

Beam Detection Based on Machine Learning Algorithms

Haoyuan Li, *hyli16@stanofrd.edu* Qing Yin, *qingyin@stanford.edu*

Abstract—The positions of free electron laser beams on screens are precisely determined by a sequence of machine learning models. Transfer training is conducted in a self-constructed convolutional neural network based on VGG16 model. Output of intermediate layers are passed as features to a support vector regression model. With this sequence, 85.8% correct prediction is achieved on test data.

Index Terms—Beam Detection, SVM, CNN

I. INTRODUCTION

THE free electron laser(FEL) at Stanford Linear Accelerator Center(SLAC) is an ultra-fast X-ray laser. As one of the most advanced X-ray light source [1] [2], it is famous for its high brightness and short pulse duration: it is 10 billion times brighter than the world’s second brightest light source; the pulse duration is several tens femtoseconds. It plays a pivotal role in both fundamental science research and applied research [2]. The mechanism behind this laser is very delicate [1]. Thus to keep the laser in optimal working condition is challenging. The positions of the electron beams and the laser beams are of fundamental importance in the control and maintenance of this FEL.

Currently, the task of locating beam spots heavily depends on human labor. This is mainly attributed to the wide varieties of beam spots and the presentation of strong noises as demonstrated in Figure 1, where the white square marks the boundary of the beam spot. Each picture requires a long sequence of signal processing methods to mark the beam position. To make things even worse, different instrument configurations and working conditions require different processing parameters. Within the frame of same configuration, the parameters will also drift away along with time advancement because of the inherent delicacy of the instrument. This makes tuning and maintaining the FEL a tedious and burdensome task for researchers. Currently, the data update frequency of the laser is 120 *Hz*. We can barely handle this. In 2020, after the scheduled update, the frequency will climb to 5000 *Hz*. Thus the only hope lies in automatic processing methods. Considering that simple signal processing method can not handle such

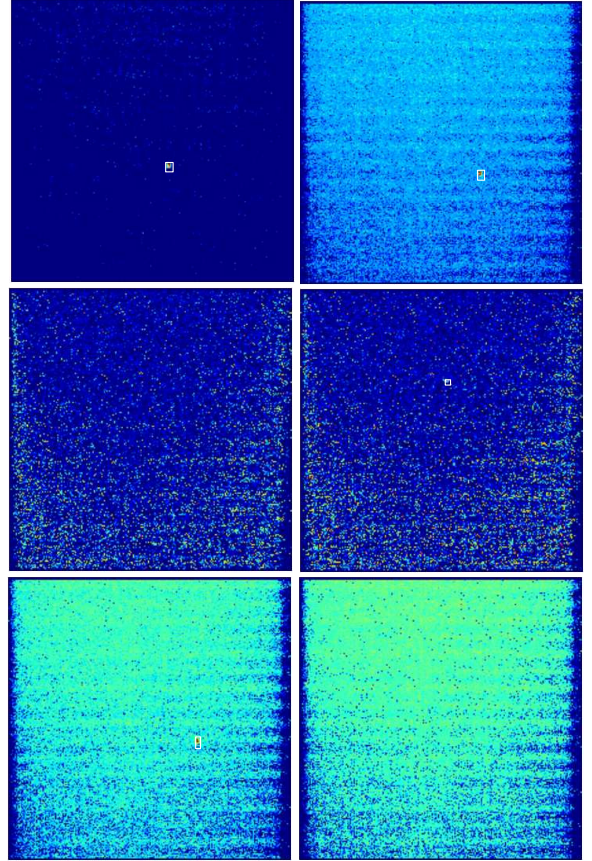


Fig. 1: Raw photos

nasty condition, We hope to come up with a general machine learning algorithm which is capable of locating the beams’ positions quickly and automatically. In this report, we demonstrate the consecutive application of neural network and supportive vector machine (SVM) to determine the positions of beam spots on virtual cathode camera (VCC) screen in simple cases.

II. CHALLENGES AND STRATEGIES

A bird view of such a simple-looking task reveals the challenges behind it.

- Complicated background noises. The background noises are not static. In contrast, it is coherent in both time and space domain due to quantum

effect. Thus we have to dynamically change the background when doing background subtraction for different images.

- Large variety in the intensity and shape of the beam spot. As is indicated above, the maximum intensity of the beam is incredibly high. However, it can also be 0 which corresponds to the case that no beam spot presented. The shape of the beam can also varies significantly from case to case.
- Ground truths is hard to find. The signal processing methods fail in finding the beam spot sometimes. There are also cases when the photo is so terrible that even human can not find the beam spot after all the signal processing procedures.

The task is similar to face recognition and the expectation of solving it in one strike is unrealistic. Thus, we develop the following strategy. First, we develop an algorithm that is capable of dealing with cases where we have implemented signal processing to realize the fundamental functions. Second, extend the application to cases that we gradually ignore pre-processing steps, e.g. Gaussian blur. In the end, extend the application to raw images. In the essay, we report the accomplishment of the first step that realizes fundamental functions.

III. RELATED WORK

A similar dilemma stated above occurred when researchers tried to pinpoint the electron bunch's trace [3]. It was solved by first feeding the images into a convolutional neural network [4], VGG16 [5], and then performing linear regression on its codewords. This gives us the inspiration of this project. Signal processing algorithms have been developed for targeting the beam spot. Thus, we use the results from these algorithms as the ground truth for our training.

IV. DATASET

Training dataset for this project consists of 162 original VCC screen figures and 16200 figures generated from the original ones (each original image corresponding to 100 generated images). Each figure contains only one beam spot. To generate new figures, we first cut out part of figure that contains the beam spot, then rotate the small patch of figure to some random degree, and put the patch on a random point. In the end, cut out the covered part of the figure and paste it to the original position of the beam spot. The position of the beam on each of the original figures can be determined by the signal processing algorithm. The position of the beam on the generated figures are determined by the generation process. The position of beam in each figure is

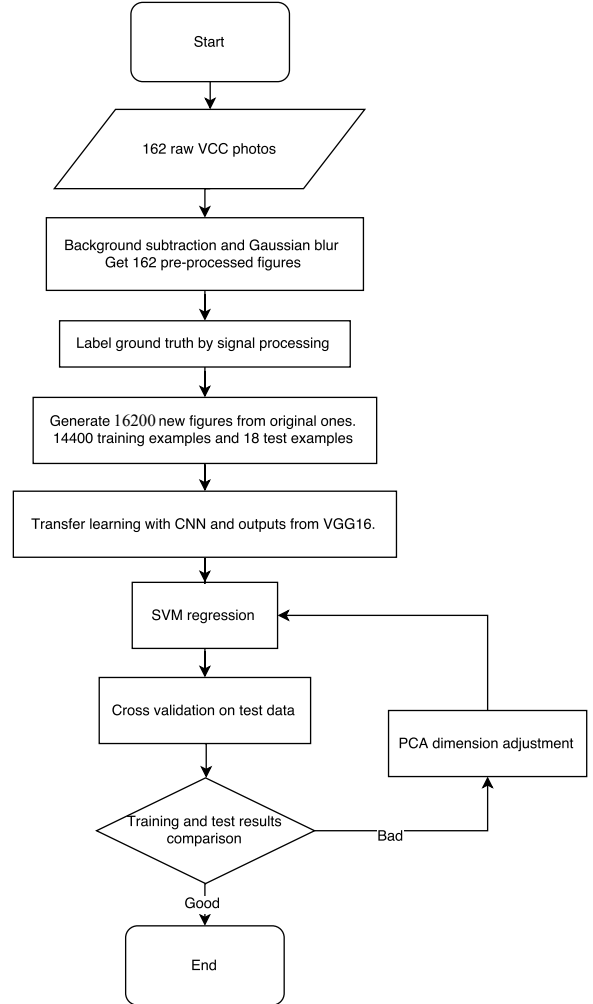


Fig. 2: Algorithmic flowchart

represented by the coordinates of two diagonal vertices:
 $z^{(i)} = (y_{\min}^{(i)}, y_{\max}^{(i)}, x_{\min}^{(i)}, x_{\max}^{(i)})^T$

V. METHOD

A. Pipeline

To obtain an algorithm robust to variations of beam spot and noises, we utilize a two step method.

First, we use a convolutional neural network (CNN) as a feature extractor and we take the intermediate outputs of the CNN as features and train a Supportive Vector Regression(SVR) program with them. The pipeline is demonstrated in Figure 2.

B. Feature Extraction

Since we have moderate sample size, transfer training seems suitable for this purpose. VGG16 is selected as the pre-trained model.

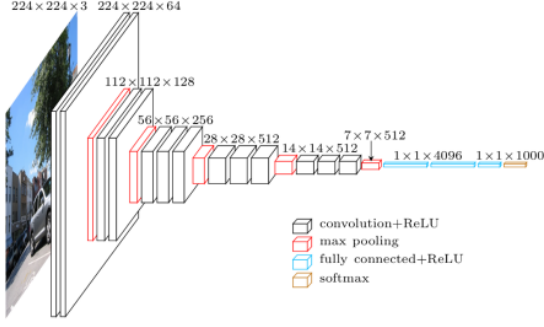


Fig. 3: Structure of VGG 16 [6]

VGG16 is a deep convolutional neural network. There are totally 13 layers of convolutional layer, 5 pooling layer and 3 full connected layer. The structure of this network is demonstrated in Figure 3.

The output of the first and second fully connected layers are both 4096-dimensional vectors. We feed in our samples and use these two vectors as the features of the image.

To do transfer training, first we re-construct the VGG16 with *python 2.7* and *tensorflow 0.11*. The last layer of the original VGG16 is a classifier of a thousand classes. We modify it to output the position of the center of the beam spot. The loss is defined as the squared distance between the predicted beam center and the true beam center. A l^2 regularization term is added which contains parameters of the last three fully connected layer.

$$\mathcal{L}_{CNN,loss} = \frac{1}{m} \sum_{i=1}^m \|\vec{x}_i - \vec{\tilde{x}}_i\|^2 + \frac{\lambda}{m} \|W\|^2 \quad (1)$$

where \vec{x}_i and $\vec{\tilde{x}}_i$ indicate respectively the predicted center of the beam and the true center of the beam which is represented by the center of the box. W is a vector consisting of all parameters in the last three full connected layers, i.e. we only tune the parameters in the last three layers.

The reason for only tuning the last three layers is that we don't have enough data to tune all parameters. The first several layers, according to modern viewpoint, mainly detect the detailed structures while the upper layers concern more about larger scale objects. So it seems more reasonable to try to tune those layers affecting our next step directly.

C. Support Vector Regression

The feature for the i th figure is denoted by $q^{(i)} \in \mathbb{R}^{8192}$. To avoid overfitting, we use l^2 regularization when implementing supportive vector regression (SVR) with Gaussian kernel to model the situation. The four parameters of the position are independent. So we train a SVR model for each of the four labels by maximizing the following dual function [7]

$$\mathcal{L}_s = \sum_{i=1}^m \alpha_i^s z_s^{(i)} - \frac{1}{2} \sum_{i,j=1}^m \alpha_i^s \alpha_j^s K(q^{(i)}, q^{(j)}) \quad (2)$$

$\alpha_i \in [-t, t]$, $i \in \{1, 2, \dots, m\}$, $s = 1, 2, 3, 4$. m is the number of training samples. α_i^s 's are the optimization parameters. t is the penalty ratio of the difference between the predicted label and the real label. This dual function is obtained from the primary optimization problem [7] in the following equations by eliminating the original variables:

$$\mathcal{L}_s = \frac{1}{2} |\theta_s|^2 + t \sum_{i=1}^m \xi_i^+ + t \sum_{i=1}^m \xi_i^- \quad (3)$$

under the constraints that $\forall i \in \{1, 2, \dots, m\}$

$$\xi_i^+ \geq 0 \quad (4)$$

$$\xi_i^- \geq 0 \quad (5)$$

$$z_s^{(i)} - \theta^T q_s^{(i)} \leq \xi_i^+ \quad (6)$$

$$\theta^T q_s^{(i)} - z_s^{(i)} \leq \xi_i^- \quad (7)$$

where ξ_i^+ and ξ_i^- are the threshold for training error.

VI. TRAINING

A. Convolutional Neural Network

Nesterov momentum method [8] is utilized to minimize the CNN loss function. We use a vanishing learning rate which decays as t^{-1} :

$$\alpha = 10^{-4} \times (1 + t)^{-1} \quad (8)$$

B. Cross Validation

To maximize the usage of limited data, we use 9-fold cross-validation. First we randomly shuffle the 162 original images to erase the potential correlation between different images. Then divided them into 9 batches, each of 18 images. For each batch, the other 144 original images together with the related 14400 generated images form the training data set for the next two steps. This guarantees the independence between the training and testing process.

C. PCA and SVM

Sequential minimal optimization (SMO) algorithm is utilized to minimize the SVR loss function. Package *scikit-learn* is used throughout the process to guarantee the performance.

The feature has a huge dimension of 8192, while the prediction only contains 4 numbers. To prevent over-fitting, we use principle component analysis (PCA) to extract the most influence dimensions and iterate the training and testing procedure for 12 different dimensions equally distributed in *log-scale*, i.e. {5, 10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 8000}.

VII. RESULTS AND DISCUSSIONS

A. Performance

After re-tuning, the regression algorithm gives superior performance. Beam positions are corrected predicted in 139 of the 162 original images. The following diagram demonstrates the typical performance of the regression on the test set. When the regression algorithm catches the spot, it gives really precise prediction, which implies that the algorithm does have learned how to pinpoint the position of the spot.

Yet, when it can not catch the spot (picture in second column of Figure 4), the regression algorithm just randomly guess a point in the center of the screen. After checking the pictures where the algorithm miss the spot, we found that the missing spots are not dim at all. In fact the maximum point in those figures are even larger than those with perfection predictions. But the beam spots are all very small in those figures.

B. Effect of PCA dimension

To prevent overfitting, we use PCA to pick out the most influential dimensions of the features before regression. Figure 5 demonstrates the training error of the SVR for different PCA dimensions. The error is defined as the distance between the predicted beam center and the true beam center. Figure 6 demonstrate the performance of the SVR for different PCA dimensions. To quantify the performance, we use the average ratio between the overlapped area and the area of the true spot as the indicator of the performance.

From these two figures, it is clearly shown that when the PCA dimension exceeds 1000, there will be severe overfitting problem, as the training vanishes and the performance drops in that area.

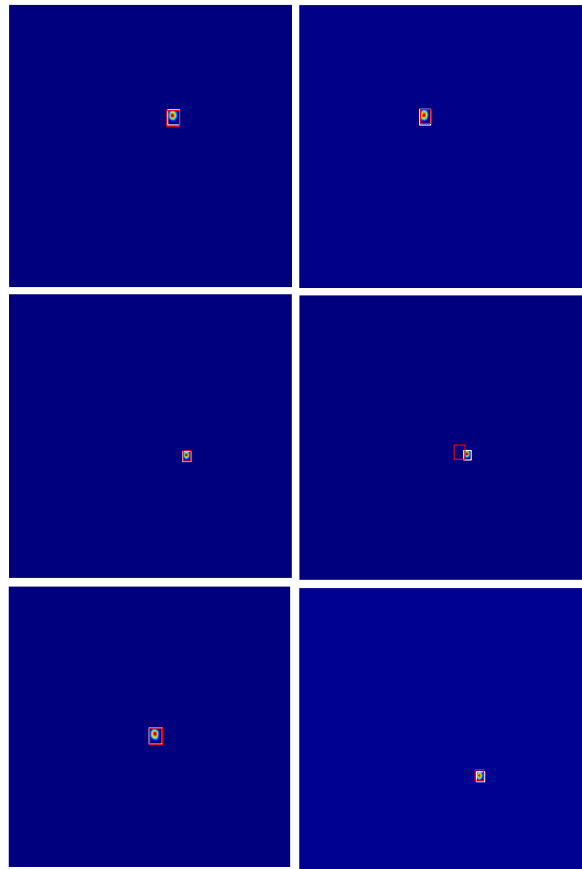


Fig. 4: Test Results

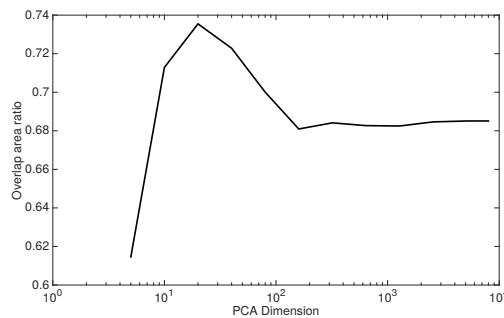


Fig. 5: Test performance for different PCA dimensions

VIII. CONCLUSIONS AND FUTURE WORKS

With newly generated figures and re-tuned neural network, the performance of the SVR gains great improvement. Yet it is still far from applicable in real environment. Three approaches are scheduled for further improvements.

A. Fine-Tuning Neural Network

VGG16-like neural networks are huge and hard to tune. It is highly likely that with improved tuning skill and a bit of luck and patience, we can get better performance.

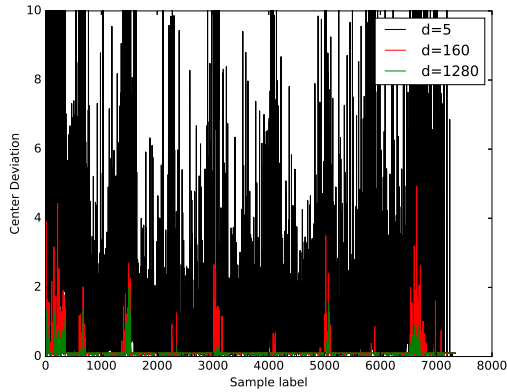


Fig. 6: Training error for different PCA dimensions

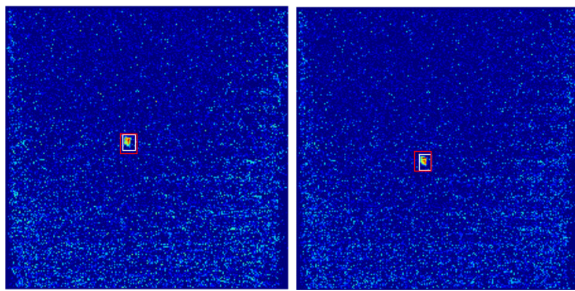


Fig. 7: Test results for raw pictures

B. More and Harder Data

Noisy figures are currently beyond the signal processing algorithm's capability. But as we can see sometimes our regression algorithm has done better job on the existing samples. And even for those raw pictures, our model can generate good prediction results occasionally (Figure 7). So we may try the algorithm on harder samples in the future. Especially we will try to mark the position of the beam by hand for noisy figures and try to train the model on those samples as well.

C. Building New Neural Network

It is possible that a smaller but better training neural network can do better job. Only after trying both methods can we decide which way is better. We have already constructed another smaller neural network. Training of this one will soon begin.

REFERENCES

- [1] Z. H. Kwang-Je Kim and R. Lindberg, *Synchrotron Radiation and Free-Electron Lasers*. Cambridge University Press, 2017.
- [2] C. Pellegrini, A. Marinelli, and S. Reiche, "The physics of x-ray free-electron lasers," *Rev. Mod. Phys.*, vol. 88, p. 015006, Mar 2016. [Online]. Available: <http://link.aps.org/doi/10.1103/RevModPhys.88.015006>

- [3] "Spatial Localization - AI at SLAC - SLAC Confluence." [Online]. Available: <https://confluence.slac.stanford.edu/display/AI/Spatial+Localization>
- [4] "CS231n Convolutional Neural Networks for Visual Recognition." [Online]. Available: <http://cs231n.github.io/>
- [5] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," sep 2014. [Online]. Available: <http://arxiv.org/abs/1409.1556>
- [6] "VGG in TensorFlow · Davi Frossard." [Online]. Available: <https://www.cs.toronto.edu/frossard/post/vgg16/>
- [7] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.
- [8] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in Optimizing Recurrent Networks," dec 2012. [Online]. Available: <http://arxiv.org/abs/1212.0901>