CS230

SACLA Timing Tool Calibration
Based on Neural Network Algorithm

Viktor Krapivin
Yijing Huang[*]
Department of Applied Physics
Stanford University

Abstract

We calibrated SACLA timing tool using neural network algorithm and was able to correct the problems caused by traditional calibration method. We trained two neural networks, the first is used to filter cases where timing tool physically fails, the second returns the value we need for time tool calibration. This is a more user friendly scheme as it does not require user to switch between models and tune the parameters. It also retains more data points because and could potentially help us improve time resolution.

1   Introduction

SACLA timing tool uses the "spatial encoding" method to calibrate the jitter between ultrashort X ray pulse and optical pulse. The X ray pulse decreases the transmission of optical pulse through GaAs crystal. Therefore when two beams overlap in space and time, certain parts on the image taken of the transparent GaAs crystal turn darker than it would be without two beams overlapped. The shot by shot jitter between the two pulses is encoded in the position of an identifiable edge between dark part and bright part of this timing tool image [1](see 1 (a)). The goal is to first to distinguish between cases where this jitter information is physically encoded/not encoded, and second, if it is encoded, return the edge position as a number which will later be translated into timing jitter.

2   Related work

Traditional timing tool analyzer program will filter out images that are not physically encoded and then perform fitting algorithm on images that are successfully encoded.
The cases of physcially failed encoding can be categorized into X-ray caused, laser caused or other, see [1]. Important cases include, weak X ray pulse, low laser intensity, saturated laser intensity, temporal overlap out of range, lack of edge, and false edge. The paper published for the earliest commissioning of the calibration tool indicates that these cases in total only contribute 0.5% of the total shots, see [1]. In our experience, long term use of the machine causes damage spots. Furthermore due to temperature fluctuations (such as night to day) experimental conditions produced by the 1 kilometer long laser may change. It is important to develop a more robust method.
For successfully encoded images, the user program first takes the input image(see 1 (a)) and integrates over vertical axis to project on horizontal axis(see 1 (b)). The orange trace is projected from reference image with no X ray, and blue trace is projected from a successfully encoded X ray

---

[*]krapivin@stanford.edu, huangyj@stanford.edu

on images, which shows a clear rising edge. Dividing the X ray on image with the X ray off image yields 1 (c). This preprocessed curve is then passed to an edge fitting algorithm chosen by user, including 1) edge fitting with an erf function, 2)smoothing and taking the maximum of its derivative to find the edge. These algorithms suffer from high failure rate when there is intensity dependent damage spot. In practice, this fitting scheme relies on the edge preserving an ideal shape across the experiment. Therefore when machine drifts on year scale and damage spots occur, users need to fine tune the fitting/smoothing parameters, as well as region of interest(ROI) to make the classification and edge finding work. This becomes very user unfriendly, and is a great distraction from other beam time tasks.

We believe that the deep learning algorithm will be a solution to the problems. We can train neural networks to address drifting conditions through data from multiple days and experiments. The task can be classified as image discrimination(returns a True/False label) and image regression(returns the encoded value). For the second task we refer to [2].

3   Dataset and Features

Throughout the beamtime we have on the order of 1e7 images. Each raw image is cropped into (1920, 110).

For training the discriminator, we use the raw images. We can easily get images where we know X ray laser is off(X ray laser status is recorded in the data stream), and call it M1-Dataset1, we then hand label the images where the traditional algorithm returns a NaN(not a number) . These include images where the algorithms fails but still have identifiable edges, and images where the X ray could be on but the encoding is bad. We call it M1-Dataset2. M1-Dataset1 includes 5e3 images and M1-Dataset2 includes only 5e2 images. We use both datasets for training, and preserve a portion of them for test and development.

For training the image regression machine, we use raw image divided by reference X ray off image. Note that a damage spot which does not exist in the original commissioning report[1] shows up for some shots but not the others. The eventual "label" Y will be a scalar representing the pixel value of the edge. We create M2-Dataset1 using the encoded value as output by SACLA user timing tool analyzer, which is of size 1e6. We also create a smaller M2-Dataset2 by supervising the peak finding, this one is of size 5e2. We split M2-Dataset1 for training set and development set. The data size of training set for our basic algorithm is approximately $2 * 10^5$ images and the dev set is $5 * 10^4$ images. We use M2-Dataset2 for test set.

4   Methods

The Discriminator flattens the image and put it into 4 dense layers. The neuron numbers of each layer are 3, 10, 10, 10 separately. The loss function is binary cross entropy with L2 regularization on weights and bias. The optimizer is Adam.

The image regression machine uses 3 convolutional neural nets followed by three densely connected neural nets. The filter sizes of the 2D ConvNets are (3,3,13),(3,3,21),(3,3,33),(3,3,44), and the densely connected layers contain 40, 10, and 1 neurons separately. The loss function is mean absolute error plus regularization on weights and bias. The optimizer is Adam. The last two layers are trained through transfer learning.

The architectures of these two neural nets are shown in figure2. Code may be seen at the github repository `https://github.com/vikrapivin/timetool_edge_finding`.

5   Experiments/Results/Discussion

We trained 1 epoch of the algorithm for a set of approximately 10^5, load a new image set and continue training until we have trained through our entire training set(in total 10^7). The reason for this is that a particular dataset in our field is typically saved as an hdf5 file from the experiment, and loading minibatches allows for training the neural network without overflowing memory.

For hyperparameter tuning, we tried a range for convolutional filter sizes for each layer, regularization penalty, dense layer sizes, and optimizer parameters including learning rate. This is done in a systematic way, in combination with several choices of loss function. We randomize the choices

and train neural nets for a while for each choice, save several metrics for each training scheme, and compare the results to find out which is the best choice. And then we decide on the hyper parameter that we want to use.

Primary metric we use is mean absolute error as this metric produced better results on the whole data set. Additionally, the best model using either the mean absolute error metric of mean squared error metric produces similar results. To visualize the results of the binary classifier we output a confusion matrix, see Table5. This discriminator has a great overall error rate of about 96%. There is a large false positive rate (which means too many shots are labeled as bad shots), but this is an improvement from the status quo as the entire data set was sampled from images that were previously classified as bad images. This discriminator will potentially allow the use of many more shots in an experiment.

TABLE 1: Confusion Matrix

	True Label	False Label
True Prediction	12	1
False Prediction	3	107

6 Conclusion/Future Work

We obtained two trained neural networks that could be used to calibrate timing jitter between X ray pulse and laser pulse at SACLA, a discriminator for good and bad encoding, and a regression machine that returns the edge position pixel number. As we get more beamtimes, we can always use the same working pipeline to continue training the neural network as the machine possibly drifts on the year scale. This would allow us to focus on more nontrivial beamtime tasks and benefit other users. We also seek to use this calibration to improve time resolution with this better calibrated time jitter, since time resolution will be crucial to certain types of experiment.

7 Contributions

Yijing Huang: Data preprocessing and labeling. Viktor Krapivin: Hyper parameter tuning and implementation on AWS server. Both group members contributed equally to model design.

References

[1]K. Nakajima, Y. Joti, T. Katayama, S. Owada, T. Togashi, T. Abe, T. Kameshima, K. Okada, T. Sugimoto, M. Yamaga, T. Hatsui, and M. Yabashi, J Synchrotron Rad 25, 592 (2018).

[2] N. Slater, Github (2016).
`https://gist.github.com/neilslater/40201a6c63b4462e6c6e458bab60d0b4`

[3] E. Linder-Norén, "2ndMessiah", and Andrea Ramazzina, Github (2019). `https://github.com/eriklindernoren/Keras-GAN/blob/master/acgan/acgan.py`
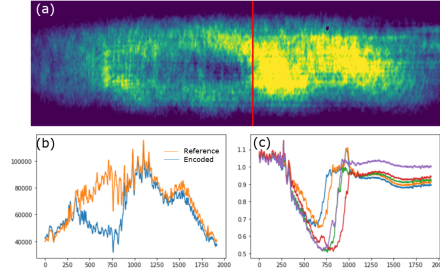
Figure 1: (a)Typical image of a successfully encoded shot. Red line shows the edge position.(b) Intensity projection of 2D image onto horizontal direction, here shows X ray on curve(orange) and the X ray off curve(blue). Dividing the X ray on trace with X ray off trace , and we get normalized intensity projection as shown in (c) for several different images. The sharp rise is then fit with an error function.
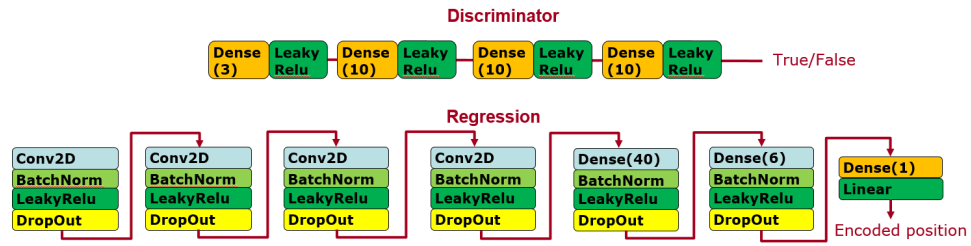

Figure 2: The Discriminator flattens the image and put it into 4 dense layers. The image regression machine uses 3 convolutional neural nets followed by three densely connected neural nets.
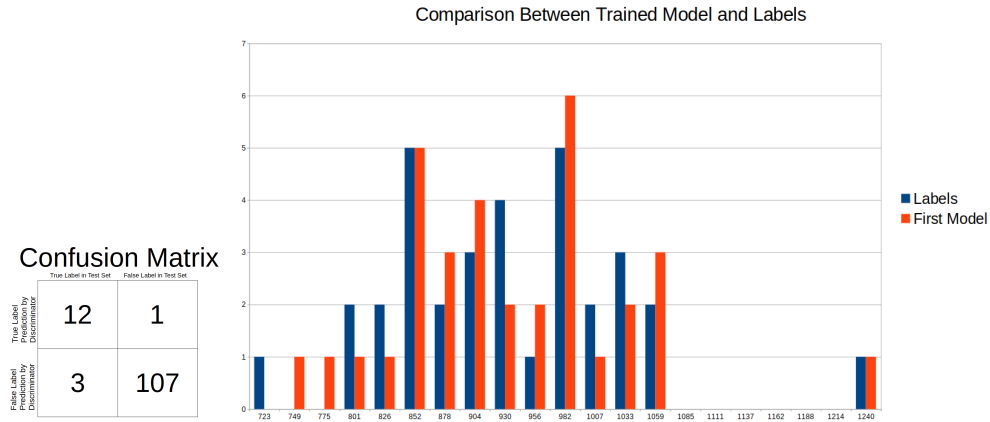

Figure 3: Left: This matrix indicates the type of errors that the transfer trained discriminator outputs. Right: this is a histogram of predictions of the transfer trained model versus the original labels.
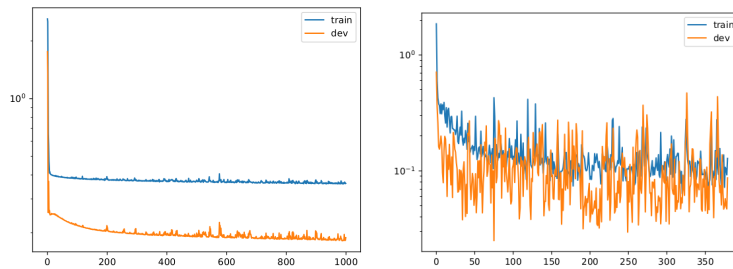

Figure 4: Left: This is the loss from the training of the transfer discriminator. Right: this is a histogram of the edge position regression.