Using a small dataset causes any complex classifier to over fit the data and produces bad validation accuracy.

To be able to utilize state-of-the-art classifiers, we split the data into training and validation set, then augmented the training data into a set of 3000 images.

The augmentation process takes the training images and produces new ones with variations in rotations, shifts, flipping, zooming and sheering.

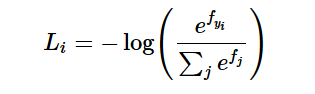
Using the augmented data along helped the classifiers generalize more.

1. **Softmax Classifier:**

Softmax classifier is a linear classifier with cross-entropy loss function.   
The score of the image (wx+b) is the weighted sum of its features. To control the parameters of the classifier we use cross entropy loss which treats the scores as the un-normalized log probabilities.

Exponentiation of these quantities gives the (un-normalized) probabilities, and the division performs the normalization so that the probabilities sum to one. In the probabilistic interpretation, it minimizes the negative log likelihood of the correct class, which can be interpreted as performing *Maximum Likelihood Estimation* (MLE).

In summary the classifier tries to get the minimum cross entropy between the true probability distribution and the estimated probability distribution.



We performed several experiments using different parameters:

First without extracting features, the classification task was very hard and the classifier failed to reach accuracy beyond 70%.

Then we used the extracted **HOG** features along with parameters:

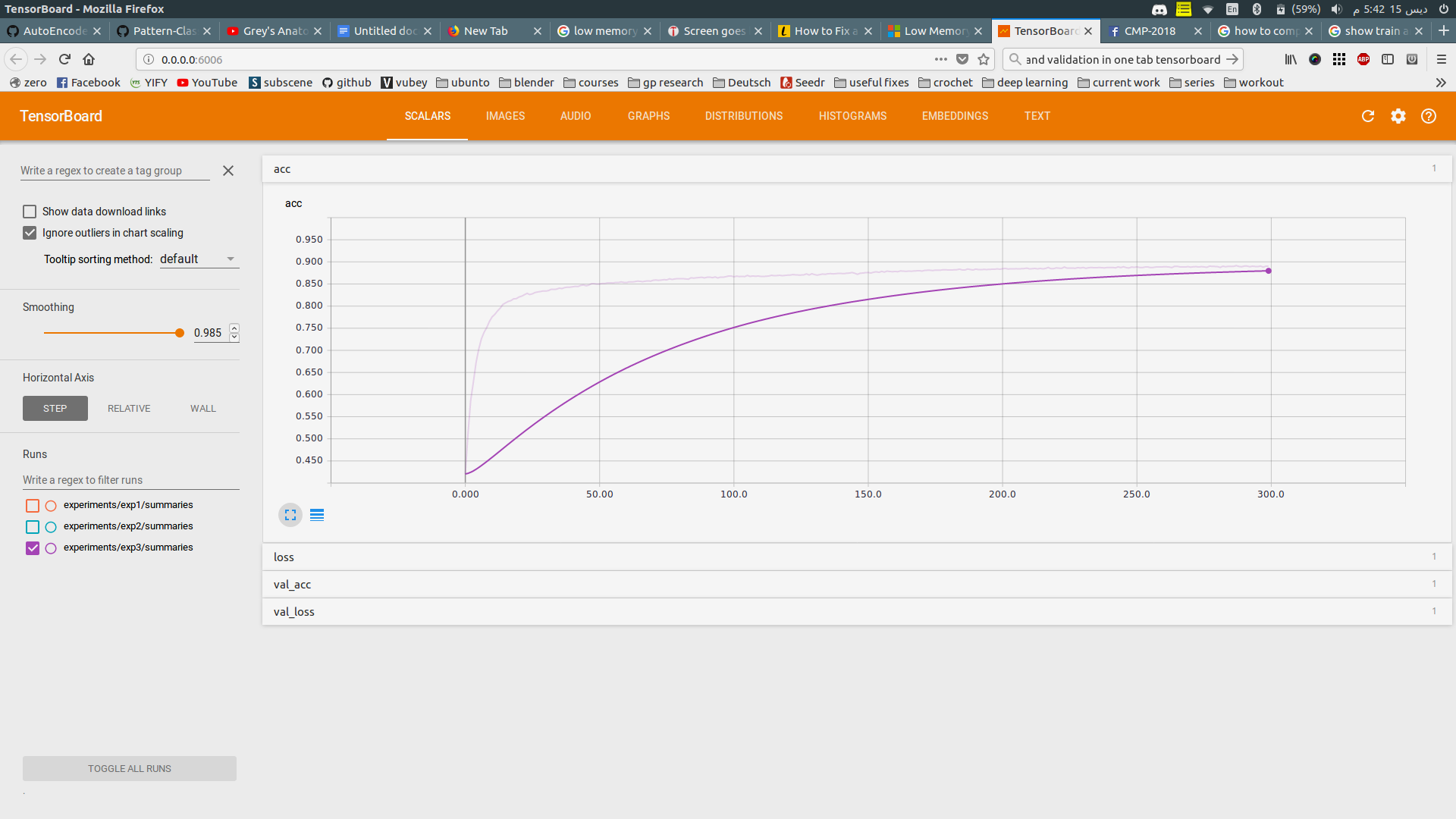
* Learning rate: 8e-3
* Regularization factor: 5e-4
* Stochastic gradient descent learning

Which resulted in:

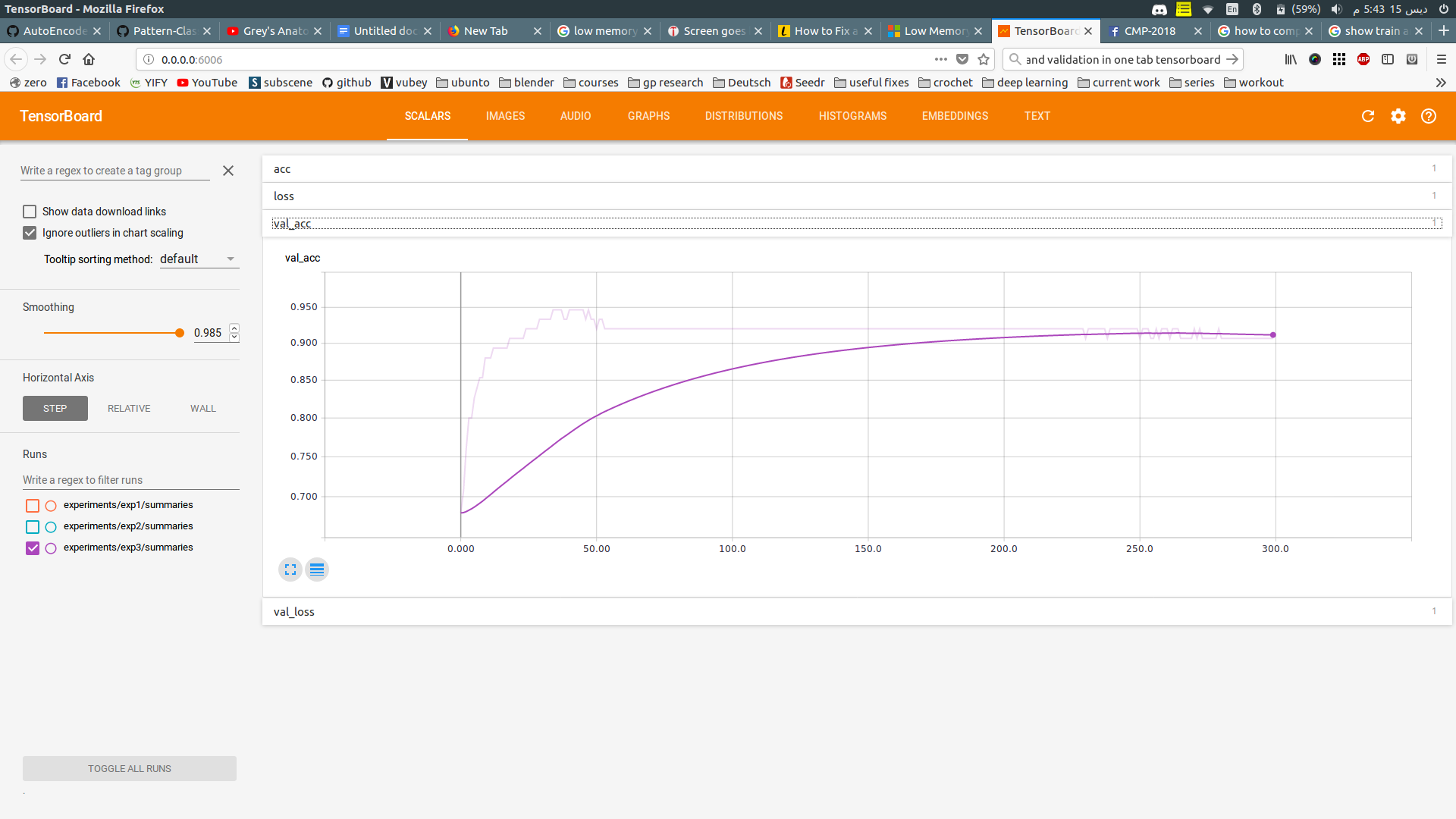
* **Training accuracy:90%**
* **Validation accuracy:89%**

learning Curves

Training accuracy:



Validation Accuracy:



1. **Fully connected neural networks:**

Neural networks are built on simple computational units called neurons followed by a nonlinearity which determines if the neuron exceeded a certain threshold or not (if it will fire).

We made a 5-layer architecture, along with dropout and regularization to prevent overfitting.

Using This architecture after using large number of features extracted from HOG, the accuracy didn’t exceed 90%

Reducing the number of features caused a large boost in accuracy.

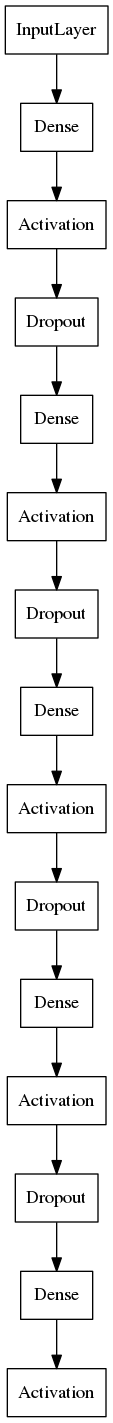
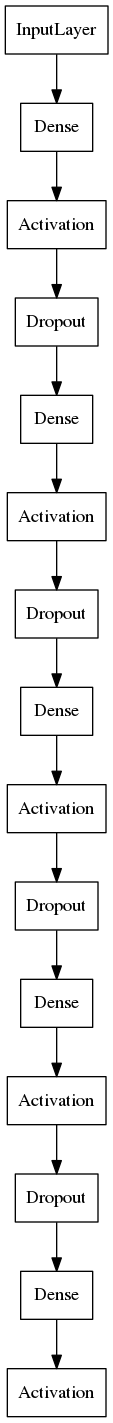
After performing multiple experiments and different parameters the best experiment parameters were:

* Number of neurons in each layer: 512 - 256 - 64 - 32 -3
* Adam learning with learning rate 1e-3
* Dropout of 0.2 after each layer.
* ReLu activation function
* Regularization factor:5e-3

Which resulted in:

* **Training accuracy:96%**
* **Validation accuracy:95%**

Architecture of the network:



1. **Convolutional Neural Networks:**

Convolutional layers in neural networks are learnable feature extractors. They are most suitable for image classifiers and are used in state-of-the-art classifiers. They don’t need explicit feature extractors before using them.

From experience, it was found that the first convolutional layers in a neural network extract the basic features such as corners, edges and so on and as the layers increase the extracted features become more complex. So for our classification task as the images don’t have complex features, a small number of convolutional layers will be enough.

Convolutional neural networks perform a series of convolutions using a number of filters on the input image, and produces an activation map, then we run a series of convolutions on this map and so on.

We tried two architectures inspired from VGG-Net.

First Architecture:

4 convolutional layers and 4 fully connected layers along with dropout

For the convolutional layers:

* Number of filters: 32
* Stride: 1
* Zero padding: same (keep the size of image)
* Filter size: 3x3
* Each convolutional layer is followed by a RELU and max pooling layer

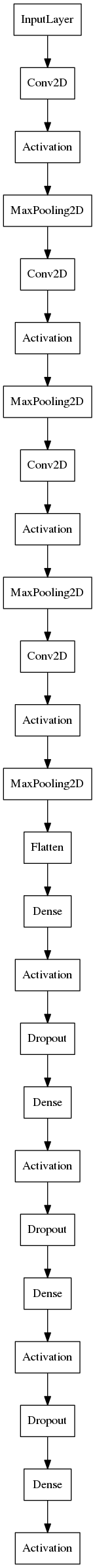
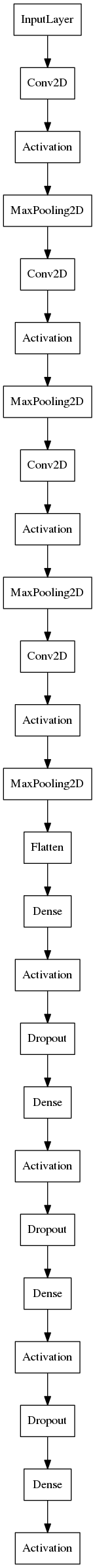
For the fully-connected layers:

* Number of neurons in hidden layers: 128 -64-32-3
* ReLu activations
* RMSprop learning
* learning rate 1e-3
* Dropout 0.1

**Training accuracy:98%**

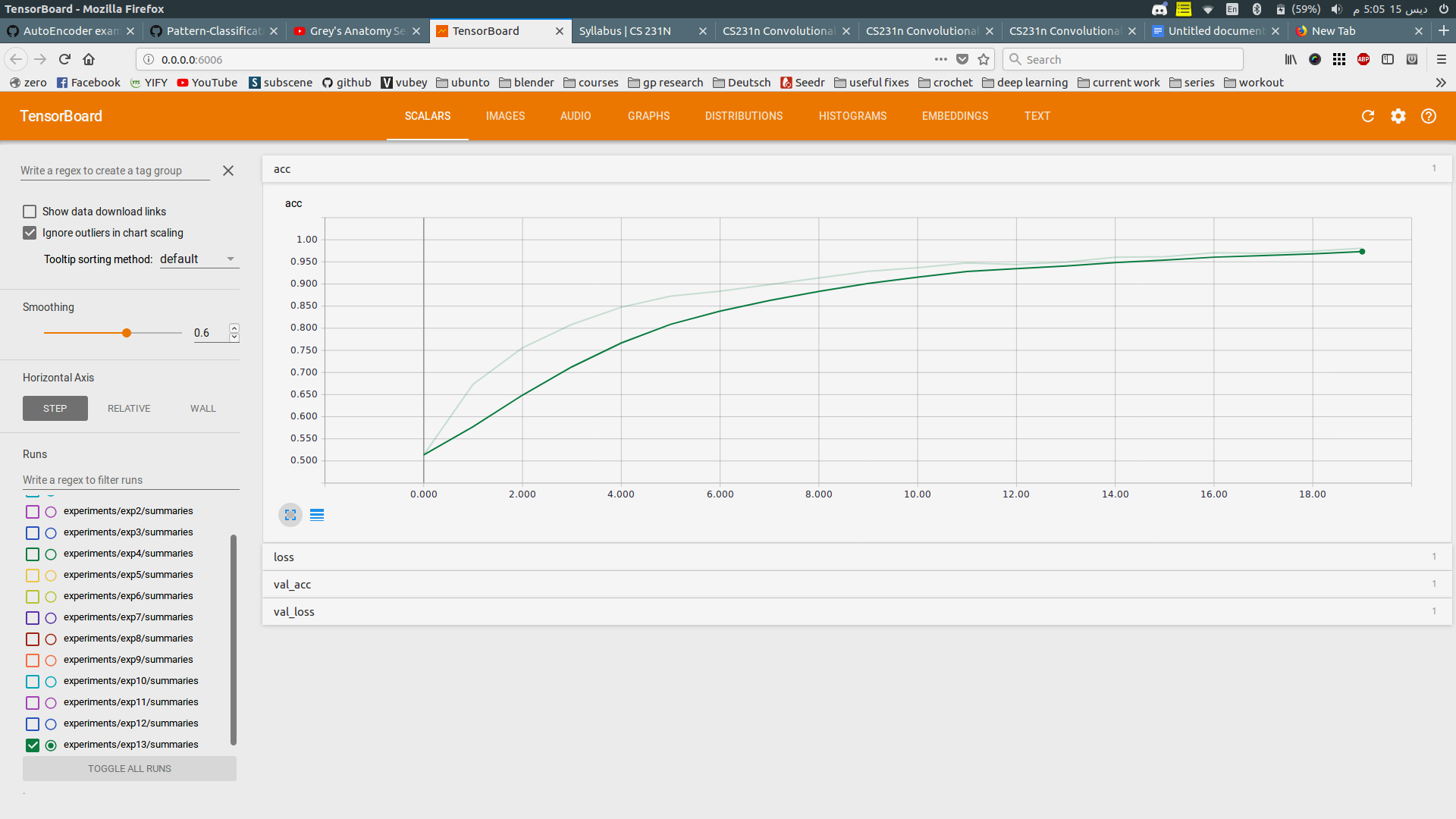
**Validation accuracy: 97%**

Architecture Graph:

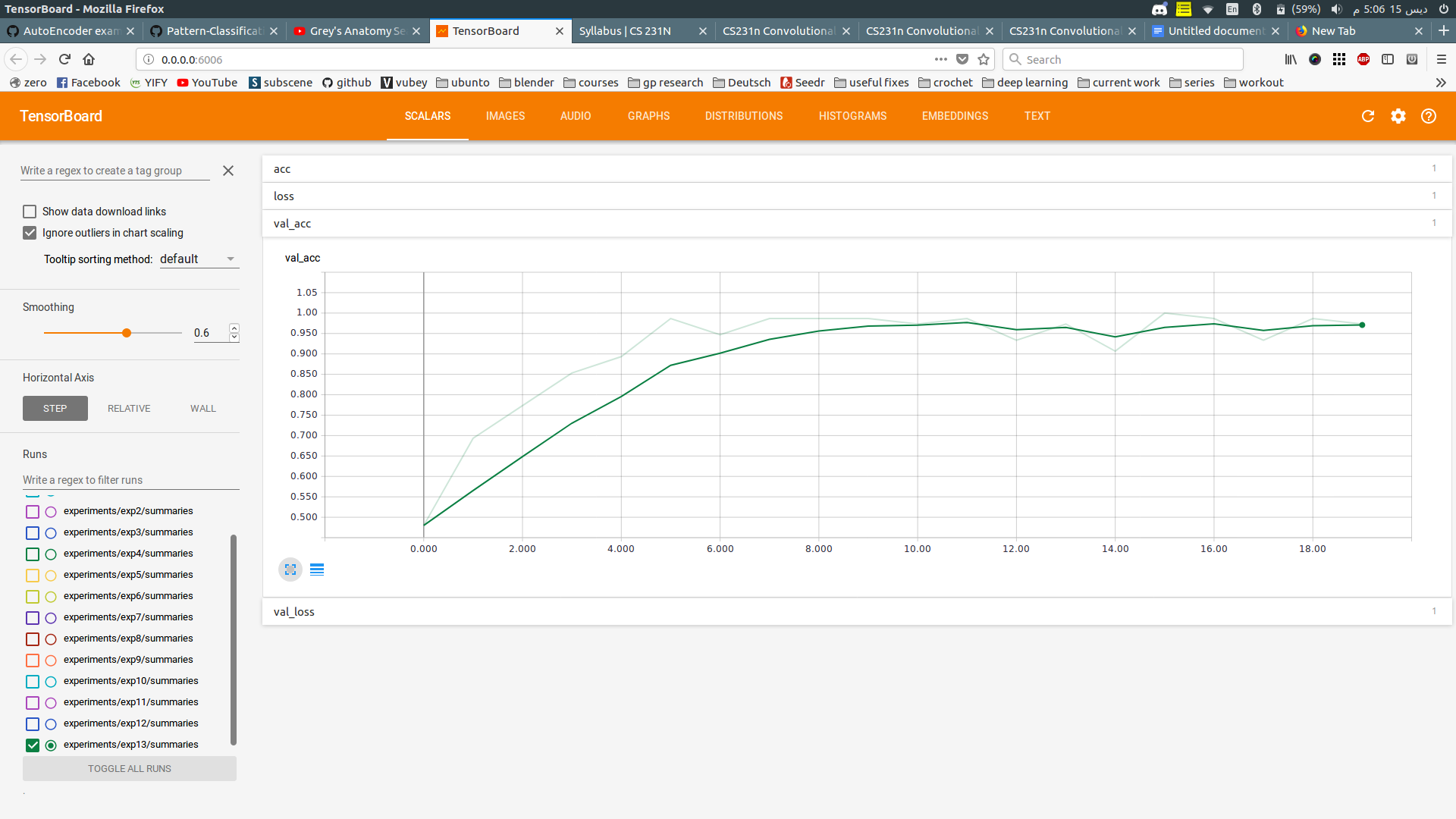


Learning Curves:

Training Accuracy:



Validation Accuracy:



Second Architecture:

5 convolutional layers and 3 fully-connected layers

For convolutional layers:

* Same parameters as the first architecture

For fully connected layers:

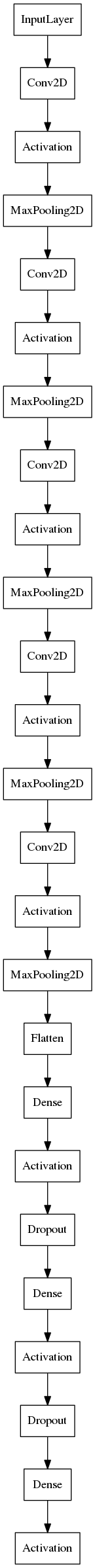
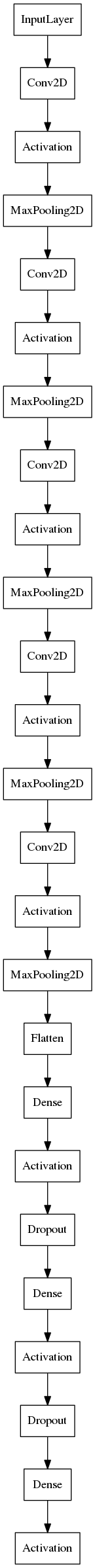
* Number of neurons in each hidden layer:64-32
* Dropout 0.1 after each one
* Relu activation
* Softmax activation for final layer and cross-entropy loss.

Increasing the number of convolutional layer didn’t improve the accuracy by much, which proves that there’s no need to add more parameters to the model.

**Train accuracy:98.7%**

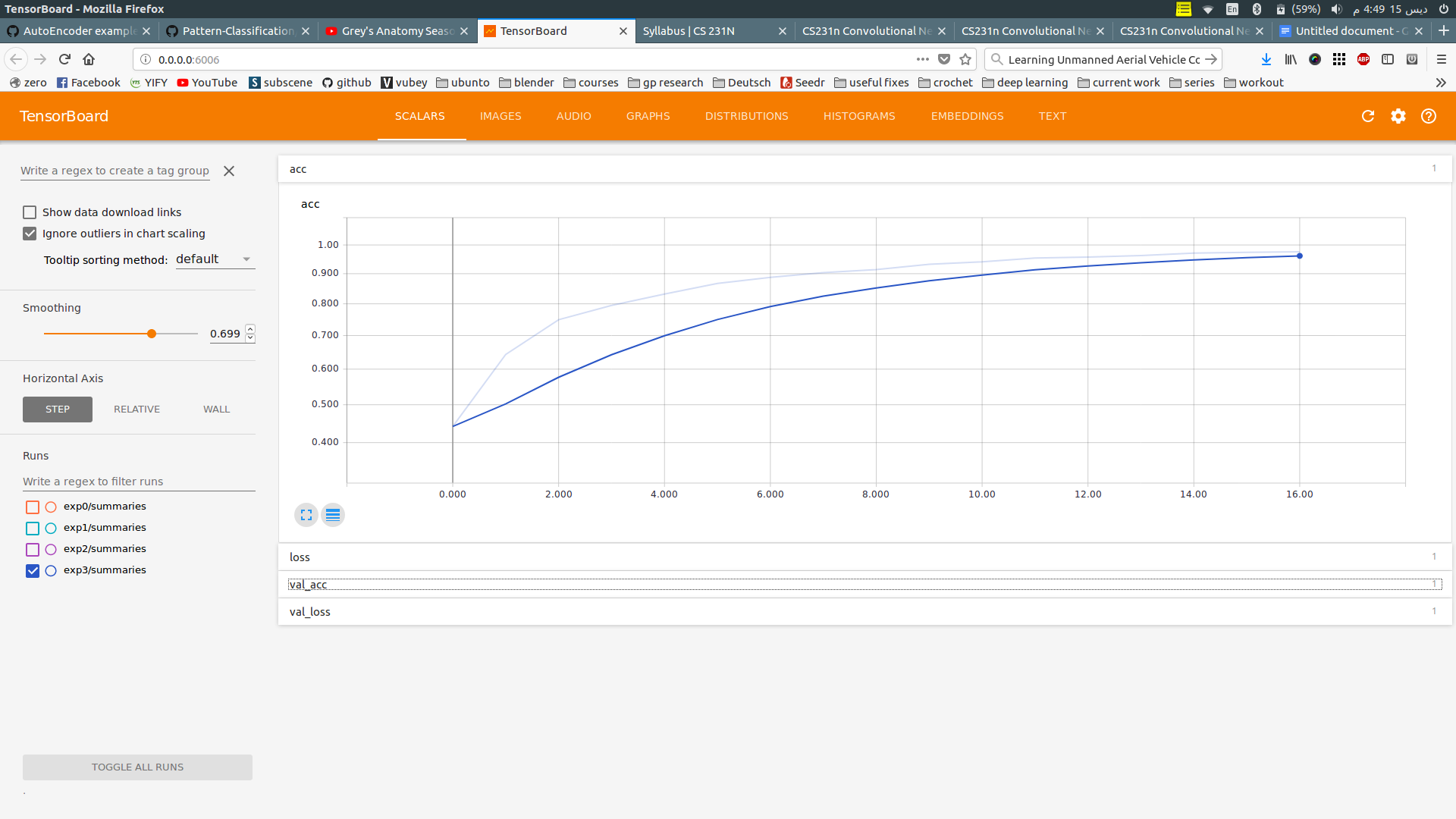
**Validation accurcay:98%**

Architecture Graph:

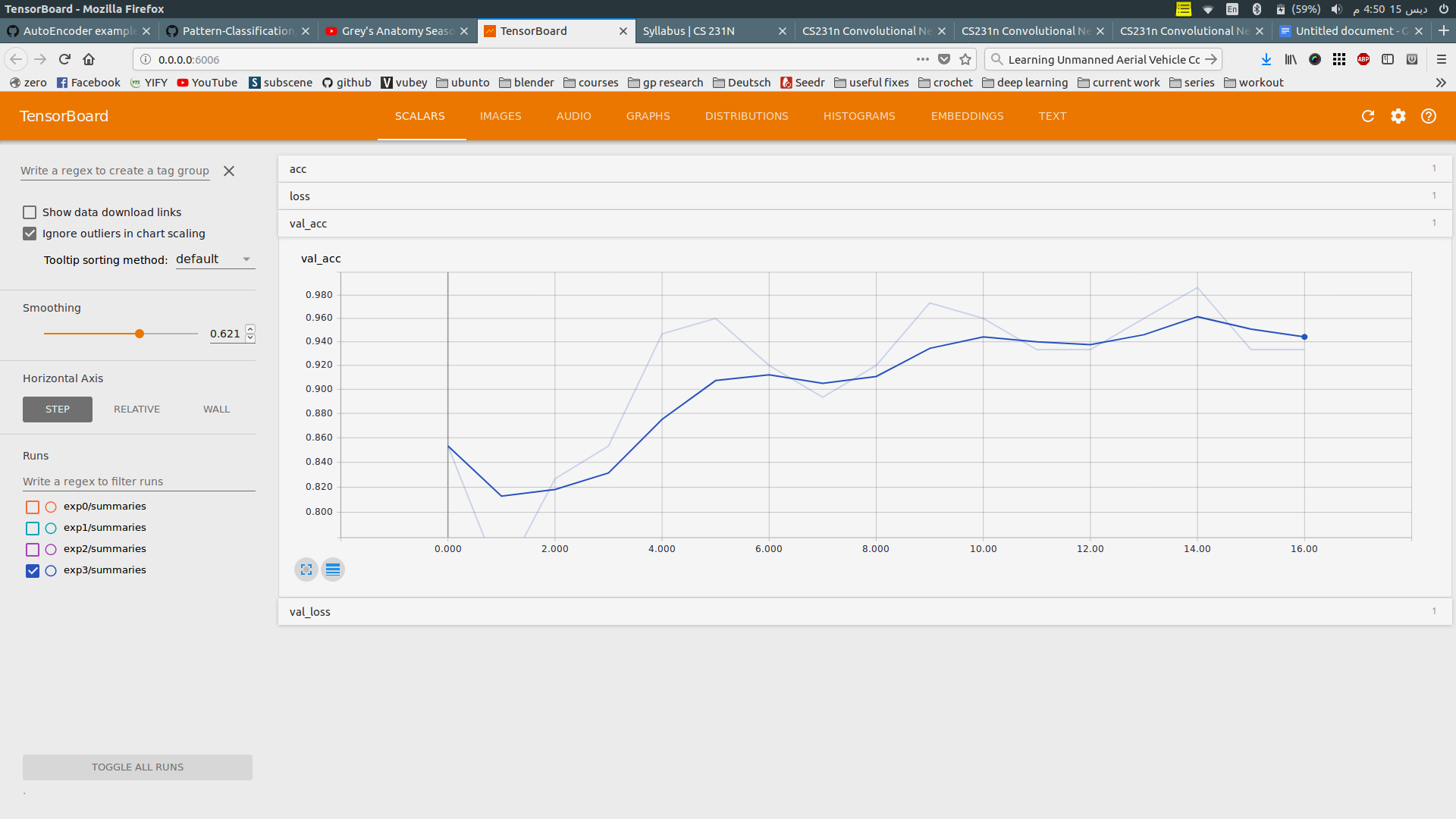


Learning Curves:

Training Accuracy



Validation Accuracy:



Support Vector Machine Model:

* Linear Kernel:

|  |  |
| --- | --- |
| Parameter | Result |
| C=1 | training Accuracy: 1.0  validation Accuracy: 0.9066666666666666 |
| C=5 | training Accuracy: 1.0  validation Accuracy: 0.9066666666666666 |

Using the linear kernel the model always overfitted the data and changing the parameters didn’t affect the model’s accuracy.

* Gaussian Kernel (rbf):

|  |  |
| --- | --- |
| Parameters | Result |
| c=5  gamma=1/n | training Accuracy: 0.7258064516129032  validation Accuracy: 0.9333333333333333 |
| c=10  gamma 1/n | training Accuracy: 0.8001265022137888  validation Accuracy: 0.9333333333333333 |
| c=10  gamma=0.005 | training Accuracy: 1.0  validation Accuracy: 0.8933333333333333 |
| c=15  gamma=1/n | training Accuracy: 0.8235294117647058  validation Accuracy: 0.9066666666666666 |

Rbf kernel improved the results and changing the parameters affects the model’s accuracy

Best Accuracy learning curve of rbf kernel:

