COMP4211 PA2

20316792

Shayan Imran

2019

Contents

[CNN Classifiers 2](#_Toc5845631)

[Code for CNN Classifiers 2](#_Toc5845632)

[Class for CNN Classifier 2](#_Toc5845633)

[CNN from Scratch and PT Train 3](#_Toc5845634)

[Run CNNs 4](#_Toc5845635)

[Validation Results 6](#_Toc5845636)

[Training from Scratch 6](#_Toc5845637)

[Training with Pretrained Weights 7](#_Toc5845638)

[Analysis of Training from Scratch 8](#_Toc5845639)

[Test Results 8](#_Toc5845640)

[Test Metrics for Training from Scratch 8](#_Toc5845641)

[Test Metrics for Training from Pre-trained Encoder 8](#_Toc5845642)

[Test Curve for Training from Scratch 9](#_Toc5845643)

[Test Curve for Training from Pre-trained Encoder 12](#_Toc5845644)

[CNN Classifiers Analysis 15](#_Toc5845645)

[CAE with Pretrained Encoder 16](#_Toc5845646)

[Code for CAE Decoder 16](#_Toc5845647)

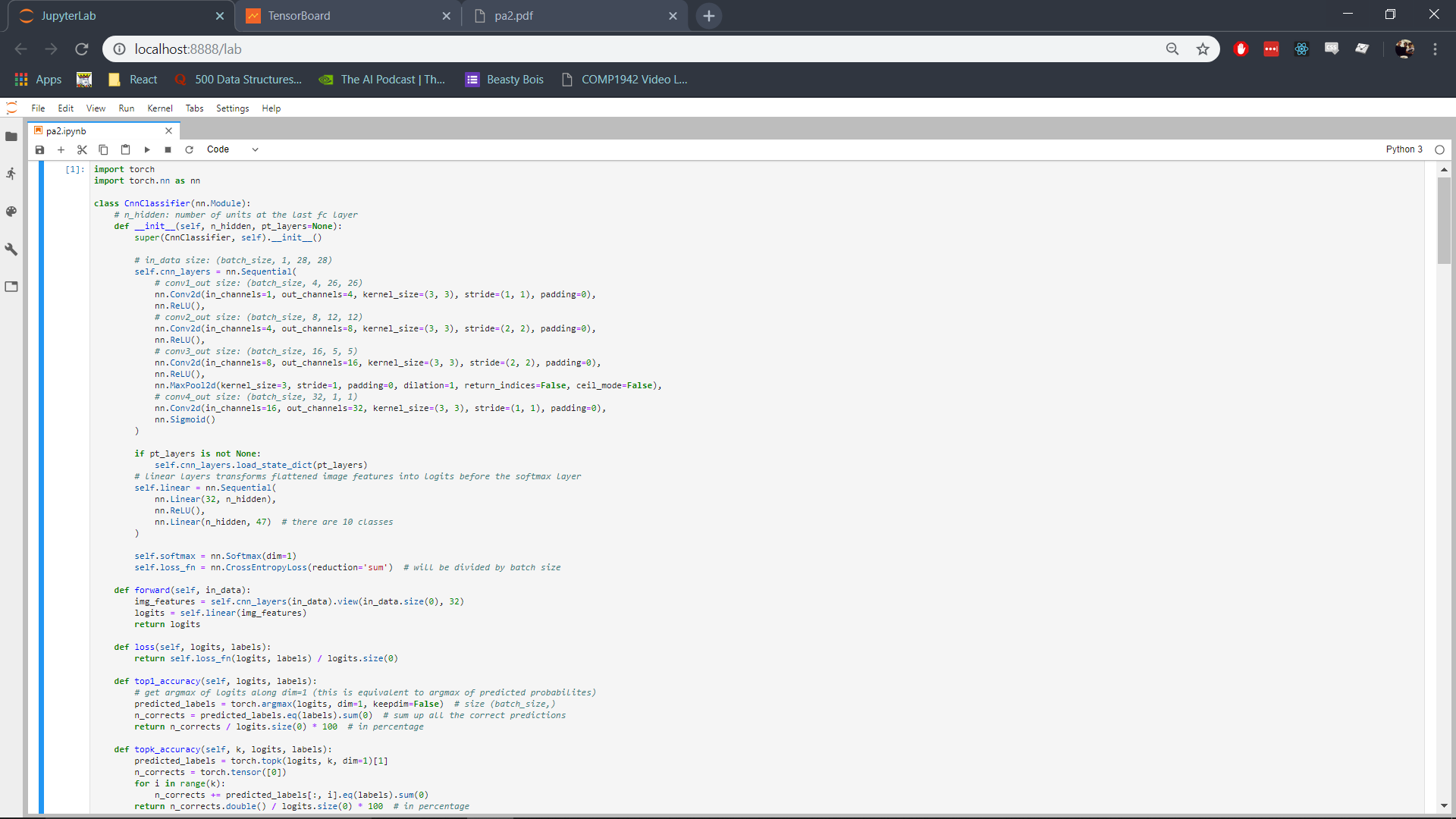
[Validation Results 19](#_Toc5845648)

[Test Results 20](#_Toc5845649)

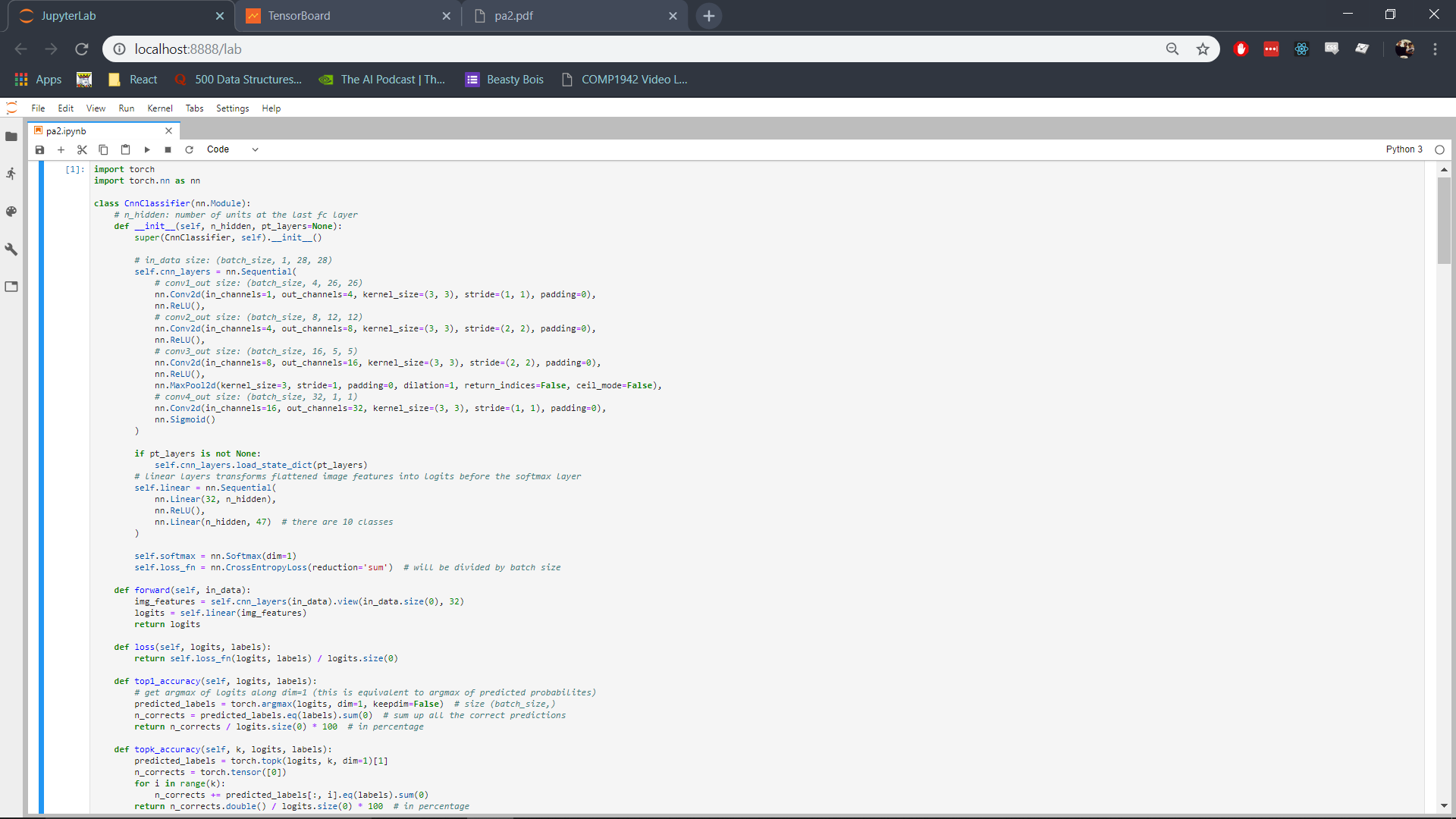
# CNN Classifiers

## Code for CNN Classifiers

### Class for CNN Classifier

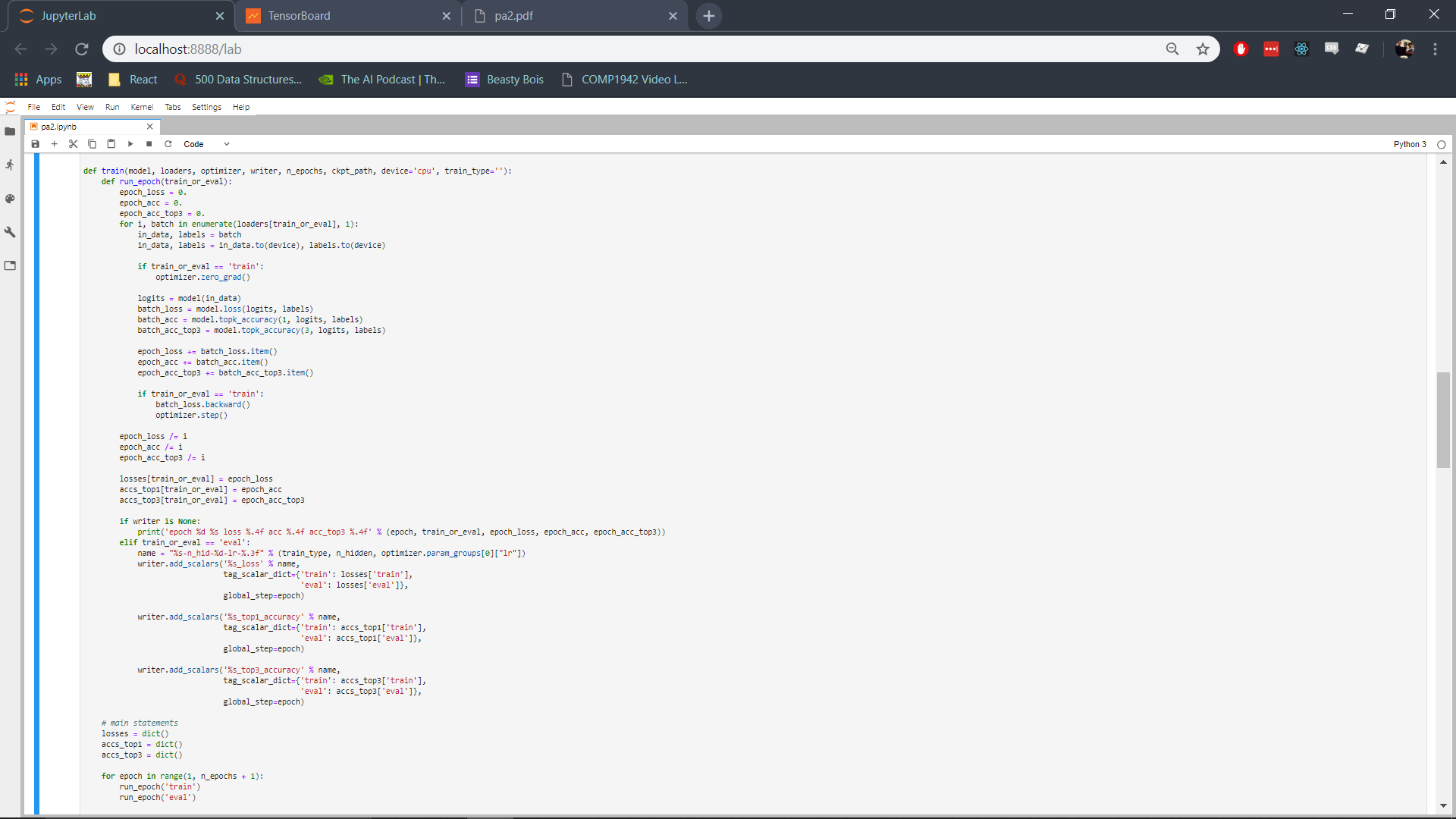


This class is used for making a classifier from scratch and to initialize the weights in case a pretrained network’s state\_dict is supplied in the constructor as the pt\_layers argument. The weights are loaded using the load\_state\_dict function of the nn.Sequential class as show below.

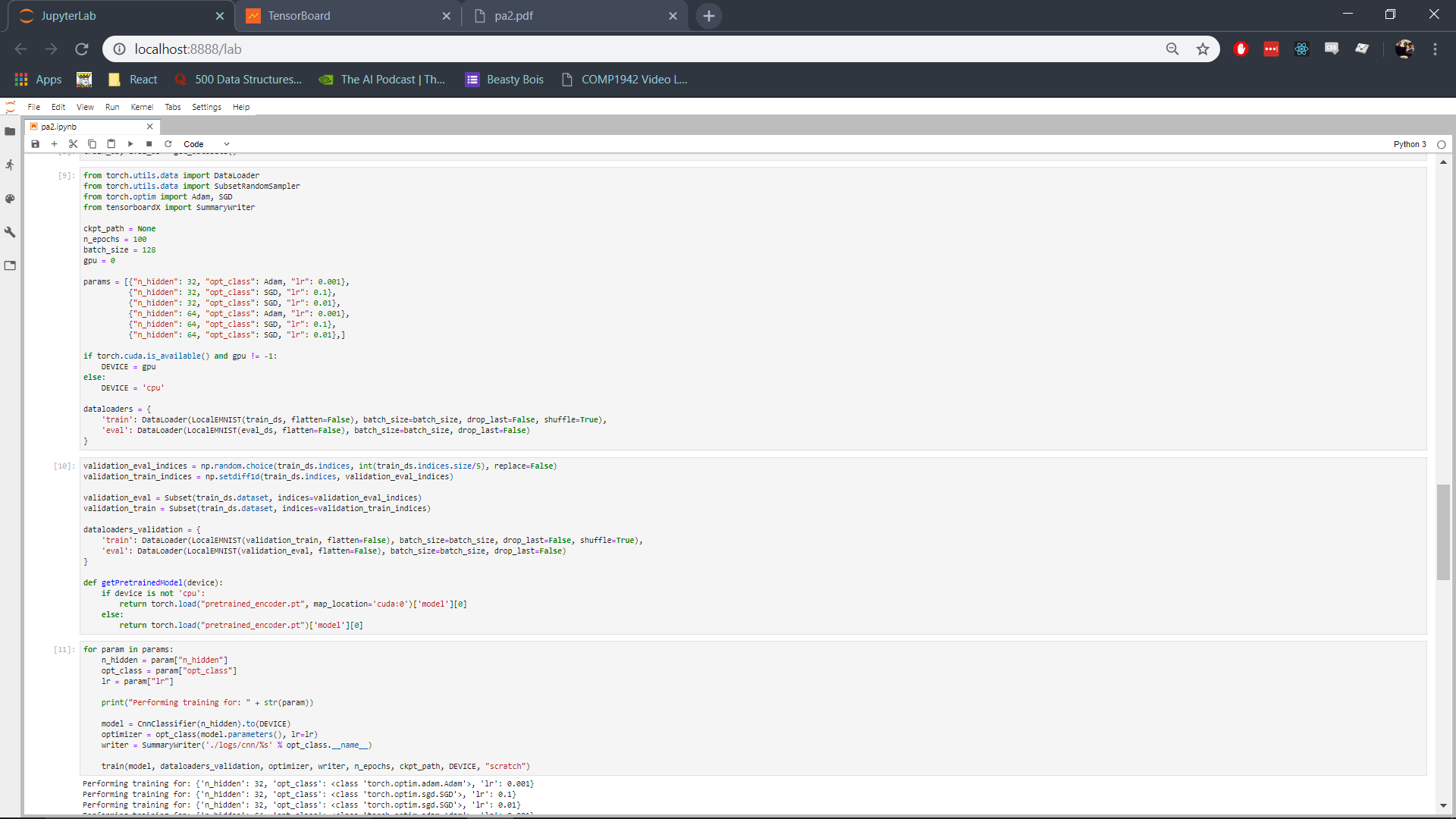


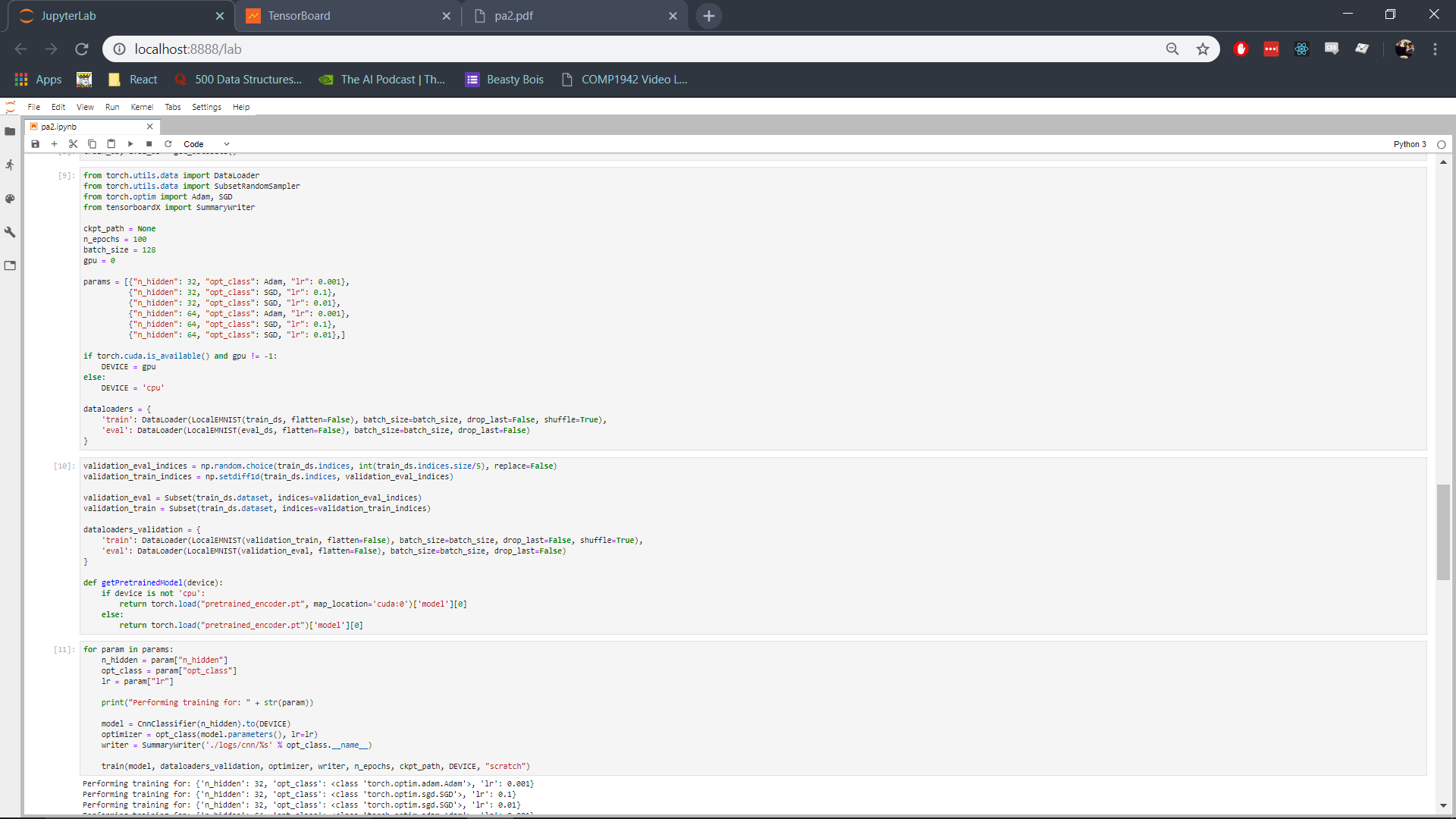
More details for loading the state\_dict from the pretrained layers are given below.

### CNN from Scratch and PT Train



### Run CNNs









## Validation Results

### Training from Scratch

|  |  |  |  |
| --- | --- | --- | --- |
| ***H -* Optimizer** | 32 - Adam | 32 - SGD | 32 - SGD |
| **Learning Rate** | 0.001 | 0.01 | 0.1 |
| **Optimal Loss** | 0.6061 | 3.85 | 0.5471 |
| **Loss** |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| ***H -* Optimizer** | 64 - Adam | 64 - SGD | 64 - SGD |
| **Learning Rate** | 0.001 | 0.01 | 0.1 |
| **Optimal Loss** | 0.5474 | 1.537 | 0.5532 |
| **Loss** |  |  |  |

### Training with Pretrained Weights

|  |  |  |  |
| --- | --- | --- | --- |
| ***H -* Optimizer** | 32 - Adam | 32 - SGD | 32 - SGD |
| **Learning Rate** | 0.001 | 0.01 | 0.1 |
| **Optimal Loss** |  |  |  |
| **Loss** |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| ***H -* Optimizer** | 64 - Adam | 64 - SGD | 64 - SGD |
| **Learning Rate** | 0.001 | 0.01 | 0.1 |
| **Optimal Loss** |  |  |  |
| **Loss** |  |  |  |

### Analysis of Training from Scratch

From the results of the validation stage the best hyperparameters must be chosen. The SGD optimizer with a learning rate of 0.01 can be easily discarded as, from the loss curves for both *H* values, it can be seen that this setting does not perform well. When 32 is set for the value of *H* the network is unable to meaningfully train, with loss staying at its initial value of 3.85 even after 100 epochs of training. With 64 hidden layer units the model performs better but the training is still unpredictable; it stays at a loss value of 3.86 for around 80 epochs and then it finally decreases to 1.5 at the hundredth epoch. It is possible that the loss would have decreased even more if trained for more than 100 epochs but it taking this long to even start decreasing in loss while other parameters perform much better already makes using it infeasible. The best validation loss value for 32 *H* is 3.85 and for 64 *H* it is 1.537.

SGD works much faster with a learning rate of 0.1, the loss remains steady at 3.85 for 20 epochs when using 32 for *H* and for only 10 epochs when using 64 for *H* after which it decreases in both cases to a final value of around 0.5. While this is much better than the previously mentioned setting, it is outperformed by the final hyperparameter setting which used Adam. The best validation loss value for 32 *H* is 0.5471 and for 64 *H* it is 0.5532.

The Adam optimizer with a learning rate of 0.001 clearly outperforms the other settings with it resulting in an immediate decrease in the value for loss, reaching a value under 1 in just 5 epochs, and becoming steady at around 0.5 in 40 epochs. Both values for *H* perform very similarly, however 64 performs marginally better with its best loss value 0.5474 while the best loss value for 32 *H* is 0.6061. While it may seem that SGD with 32 *H* and a learning rate of 0.1 is better with a minimum loss of 0.5471, it is less reliable as the loss only starts decreasing after 20 epochs, and its loss values are also less consistent, which is why the best parameters to use for the entire training set are the Adam optimizer with 64 hidden layer units and a learning rate of 0.001.

## Test Results

### Test Metrics for Training from Scratch

|  |  |  |
| --- | --- | --- |
|  | Mean | Standard Deviation |
| Cross Entropy Loss | 0.572201 | 0.006653 |
| Top 1 Accuracy | 81.245332 | 0.349645 |
| Top 3 Accuracy | 96.581871 | 0.069623 |

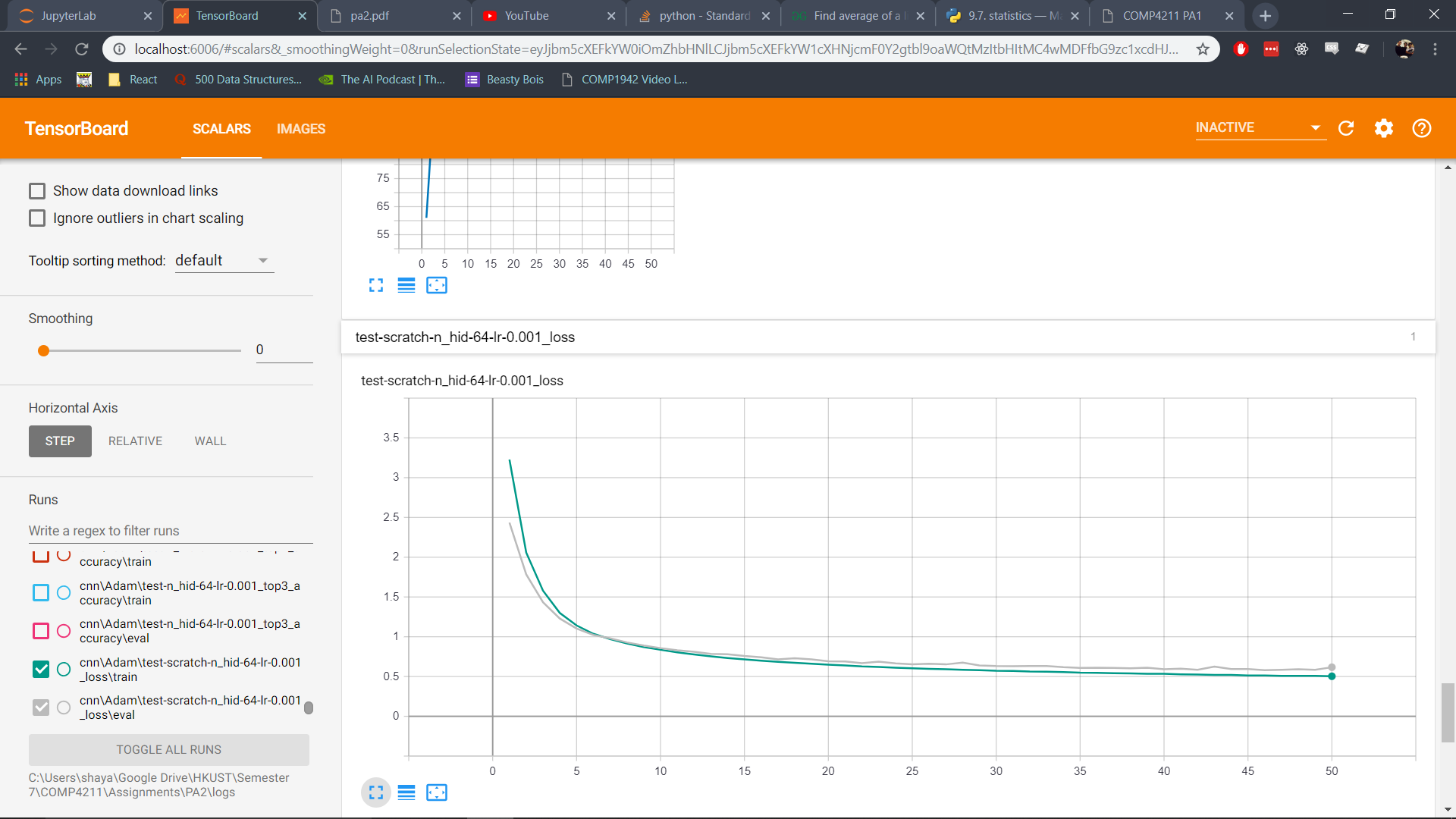
### Test Metrics for Training from Pre-trained Encoder

|  |  |  |
| --- | --- | --- |
|  | Mean | Standard Deviation |
| Cross Entropy Loss | 0.579712 | 0.011358 |
| Top 1 Accuracy | 81.538027 | 0.190894 |
| Top 3 Accuracy | 96.524849 | 0.052576 |

### Test Curve for Training from Scratch

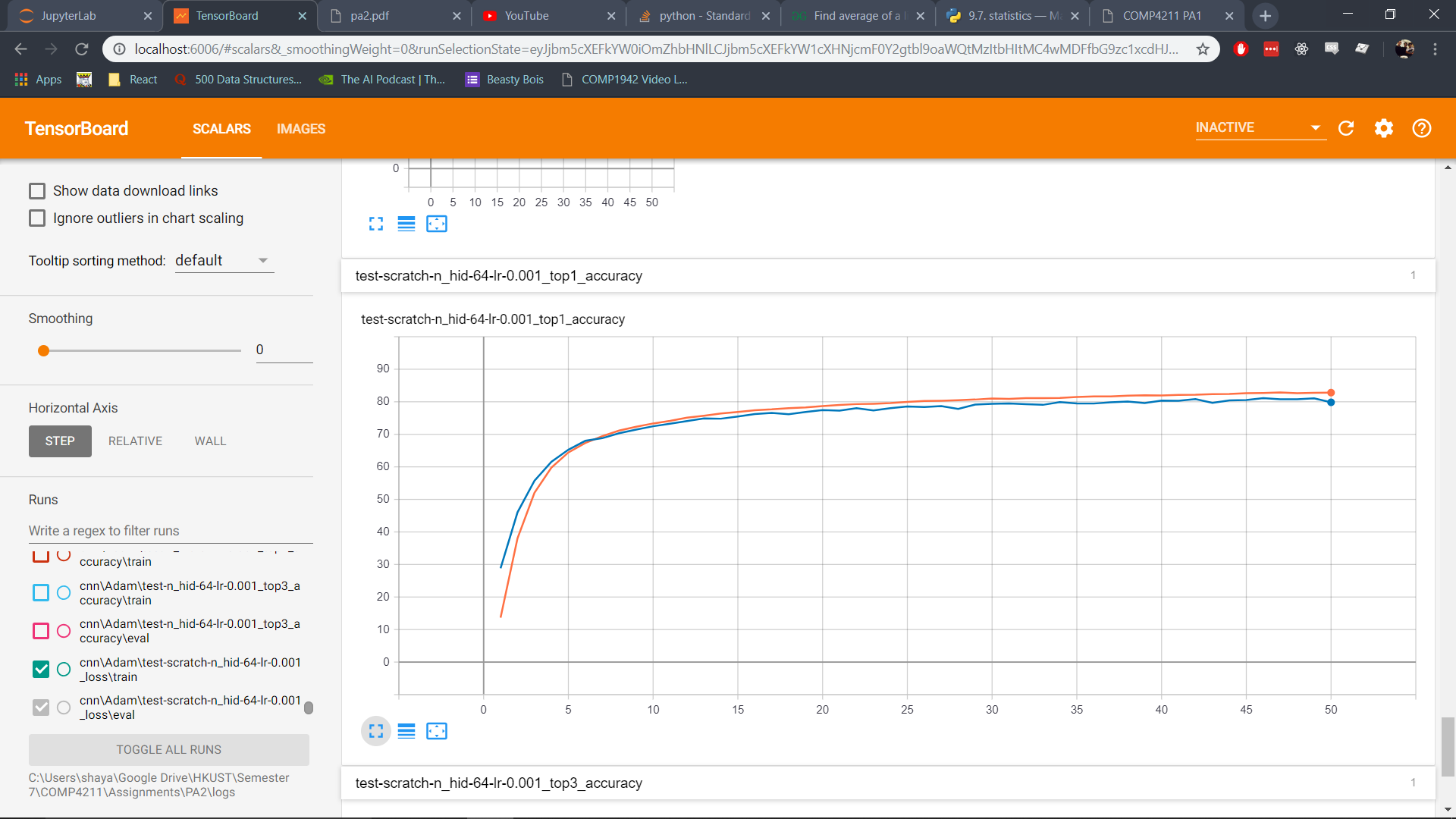
#### Loss

Train: Green  
Test: Grey



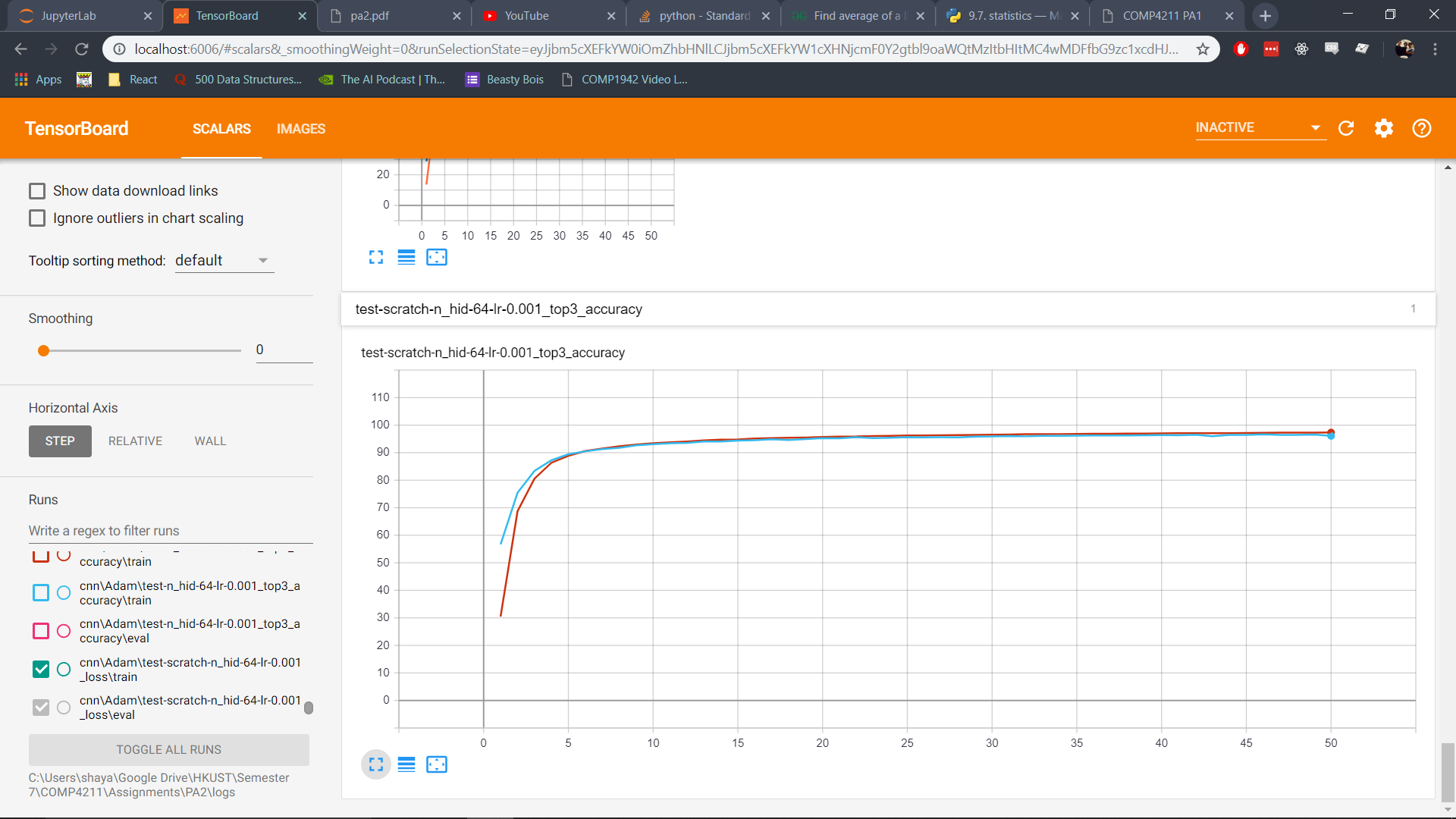
#### Top 1 Accuracy

Train: Orange  
Test: Blue



#### Top 3 Accuracy

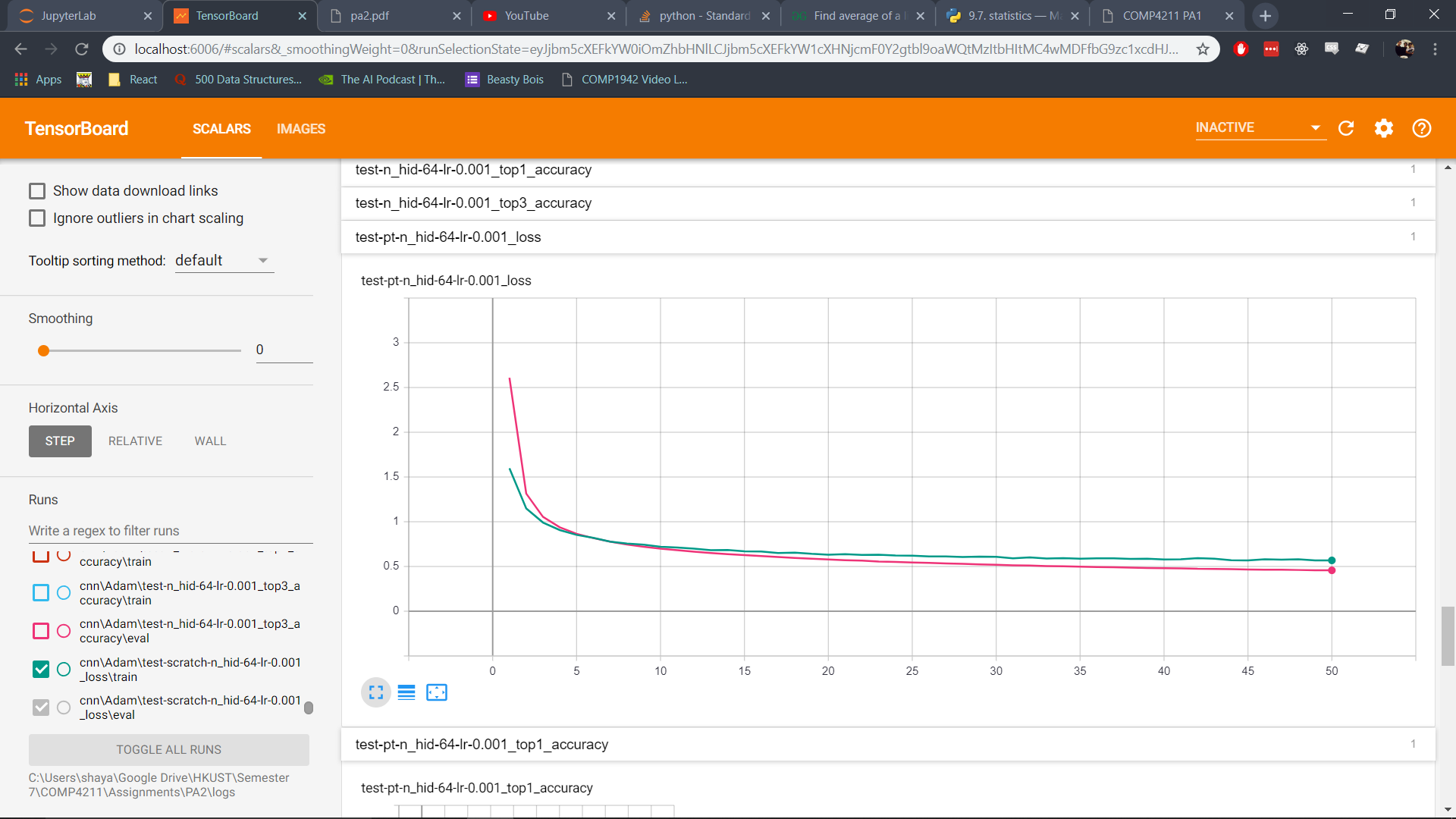
Train: Red  
Test: Blue



### Test Curve for Training from Pre-trained Encoder

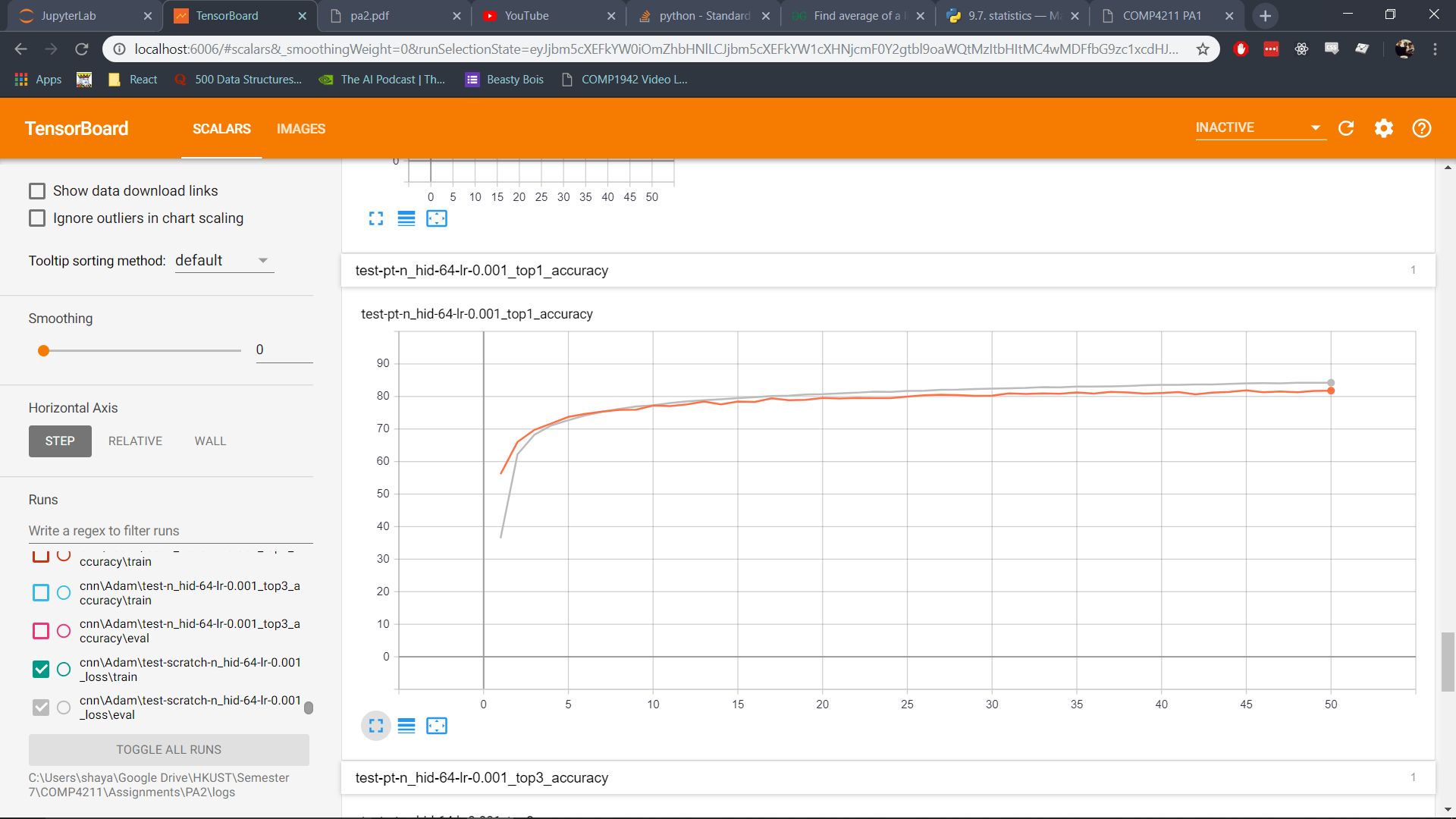
#### Loss

Train: Pink  
Test: Green



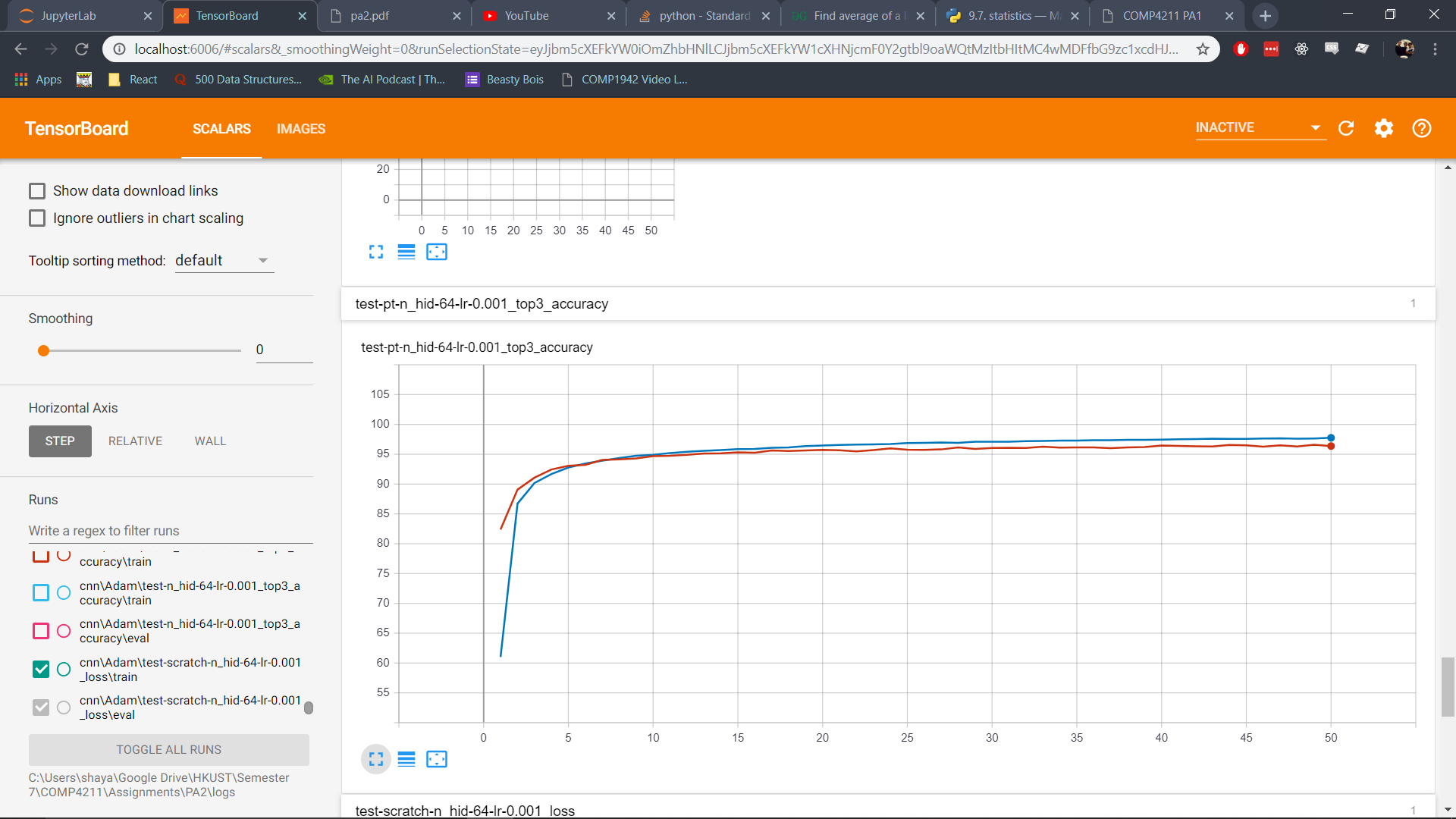
#### Top 1 Accuracy

Train: Grey  
Test: Orange



#### Top 3 Accuracy

Train: Blue  
Test: Red

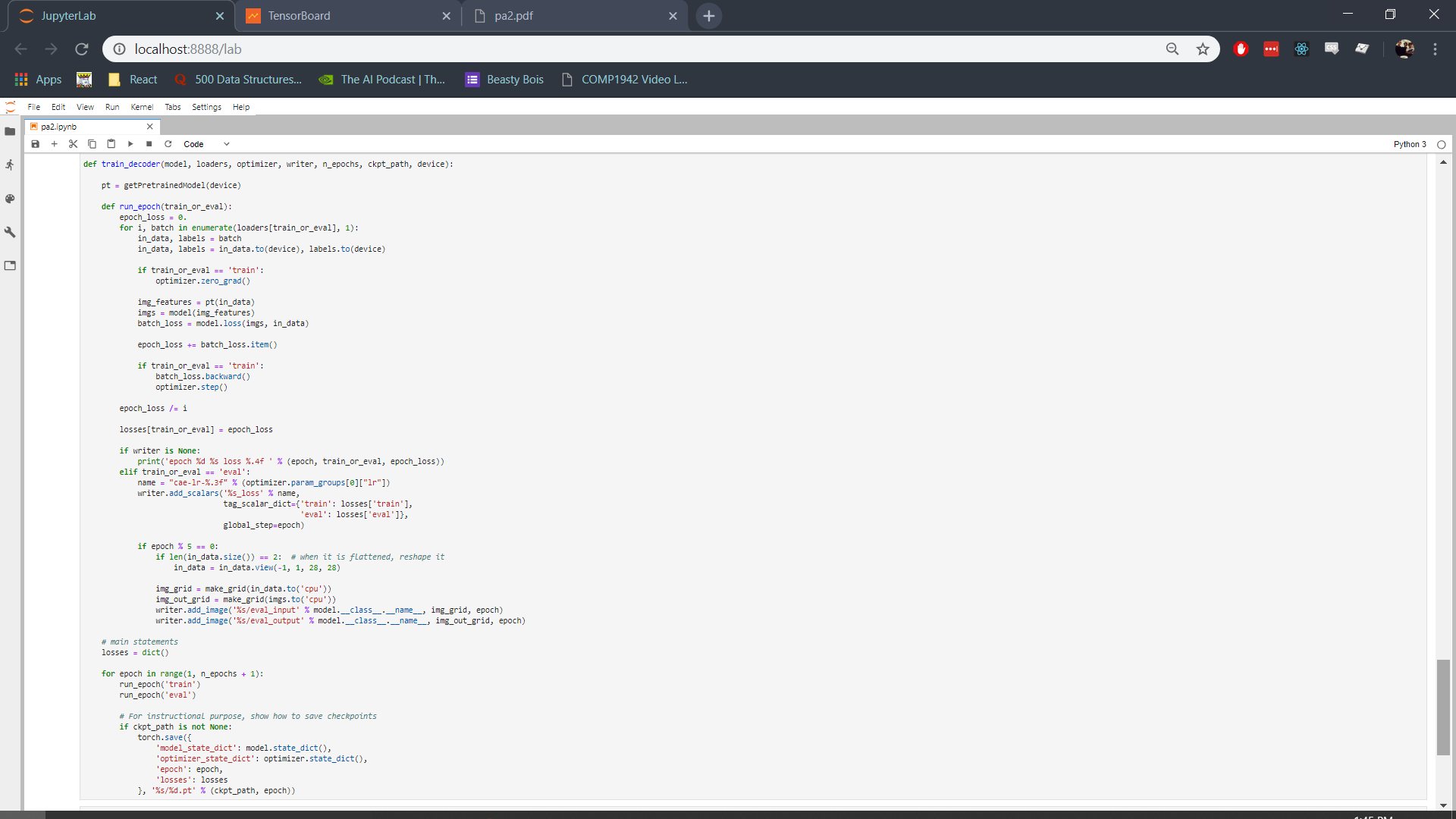


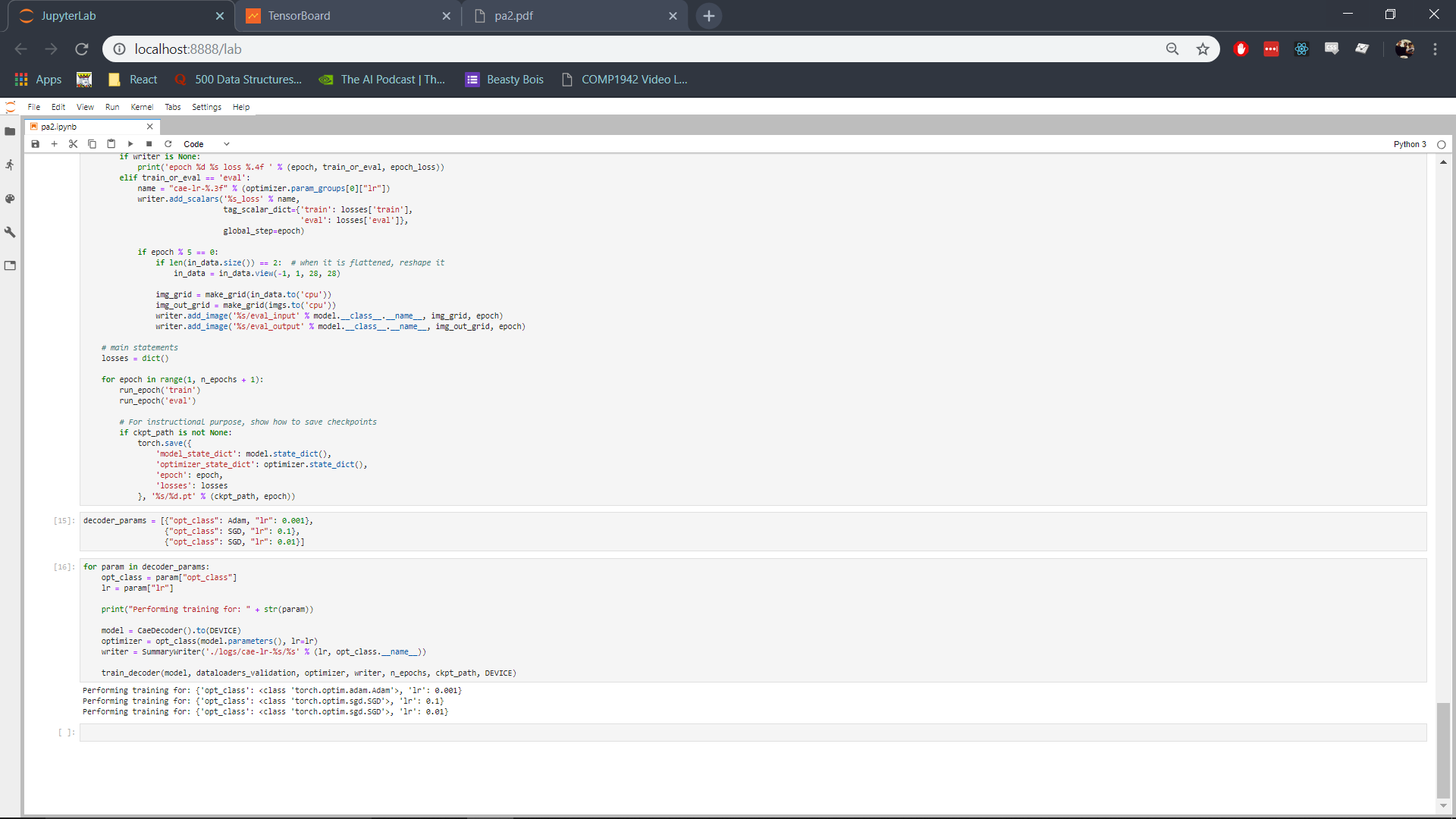
## CNN Classifiers Analysis

CAE with Pretrained Encoder

## Code for CAE Decoder

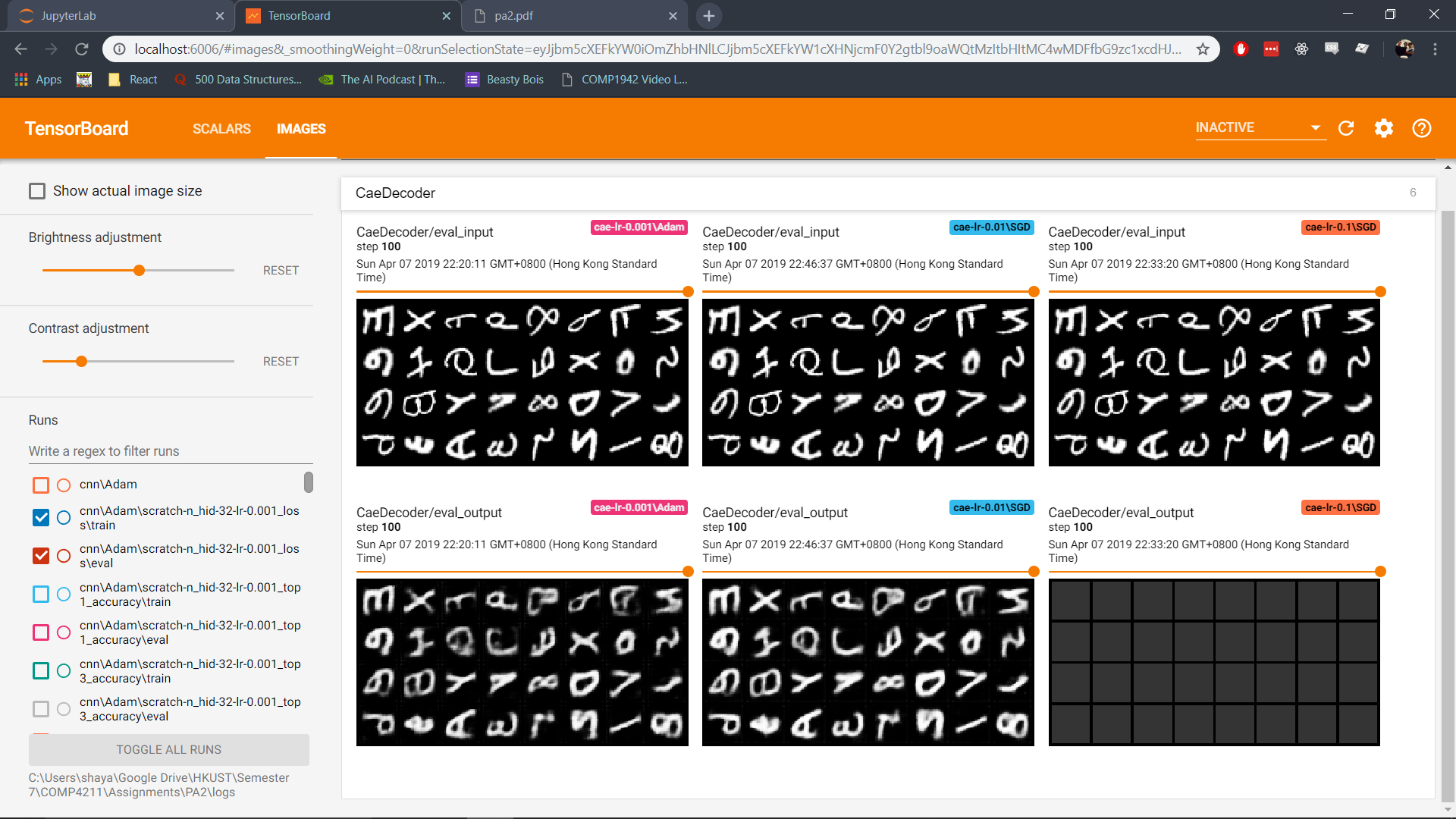






## Validation Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Optimizer** | Adam | SGD | SGD |
| **Learning Rate** | 0.001 | 0.01 | 0.1 |
| **Optimal Loss** |  |  |  |
| **Loss** |  |  |  |
| **Decoder Output** |  |  |  |



## Test Results