First Quarter Rotation Summary

Beam Detection with Machine Learning Methods
Haoyuan Li
hyli16@stanford.edu

December 18, 2016

1 Introduction

1.1 Project Purpose

This project aims to develop an automatic and general way to locate the beam spot on the screen.

1.2 Current strategy

Use a convolutional neural network as a feature extractor. Use those features to train a regression program which will give the position of the beam.

1.3 Current Status

For preprocessed images, this strategy behaves well. 138 out of 162 test samples obtain precise predictions. There is no obvious reason why the other 24 fail which always fail for any methods I tried this quarter.

For raw images, this strategy has terrible performance. It is merely better than random guess.

2 Results Analysis

2.1 Results For Processed Images

Fig 1 is a typical performance of the strategy. I randomly pick up 8 images. It seems that all of them have really great performance.Some of them may have even better prediction than the original "ground truth".

But there are also situations where the program can not catch the beam spot.Fig 2 gives a subset of samples that this strategy fails. It can seen that they do not merely fail. Actually,

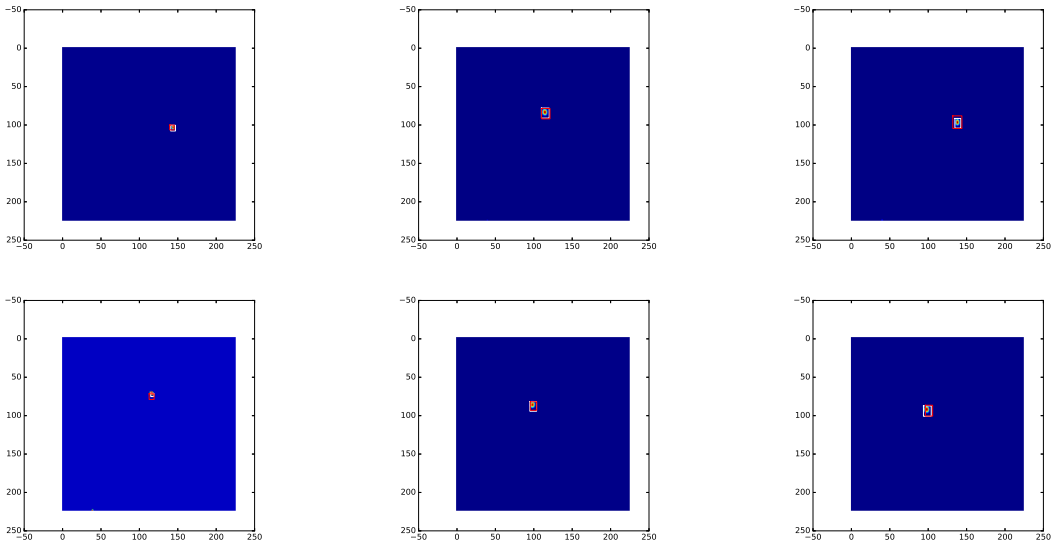

Figure 1: Test Results

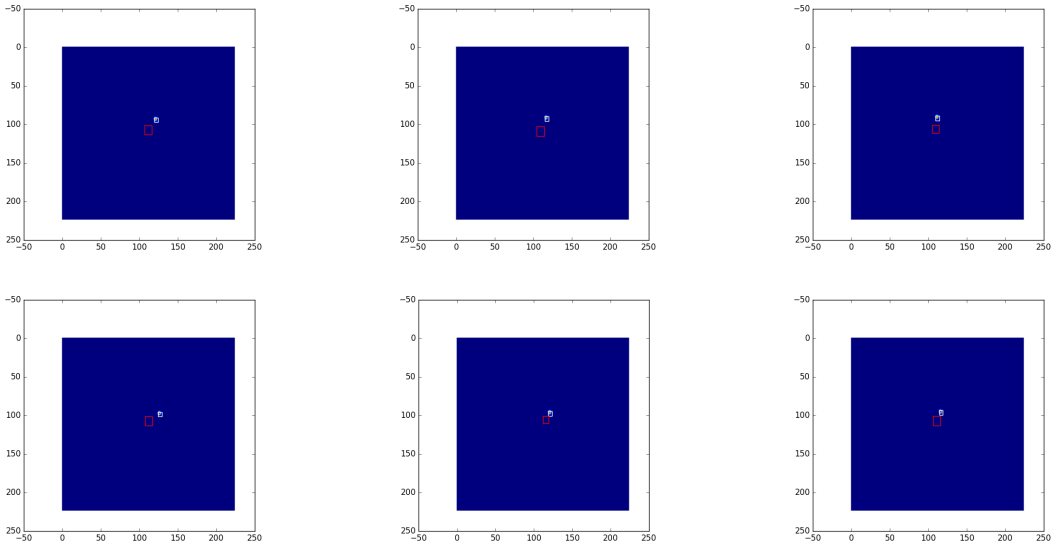

Figure 2: Test Results

it seems that the algorithm believe that there is nothing in the screen so it just pick out the center of the screen. I have checked out whether it is because the beam spot in these pictures are extremely dim or not. It turns out that the beam in these diagrams have even larger intensity than some of the diagrams in Fig 1.

Because of the high dimension of the feature space, I suspect that if I directly train SVM with the features, there may be risk even after I generate several thousands of new pictures. Fig 2 demonstrates the training error and test errors for different PCA dimensions. For $d =$ in the diagram indicates the PCA dimensions. The right hand images demonstrate the average

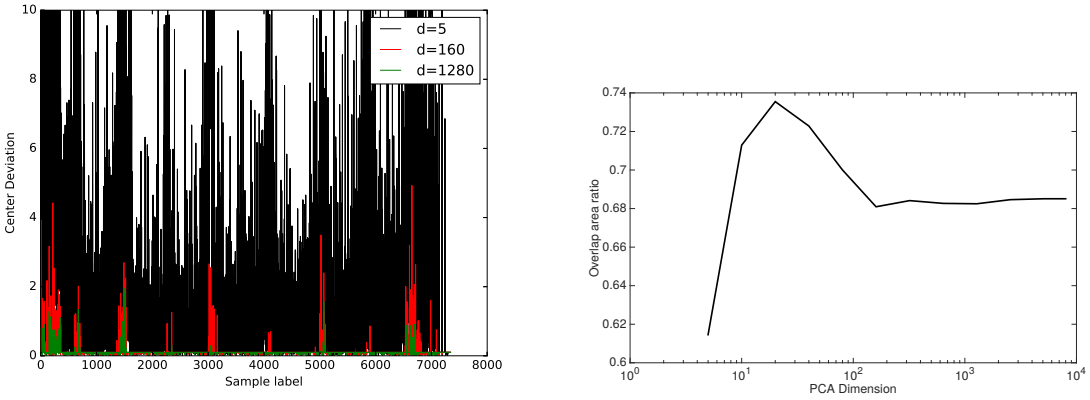

Figure 3: Training Error for Different PCA Dimensions (Left); Overlap Ratio for Different PCA Dimensions (Right)

ratio between the overlapped area and the ground truth's area. It seems that in all the 12 different dimensions I inspect (5,10,20,40,80,160,320,640,1280,2560,5120,8000), $d = 20$ has the optimal performance. Latter it seems that such fact also holds for raw images. Even though the ratio seems not very high, it is mainly due to the fact that there are about 15 images the program always fails. According to the behavior of the training error and overlap ratio, over-fitting does occurred when the PCA dimension is too high.

2.2 Results For Raw Images

The results for the raw images implies that our approach may be fundamentally questionable. Diagrams in Fig 4 are exactly those in Fig 1 without signal processing.

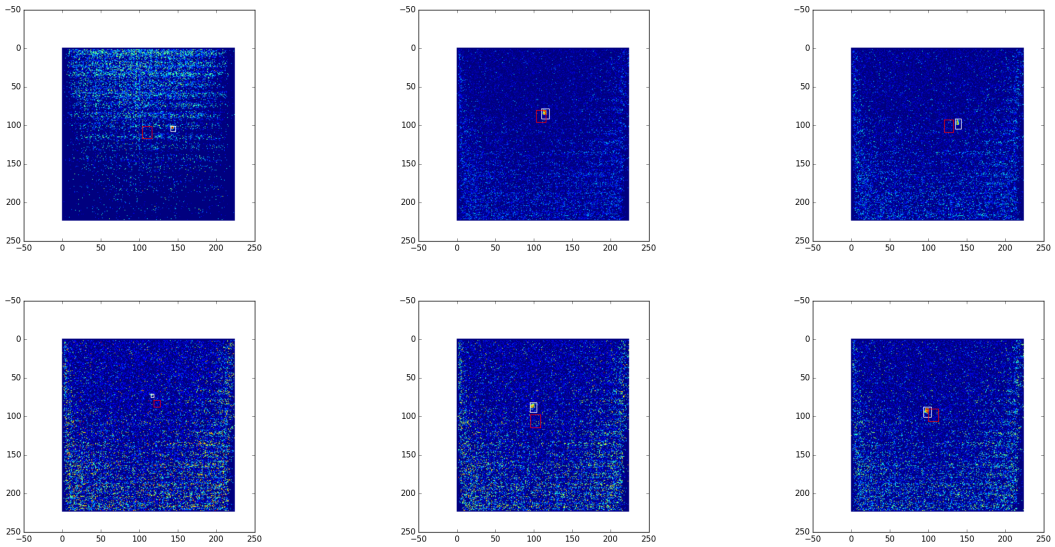

Figure 4: Test Results

The predictions can not be worse. However, the training error and testing performance show something interesting.

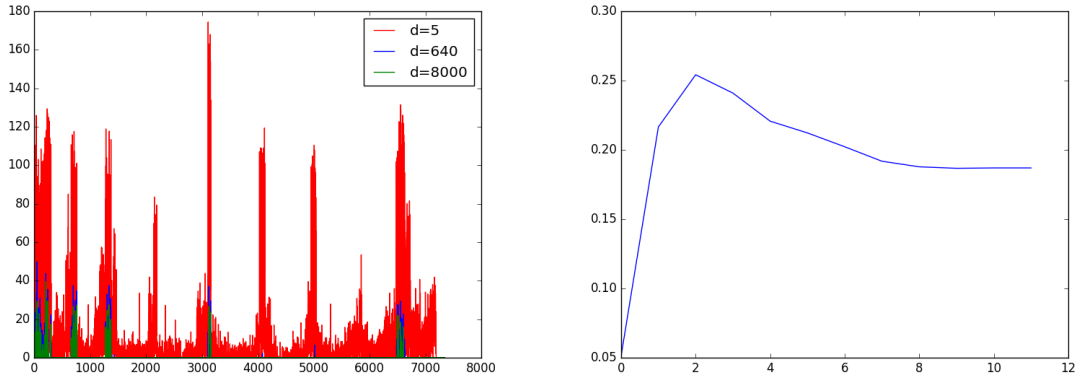

Figure 5: Training Error for Different PCA Dimensions (Left); Overlap Ratio for Different PCA Dimensions (Right)

As is indicated in the diagram. It seems that there are some images with huge training error for all PCA dimensions. But when we only consider the overlap ratio, it seems that for large PCA dimension, we still have over-fitting problem.

To echo the corresponding diagrams in the former section, we include the same diagrams in Fig 2 here, except that this time, both the prediction and images are not pre-processed.

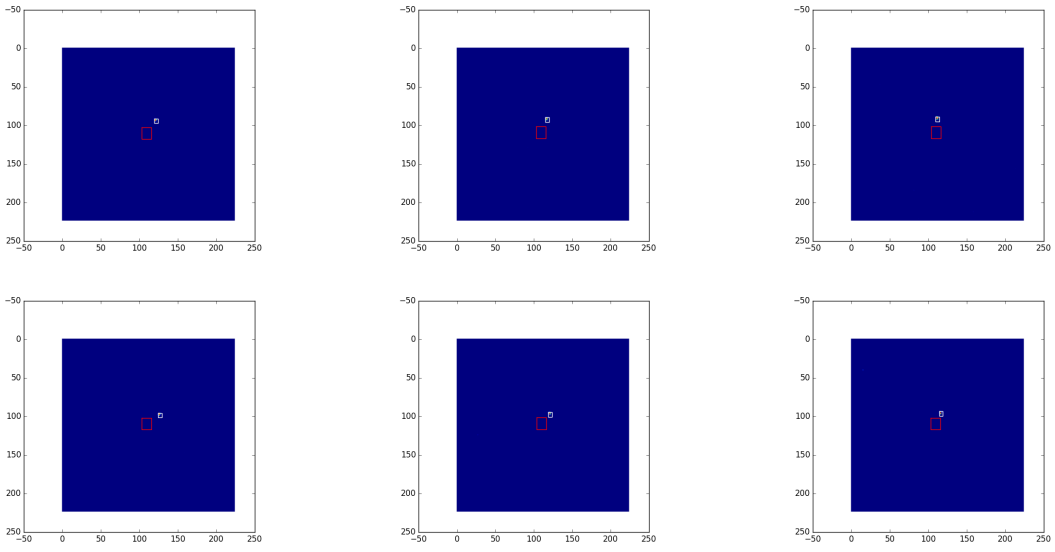

Figure 6: Test Results

Surprisingly, in both case, these images look nearly exactly the same and the predictions are wrong in the same way.

3 My Understanding

3.1 Why It Does Not Work When Beams Are Dim

Our strategy only consider rough relations between different images. But what we need to pinpoint the beam when it is very dim are exact relations between different images.

To see why we need exact correlations, let's look at the following several diagrams.

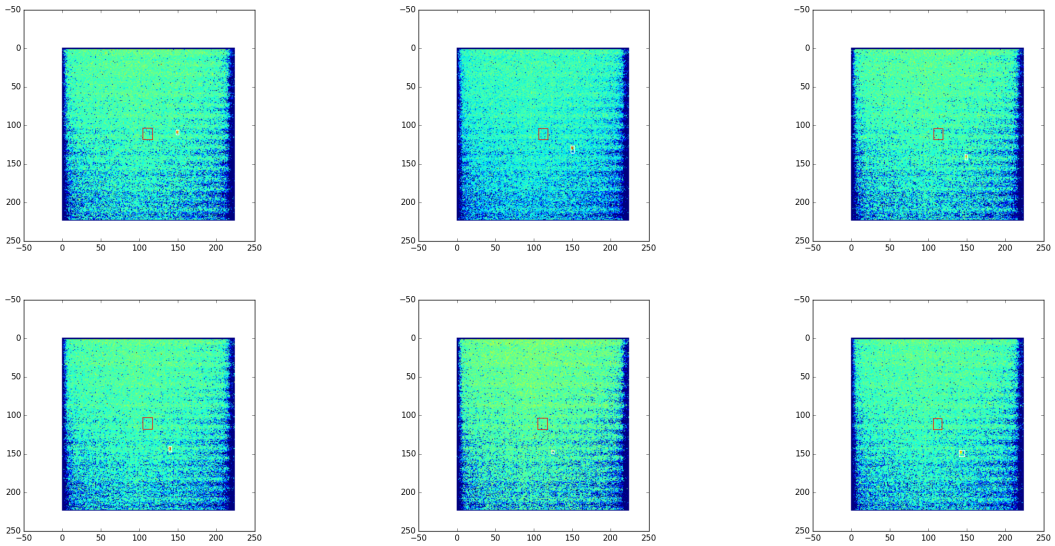

Figure 7: Test Results

I believe the most direct feeling of readers must be:*I won't be able to find the beam spot. Why should the program decide that spot to be the beam?*

The key to find the beam spot is exactly the exact relations between different images. Human have to do the background subtractions because the background is an essential part of information. There is no way to find the beam without knowing the background.

This is kind of weird at first, as in diagram 7, it is apparent to human where the beam spot is. But with a little bit of consideration, it becomes clear that this instinct is biased: what if the reality is that there is actually no beam and that the bright spot appearing to be a beam is actually a detector flaw?

So to know the exact position of the beam spot, we always need to know the background. That is kind of previous knowledge of the inspected system. Sometimes we can use zero background assumption. But for dim beam spot, we can not assume this.

Thus if the program can not find the exact relation itself, then it losses an essential part of information. It failure is then understandable.

3.2 Why It Fails When Beams Are Bright

I believe that this is because the VGG16 just does not recognize the beam spot as an object. To fix this, we have to retrain the VGG16. But it seems that I won't have chance to do that.

4 Technical Details

- The SVM used in this report are trained with the original 162 images and the 16200 related images.
- The results in Fig 1 and Fig 2 are trained with PCA dimension 1280. But actually for other dimensions, the results look nearly the same.
- Codewords in this report are generated by VGG16 via GPU.

5 Gratitude

Thanks so much for your assistance and guidance, Daniel and David! It's a pity that I am not able to finish this project this quarter, but I have learned a lot. Thanks!