not accurate, and often classify detector noise as a pulse. This issue is significant, because the separation of signal and noise is critical to the success of the experiment, and is easily generalized to any other detector of this type. Furthermore, the problem of processing data to find pulses, and characterizing them, has the potential for a wider range of uses. We implement liner regression, fully connected neural networks (FCNNs), convolutional neural networks (CNNs), and kernelized principle component analysis (KPCA) with FCNN to predict $t^{(i)}$ and found the most success with standard PCA + FCNN.

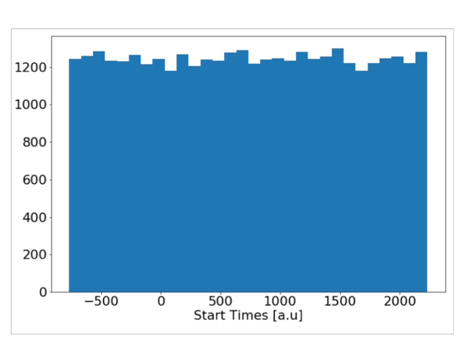
Dataset

While we don't have enough real[5] data to train on, we have do have a Monte Carlo simulation of our experiment. Combining the results from simulating with real noise, we created our dataset as represented by the following diagram.

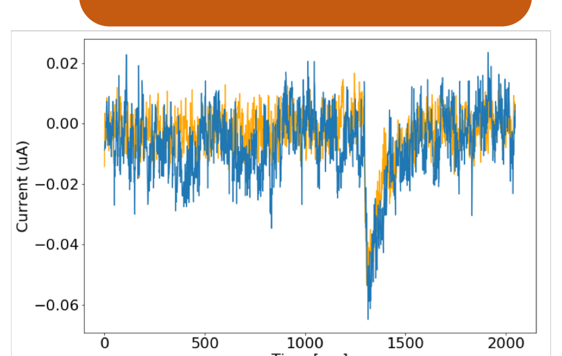




Distribution of Energies



Distribution of Start Times

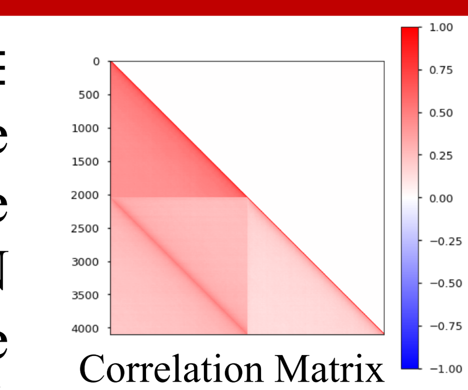


Example Pulse

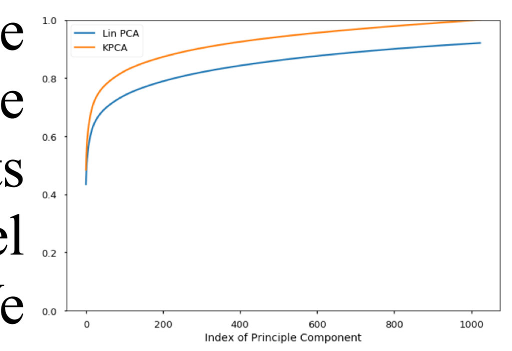
(Train, Dev, Test): (85, 10, 5)%, 39,458 total

Features

The input features were traces with two channels: $A^{(i)} \in \mathbb{R}^{2048\times 2}$. For the linear regression and FCNN we flattened the trace to a 4096 dimensional vector. For the CNN we kept the shape of the trace. For PCA + FCNN model we flattened the traces and used PCA to find the first 1024 principle of the components (PCs) using 20% of the training set – which explain 89.83% of the variance. We decided to try PCA because we plotted the correlation matrix of the traces an noticed all the points of the trace are positively correlated. For the Kernel PCA, we used a radial basis kernel, and 1024 PCs. We tried this too because we thought the relationship between the projected features could be non-linear.



Correlation Matrix Lower Triangle



Cumulative Explained Variance Ratio

Models

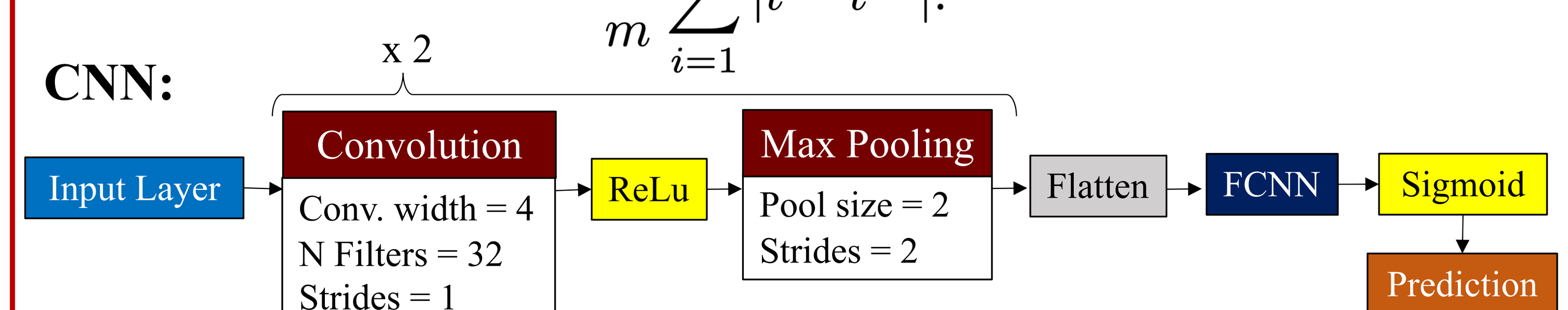
We represent $A^{(i)} \in \mathbb{R}^{2048\times 2}$ and $a^{(i)} = \text{Flatten}(A^{(i)}) \in \mathbb{R}^{4096}$.
In all the models we minimize the mean squared error (MSE):

$$\frac{1}{m}\sum_{i=1}^{m}(t-\hat{t}^{(i)})^2,$$

however we are interested in reporting the mean absolute error (MAE):

$$\frac{1}{m}\sum_{i=1}^{m}|t-\hat{t}^{(i)}|.$$

CNN:



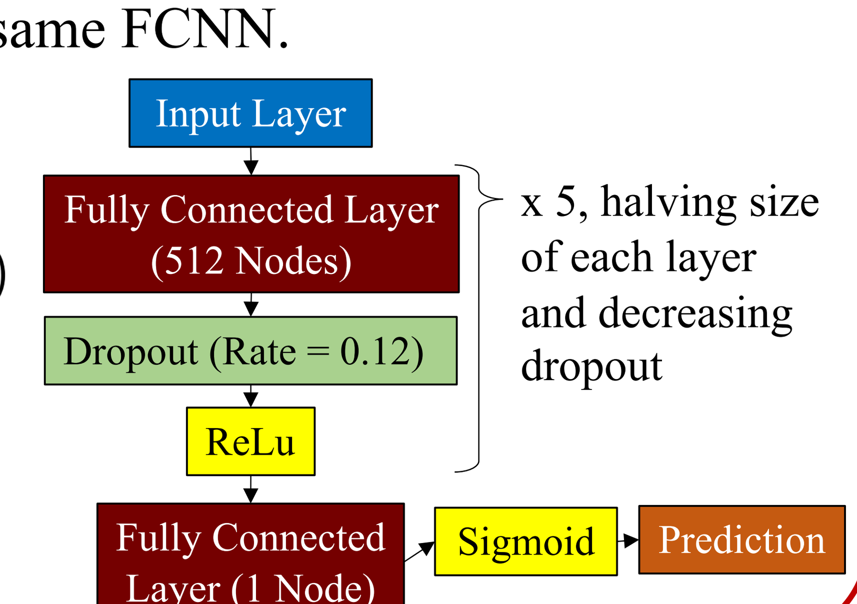
Linear PCA FCNN: We perform PCA by finding the eigen-basis of the correlation matrix C, where V^{k^T} is the k-th principle component:

$$C = \frac{1}{m}\sum_{i=1}^{m} a^{(i)}a^{(i)^T} \qquad \tilde{a}_k^{(i)} = V^{k^T}a^{(i)}$$

We then feed the projections, $\tilde{a}^{(i)}$, into the FCNN displayed below.

Kernel PCA FCNN: We apply the kernel trick on the projections as follows and feed the result into the same FCNN.

$$\begin{aligned}\tilde{a}_k^{(i)} &= V^{k^T}\Phi(a^{(i)}) \\ &= \sum_{j=1}^{m}\left(c_j^k\Phi(a^{(j)})\right)^T\Phi(a^{(i)}) \\ &= \sum_{j=1}^{m} c_j^k K(a^{(j)}, a^{(i)})\end{aligned}$$



Results

Table of Scaled MAE	Linear Regression	Shallow FCCN[7]	CNN	Linear PCA + FCNN	Radial Basis PCA + FCNN
Training	321.40	145.84	58.93	15.24	27.59
Validation	364.03	160.12	123.74	17.91	73.12
Test	368.80	210.56	180.02	21.73	104.67

We then estimate the variance using by bootstrapping: training 10 times on a random sample of the training set 90% of the size, and predicting on the test set. We got an estimated variance of 32.14.

Discussion

From this project were able to develop a methodology to construct an effective tool that we might use as part of physics experiment to determine the start-time of pulses measured by our detector. After constructing our dataset from our Monte Carlo simulations, we trained liner regression, FCNN, CNN, a standard PCA fed into FCNN, and KPCA with a radial basis kernel fed into a FCNN to predict $t^{(i)}$ and had the best Test set MAE with the standard PCA + FCNN method. Our goal was to get to a MAE around 1 or 2, however the lowest we ever got on training was 4. This is likely due to the fact that the pulses are so noisy – which is why we chose this challenging problem in the first place. An important insight from this project was that more complex models don't always produce better results, as can be seen comparing the CNN and KPCA+FCNN with the PCA+FCNN. Another lesson we learned was that producing the dataset and preparing it for training can be the most time intensive step. Finally, while we didn't accomplish exactly what we set out to do we are content with out results and will continue improving on them.

References

Watson, A. W. (2017). *Transverse position reconstruction in a liquid argon time projection chamber using principal component analysis and multi-dimensional fitting* (Order No. 10270707). Available from ProQuest Dissertations & Theses Global. (1906685475). Retrieved from https://search.proquest.com/docview/1906685475?accountid=14026

Future

If we had more time we would:

- Try other models, including Recurrent Neural Networks
- Tune hyper-parameters more methodically, keeping track of all results
- Use a larger dataset, with more examples per energy and also a wider range of energies

Acknowledgements

We would like to thank:
To Chin Yu

- For suggesting this project and providing key insights about Machine Learning.

The Northwestern SuperCDMS team

- For providing the MC simulation and the sample noise traces from which we generated our dataset.

The CS299 TAs and Professors

- For giving us the opportunity to do this project and teaching us all the material.

Appendix

[4]Quasiparticle-trapping-assisted Electrothermal-feedback Transition-edge- sensors.
[5]Produced by the detector.
[6] A time series defined by an array of 2048 values where each value represents a current measured by the detector.
[7]One 512 node hidden layer