Machine Learning Project: Emotions Recognition

Assigned by:

Andreev Alexandra (309234300)

Gropper Tom (312482037)

**Abstract:**

The report below describes the machine learning project for prediction of emotion recognition from facial expression, taken from an image/video. Model’s training is based on data taken from Cohn-Kanade(CK+) database. The input may come as an image or as video which is broken to separate frames and then is analyzed in the same manner as single images. Emotions are presented by classification of American psychologist Paul Ekman by 7 classes: Happiness, Sadness, Surprise, Fear, Anger, Disgust, Contempt. Methods of predictions compared below: SoftMax, Perceptron, RNN, SVM, AdaBoost – for multiple classes.

**Introduction:**

Recognizing emotions is very challenging task. Here are some challenges we faced:

* Those are modeled emotions, meaning emotion labels are not well validated as they refer to what was requested rather than what was performed.
* Every image has a single image, while it may correspond to multiple labels, i.e. the person might feel multiple emotions.
* Converting single-class classifiers to multi-class classifiers.
* Varying faces/angles/head movements/frame sizes/color scale and intensity – normalization is required.
* Lack of data, of course. The CK+ database includes about 5K images, for (about) 135 different subjects.

**Related Work:**

Emotion Recognition today has a wide variety of potential uses, for example:

* Schools/kindergartens: average mood monitoring
* Security: aggression detection
* Psychiatry: depression detection, mood monitoring
* Health-care: pain expression, infants monitoring
* Human-computer interaction
* Driver safety

Therefore, it is investigated actively nowadays and there are a lot of different publications with different approaches, here are some of them:

* The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression:

<https://ieeexplore.ieee.org/abstract/document/5543262/>

* Cohn-Kanade AU-Coded Expression Database:

<http://www.pitt.edu/~emotion/ck-spread.htm>

* Facial expression databases:

<https://en.wikipedia.org/wiki/Facial_expression_databases>

* Face Landmark Detection using Python and Dlib:

<http://pythongeeks.net/face-landmark-detection-using-python-and-dlib/>

* How to Measure Emotions and Feelings (And the Difference Between Them):

https://imotions.com/blog/difference-feelings-emotions/

**Background**

The report below is a summary for the final project at the Machine Learning Course at Ariel University (lead by Lee-Ad Gottlieb). The purpose of this project was to experience implementing different Machine Learning methods and to get as much intuition as possible. The topic of emotion recognition was chosen for three reasons:

1. Desired experience in data composed of images/videos, which are widely used in both Science Investigations & Experiments and different fields in the Industry.
2. Interesting topic which involves more than just numbers, but other sciences as well. For example, psychology.
3. a very important topic at present, which might improve and have numerous uses.

Applying machine learning methods supplies additional instruments for images/videos data recognition (though usually it requires much bigger data sets).

On the other hand, the fact that the amount of data used is not too large, allowed us to debug much deeper, to understand the Machine Learning mechanisms, and to get better intuition about tuning parameters at different methods. Also, it gave us a free hand in running massive number of different configurations as each run required a small amount of computing resources and performance time.

**Project Description:**

The best result (closes to the actual result) is predicted by method: RNN

1. **Data**

The data set we based our Learning on is the CK+ (Cohn-Kanade AU-Coded Expression) database. It includes more than 500 sequences from more than 130 subjects, which in summary is about 5K images.  Each sequence begins with a neutral expression and proceeds to a peak expression. The last expression in a sequence is given an emotional label. Sequences are not equal in length. Images are colored or gray scaled. The sizes of images may differ, as well as pixel intensity. Because of this, each image required normalization before they could be used as training/testing.

1. **Normalization**

Two methods were used:

* 1. **SoftMax, Perceptron, AdaBoost**
     1. Center the data on the origin.
     2. Scale it down by the same amount in both dimensions (choose the larger between two dimensions). Need to scale both xs & ys in order to preserve the distance.
     3. Move all points to be around (0.5, 0.5).
  2. **RNN, SVM**
     1. Find the center of the cluster of points representing the face.
     2. Calculate the vector between every point in the cluster and its center.
     3. Normalize the distances of the vectors by dividing them by the longest distance.
     4. Normalize the vectors angles by dividing them by 360 (angles are between 0 and 360).
     5. Replace every point with its corresponding vector.
     6. Zero pad every sequence until it has as much frames as the longest sequence.

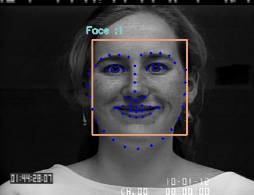
Images sequence for example:

The whole data was split into training set (80%) and test set (20%), based on random indexing.

1. **Main flow explanation**

* Given a video, split it to frames. Further steps are for image.
* Detect the face by setting 68 landmarks. Normalize and flatten it – getting vector of 136 float values.



* Train the model using the training set.
* Test the model using the test set.
* Classify results by 7 emotions: in case of probabilistic result take the emotion with maximal probability.

1. **Results: comparing all methods**

* **Files/Sequences statistics:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Anger** | **Contempt** | **Disgust** | **Fear** | **Happy** | **Sadness** | **Surprise** | **Summary** | **Train 80%** | **Test 20%** |
| Sequences | 45 | 18 | 59 | 25 | 69 | 28 | 83 | 345 | 276 | 69 |
| Image files | 1,022 | 233 | 868 | 546 | 1,331 | 547 | 1,329 | 6,109 | 4,887 | 1,222 |
| Percent from data set | 17 | 4 | 14 | 9 | 22 | 9 | 22 | 100 |  |  |
| 20 % from Image files | 204 | 47 | 174 | 109 | 266 | 109 | 266 | 1,222 | 977 | 244 |
| 80% from Image files | 818 | 186 | 694 | 437 | 1,065 | 438 | 1,063 | 4,887 | 3,910 | 977 |

* **Test/Train common stats:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **SoftMax** | **Perceptron** | **RNN** | **SVM** | **AdaBoost** |
| Test set size | 1,175 | 1,175 | 70 | 75 | 1,175 |
| Train time | 3.5 mins | 2-47 mins | 25 mins | 3 hours | 40secs – 15 mins |

**Legend:**

|  |  |  |
| --- | --- | --- |
|  | **Classified as C** | **Classified as other** |
| **Labeled as C** | True Positive (TP) | False Negative (FN) |
| **Labeled as other** | False Positive (FP) | True Negative (TN) |

* **Anger (16.7% of data):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **SoftMax** | **Perceptron** | **RNN** | **SVM** | **AdaBoost** |
| TP | 160 | 166 | 8 | 6 | 208 |
| FP | 194 | 72 | 2 | 2 | 874 |
| FN | 60 | 25 | 1 | 3 | 208 |
| TN | 787 | 853 | 59 | 64 | 208 |
| Accuracy | 0.79 | 0.91 | 0.96 | 0.93 | 0.28 |
| Recall | 0.73 | 0.87 | 0.89 | 0.67 | 0.50 |
| Precision | n/a | 0.70 | 0.80 | 0.75 | 0.19 |
| **F-measure** | n/a | 0.77 | 0.84 | 0.71 | 0.28 |

* **Contempt (3.8% of data):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **SoftMax** | **Perceptron** | **RNN** | **SVM** | **AdaBoost** |
| TP | 0 | 38 | 2 | 0 | 45 |
| FP | 0 | 11 | 0 | 0 | 617 |
| FN | 47 | 14 | 2 | 1 | 47 |
| TN | 1154 | 1053 | 66 | 74 | 45 |
| Accuracy | 0.96 | 0.98 | 0.97 | 0.99 | 0.12 |
| Recall | 0.00 | 0.73 | 0.50 | 0.00 | 0.49 |
| Precision | n/a | 0.78 | 1.00 | n/a | 0.07 |
| **F-measure** | n/a | 0.75 | 0.67 | n/a | 0.12 |

* **Disgust (14.2 % of data):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **SoftMax** | **Perceptron** | **RNN** | **SVM** | **AdaBoost** |
| TP | 75 | 113 | 19 | 10 | 176 |
| FP | 55 | 40 | 1 | 1 | 706 |
| FN | 96 | 46 | 3 | 63 | 177 |
| TN | 975 | 917 | 47 | 1 | 176 |
| Accuracy | 0.87 | 0.92 | 0.94 | 0.15 | 0.29 |
| Recall | 0.44 | 0.71 | 0.86 | 0.14 | 0.50 |
| Precision | 0.58 | 0.74 | 0.95 | 0.91 | 0.20 |
| **F-measure** | 0.50 | 0.72 | 0.90 | 0.24 | 0.29 |

* **Fear (8.94 % of data):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **SoftMax** | **Perceptron** | **RNN** | **SVM** | **AdaBoost** |
| TP | 23 | 69 | 4 | 3 | 114 |
| FP | 43 | 12 | 2 | 0 | 655 |
| FN | 81 | 24 | 1 | 2 | 114 |
| TN | 1054 | 1011 | 63 | 70 | 114 |
| Accuracy | 0.90 | 0.97 | 0.96 | 0.97 | 0.23 |
| Recall | 0.22 | 0.74 | 0.80 | 0.60 | 0.50 |
| Precision | 0.35 | 0.85 | 0.67 | 1.00 | 0.15 |
| **F-measure** | 0.27 | 0.79 | 0.73 | 0.75 | 0.23 |

* **Happy (21.79% of data):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **SoftMax** | **Perceptron** | **RNN** | **SVM** | **AdaBoost** |
| TP | 212 | 244 | 10 | 18 | 272 |
| FP | 108 | 88 | 0 | 2 | 639 |
| FN | 56 | 19 | 1 | 0 | 236 |
| TN | 825 | 765 | 59 | 55 | 272 |
| Accuracy | 0.86 | 0.90 | 0.99 | 0.97 | 0.38 |
| Recall | 0.79 | 0.93 | 1.00 | 1.00 | 0.54 |
| Precision | 0.66 | 0.73 | 0.90 | 0.90 | 0.30 |
| **F-measure** | 0.72 | 0.82 | 0.95 | 0.95 | 0.38 |

* **Sadness (8.95% of data):**

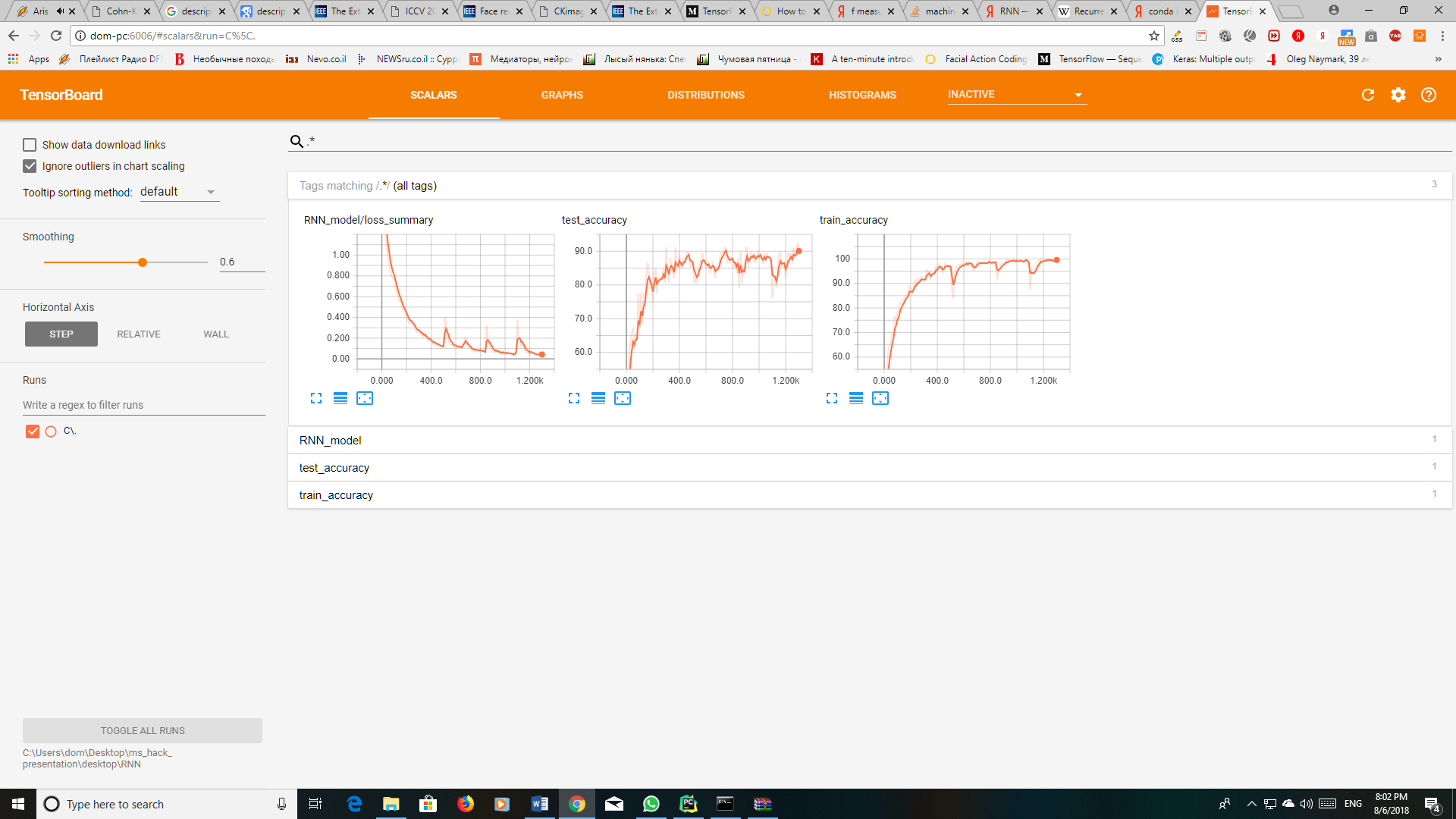
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **SoftMax** | **Perceptron** | **RNN** | **SVM** | **AdaBoost** |
| TP | 0 | 63 | 4 | 4 | 114 |
| FP | 1 | 7 | 3 | 2 | 685 |
| FN | 116 | 38 | 0 | 1 | 115 |
| TN | 1084 | 1008 | 63 | 68 | 114 |
| Accuracy | 0.90 | 0.96 | 0.96 | 0.96 | 0.22 |
| Recall | 0.00 | 0.62 | 1.00 | 0.80 | 0.50 |
| Precision | 0.00 | 0.90 | 0.57 | 0.67 | 0.14 |
| **F-measure** | n/a | 0.74 | 0.73 | 0.73 | 0.22 |

* **Surprise (21.75% of data):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **SoftMax** | **Perceptron** | **RNN** | **SVM** | **AdaBoost** |
| TP | 218 | 184 | 15 | 26 | 270 |
| FP | 112 | 9 | 0 | 1 | 652 |
| FN | 57 | 73 | 0 | 0 | 220 |
| TN | 814 | 850 | 55 | 48 | 270 |
| Accuracy | 0.86 | 0.93 | 1.00 | 0.99 | 0.38 |
| Recall | 0.79 | 0.72 | 1.00 | 1.00 | 0.55 |
| Precision | 0.66 | 0.95 | 1.00 | 0.96 | 0.29 |
| **F-measure** | 0.72 | 0.82 | 1.00 | 0.98 | 0.38 |

Loss and Accuracy progress over iterations for the best result (RNN):

Loss progress (final = 0.04: Accuracy progress (final = 90):



1. **Experiments:**

Multiple configurations had been processed and compared each vs other for the tuning

purposes. Tuning parameters validated:

* Dropout percent
* Loss type
* Iterations number
* Learning Rate (at Gradient Descent) options
* Train percent
* Neurons numbers
* Activation function

1. **Summary of models:**

* **SoftMax Regression**

SoftMax Regression (or multinomial logistic regression) is a generalization of logistic regression to the case where we want to handle multiple classes. SoftMax regression gives us a vector y where each coordinate is a probability of emotion i. Then we take the maximal probability and return it as the result.

Tuning parameters:

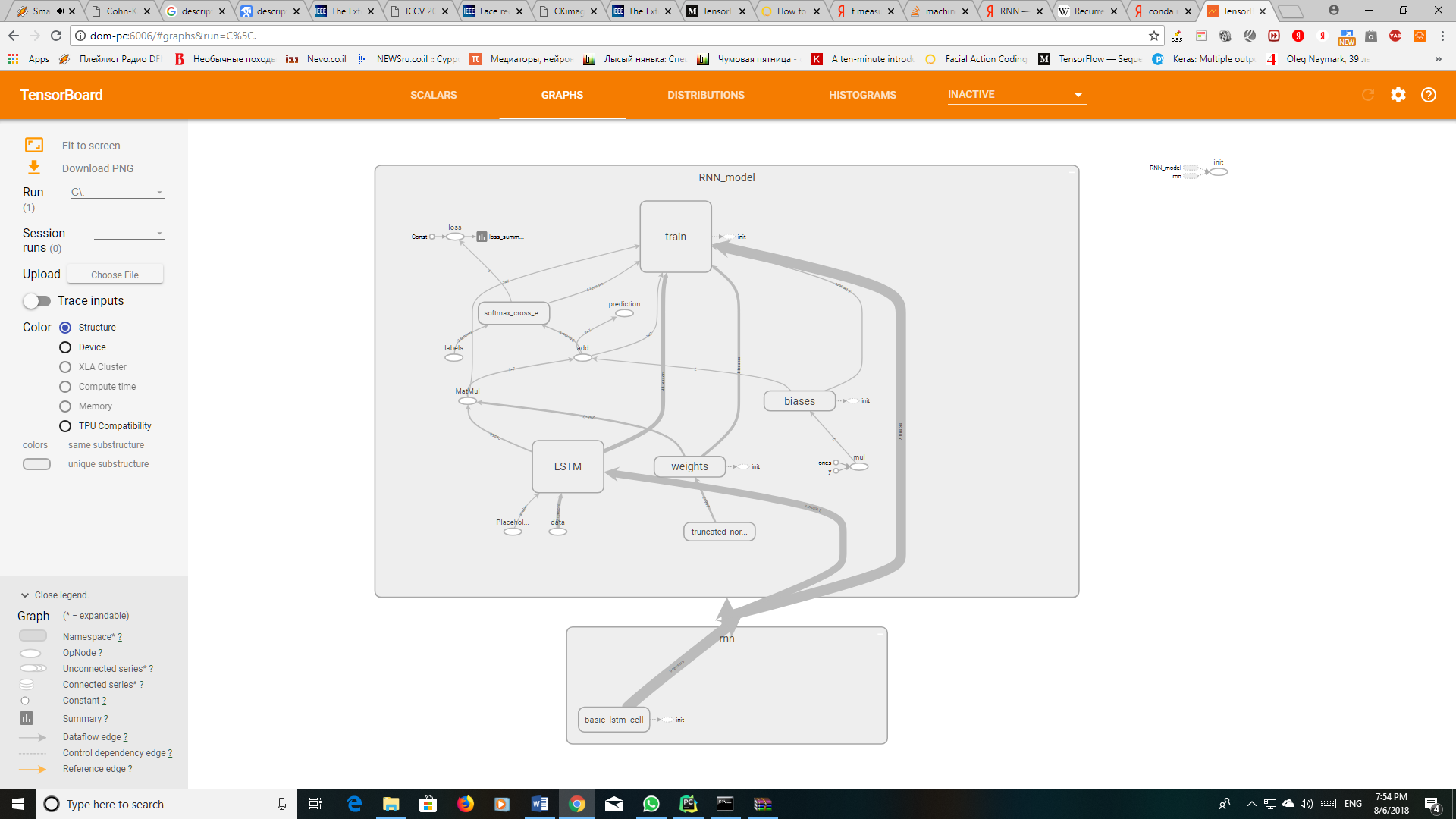
* + Using Gradient Descent Optimizer with Alpha 0.001
  + 500K iterations
  + Loss by: loss = -tf.reduce\_mean(y\_ \* tf.log(y)) – starting (after 1000 iterations) from 0.266 ending at 0.17(actually is not improved too much with iterations).
  + Resulted in tf.accuracy = 0.57, evaluation time: 47 mins.
* **Perceptron Architecture**

Perceptron is type of simples Neural Network, which used the same functions as SoftMax Regression. Two models had been analyzed: with Single Layer and the Perceptron with two layers. Tuning parameters:

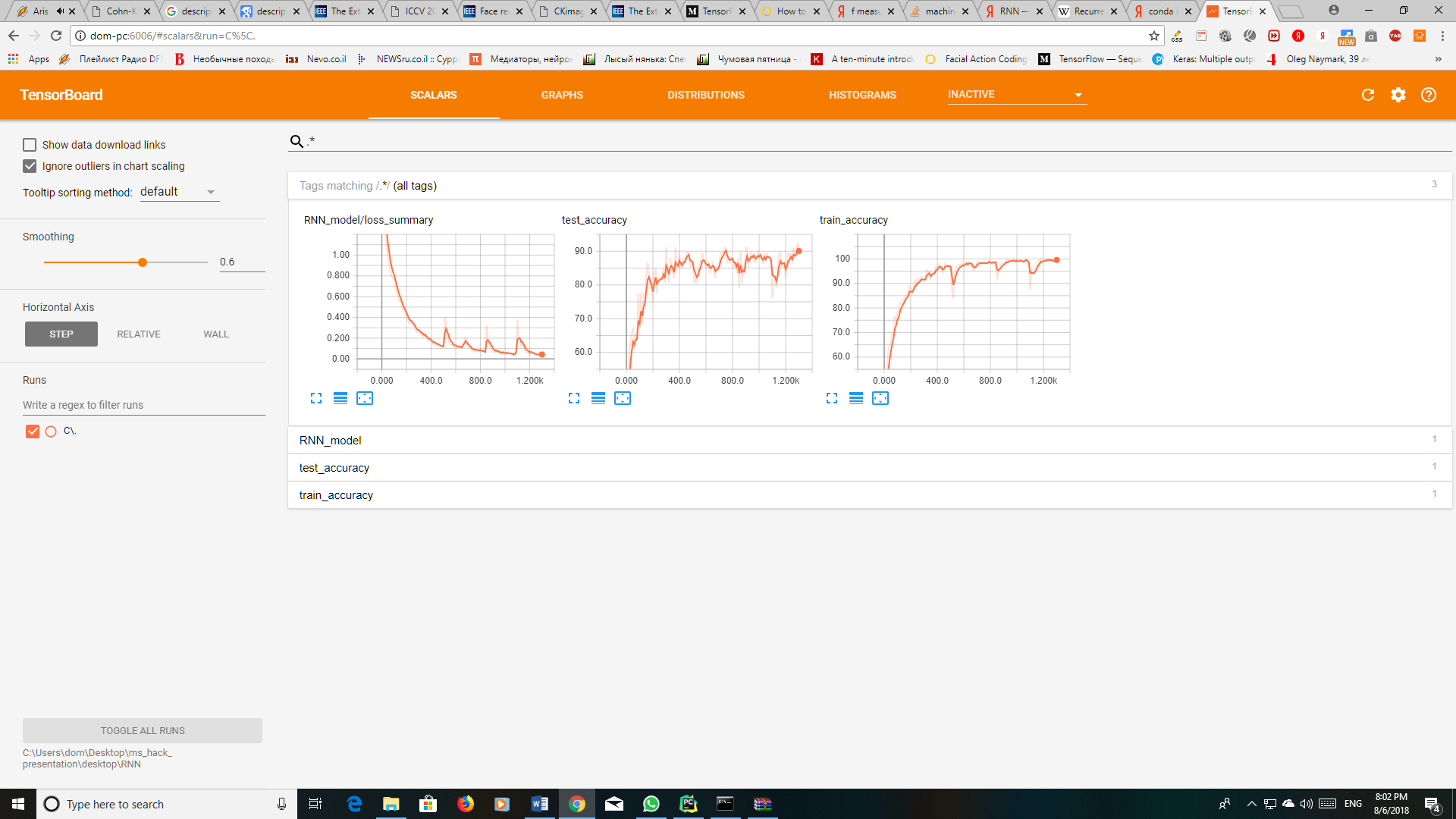
* + Single layer:
    - SoftMax for neural network
    - Cross entropy by: tf.reduce\_mean(-tf.reduce\_sum(y\_ \* tf.log(y), reduction\_indices=[1]))
    - Training step by Gradient Descent Optimizer with Alpha 0.5
    - 370,000 iterations, each iteration by batch of 100 input vectors.
    - Resulted in (tf) accuracy 0.786, evaluation time: 2:43 mins.
  + Two Layers:
    - Hidden Layers: 200, 136
    - RELU after each one
    - Third layer is the SoftMax applying same tuning parameters as a Single Layer
    - Iterations: 300K, batches of 100 vectors
    - Resulted in accuracy 0.66, evaluation time: 7 mins.
  + Best run results:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Emotion** | **Anger** | **Contempt** | **Disgust** | **Fear** | **Happy** | **Sadness** | **Surprise** |
| Data % | 16.7 | 3.8 | 14.2 | 8.9 | 21.8 | 8.9 | 21.8 |
| TP | 166 | 38 | 113 | 69 | 244 | 63 | 184 |
| FP | 72 | 11 | 40 | 12 | 88 | 7 | 9 |
| FN | 25 | 14 | 46 | 24 | 19 | 38 | 73 |
| TN | 853 | 1053 | 917 | 1011 | 765 | 1008 | 850 |
| Accuracy | 0.91 | 0.98 | 0.92 | 0.97 | 0.90 | 0.96 | 0.93 |
| Recall | 0.87 | 0.73 | 0.71 | 0.74 | 0.93 | 0.62 | 0.72 |
| Precision | 0.70 | 0.78 | 0.74 | 0.85 | 0.73 | 0.90 | 0.95 |
| **F-measure** | 0.77 | 0.75 | 0.72 | 0.79 | 0.82 | 0.74 | 0.82 |

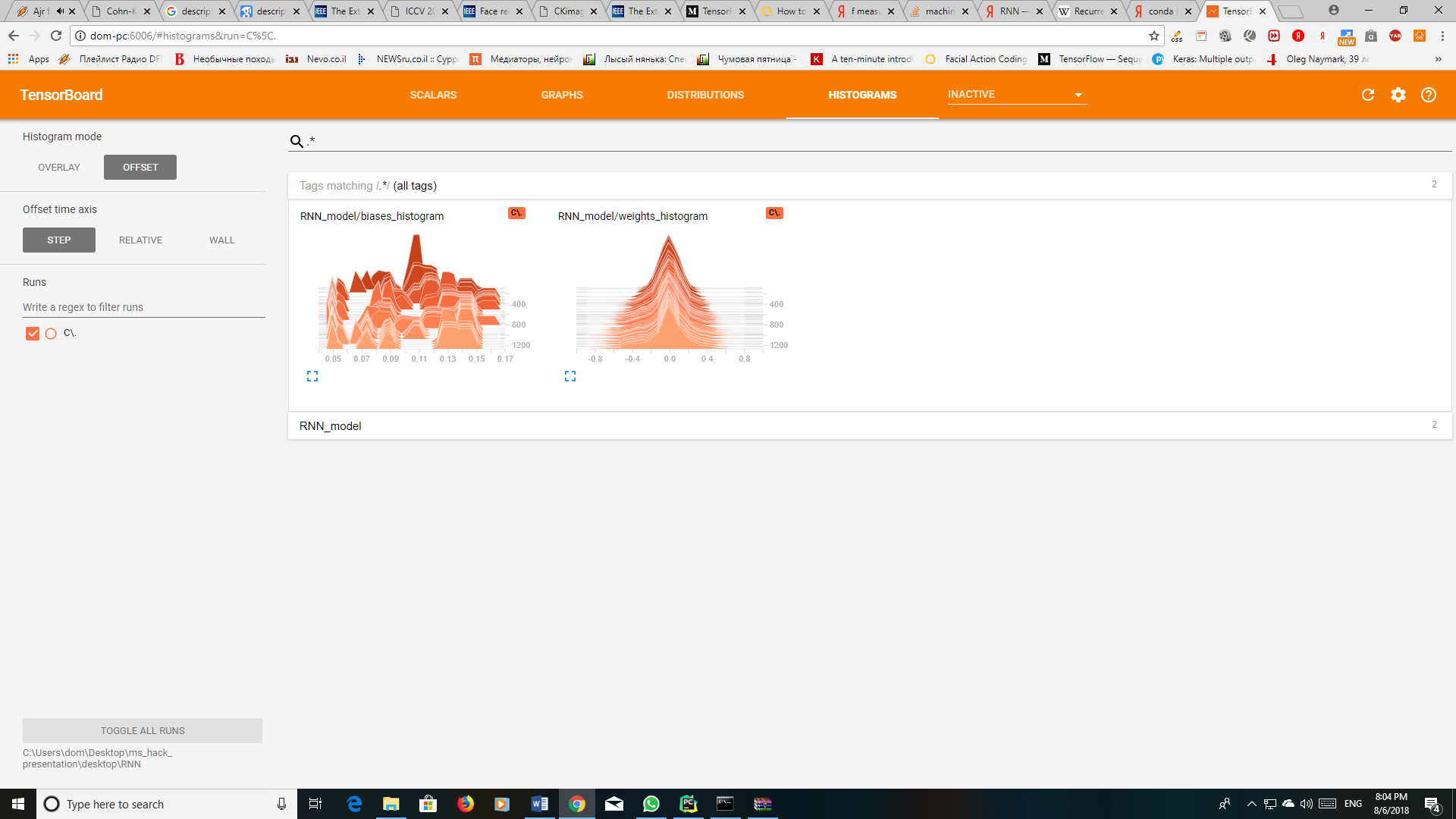
* **RNN Architecture**A recurrent neural network (RNN) is a class of [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) where connections between nodes form a [directed graph](https://en.wikipedia.org/wiki/Directed_graph) along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. Unlike [feedforward neural networks](https://en.wikipedia.org/wiki/Feedforward_neural_networks), RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected [handwriting recognition](https://en.wikipedia.org/wiki/Handwriting_recognition) or [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition).  
  Our model takes as input a zero-padded sequence and inserts it to an RNN. The output of the RNN goes through a sigmoid activation layer after which the model only picks the last relevant frames in the output (since the last frame for most of the sequences will be all zeroes). The matrix that was gathers is the values that are used in calculating the loss.
* **Details of parameters:**
* LSTM activation layer: sigmoid
* Hidden activation layer: sigmoid
* Dropout percent: 0.5
* Iterations number: 1.2k
* Learning rate: 0.001
* Neurons number: 256
* **Scheme (TensorBoard):**



**Loss and accuracy:**



**Distribution of weights:**



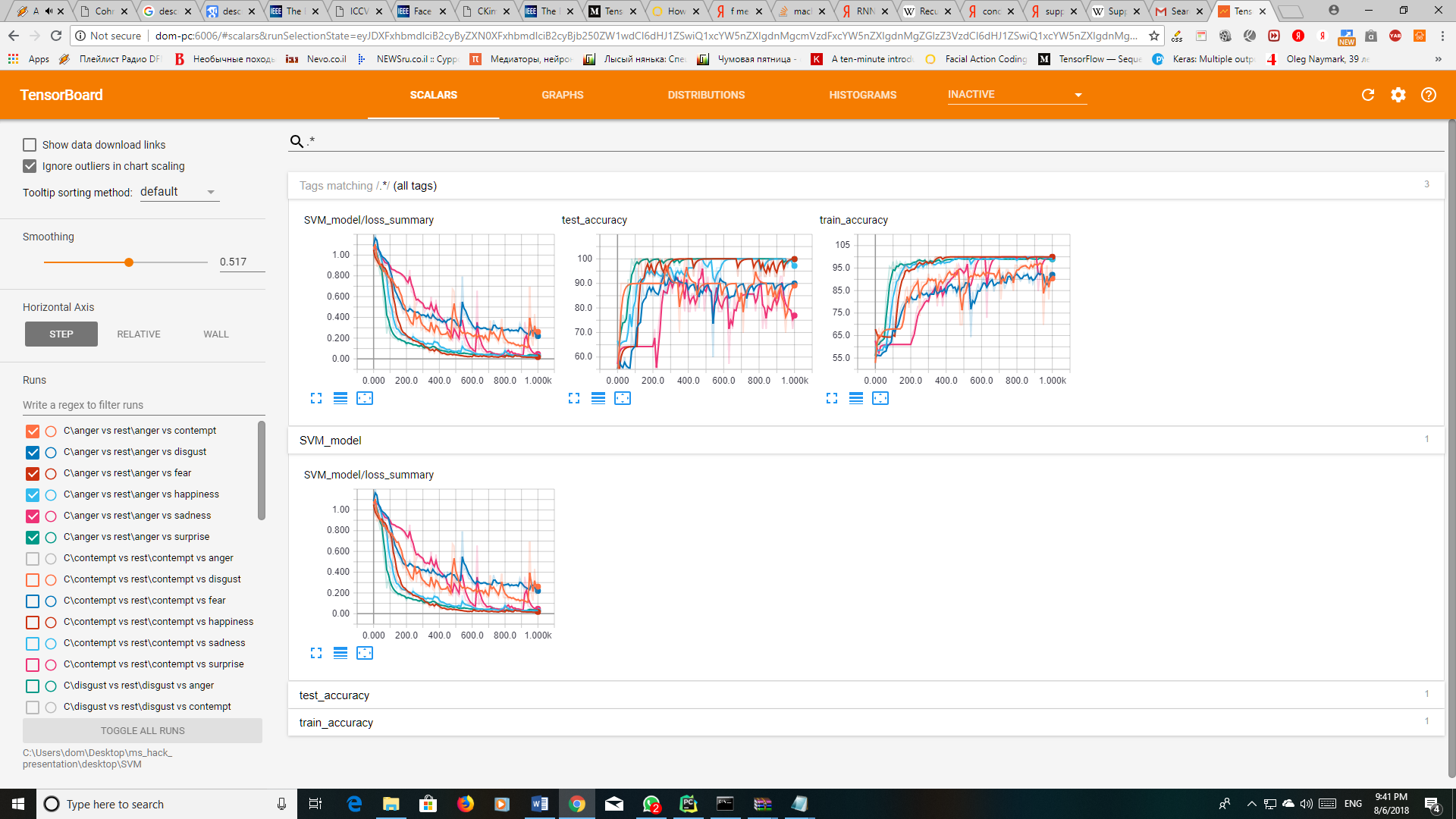
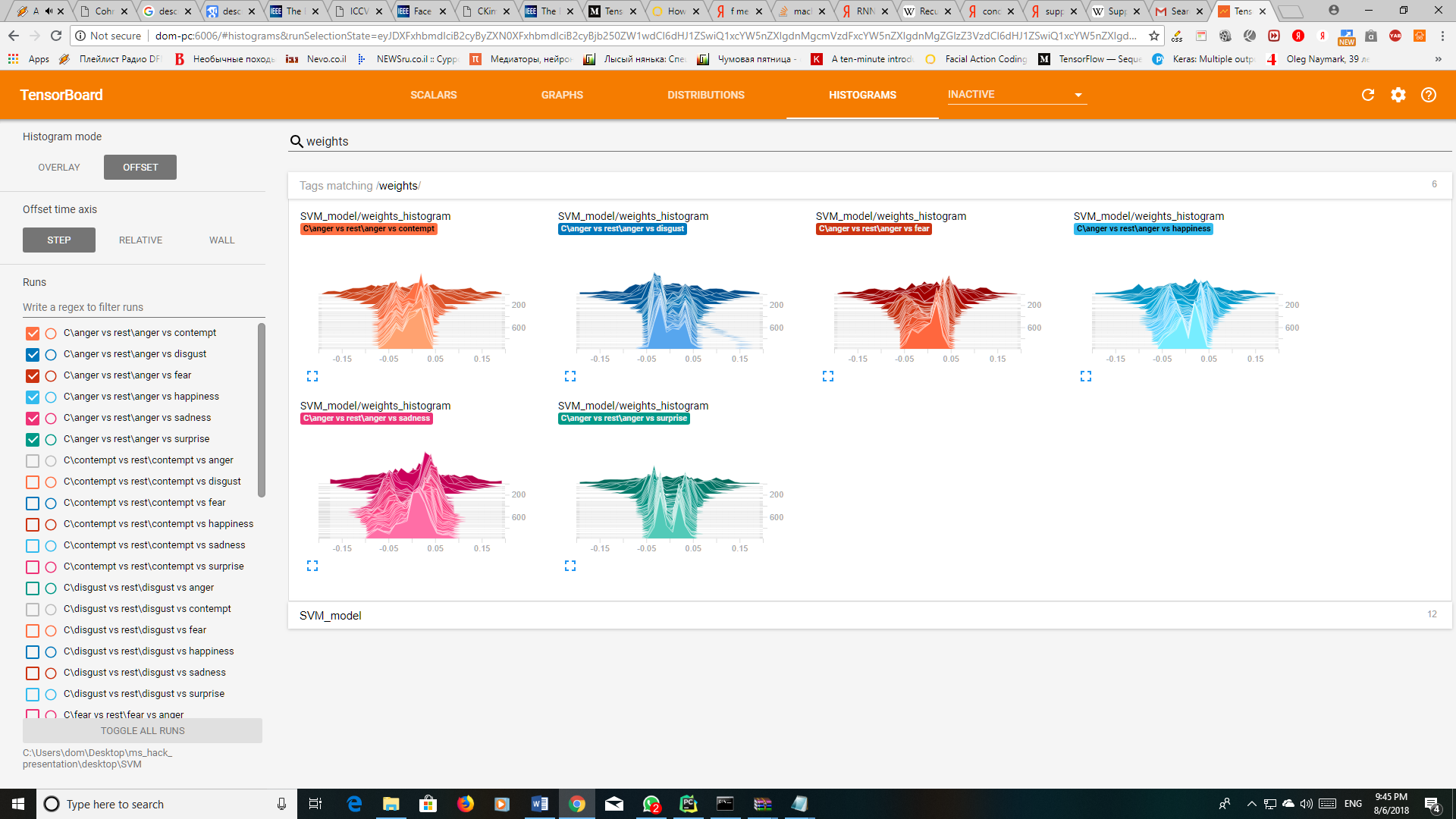
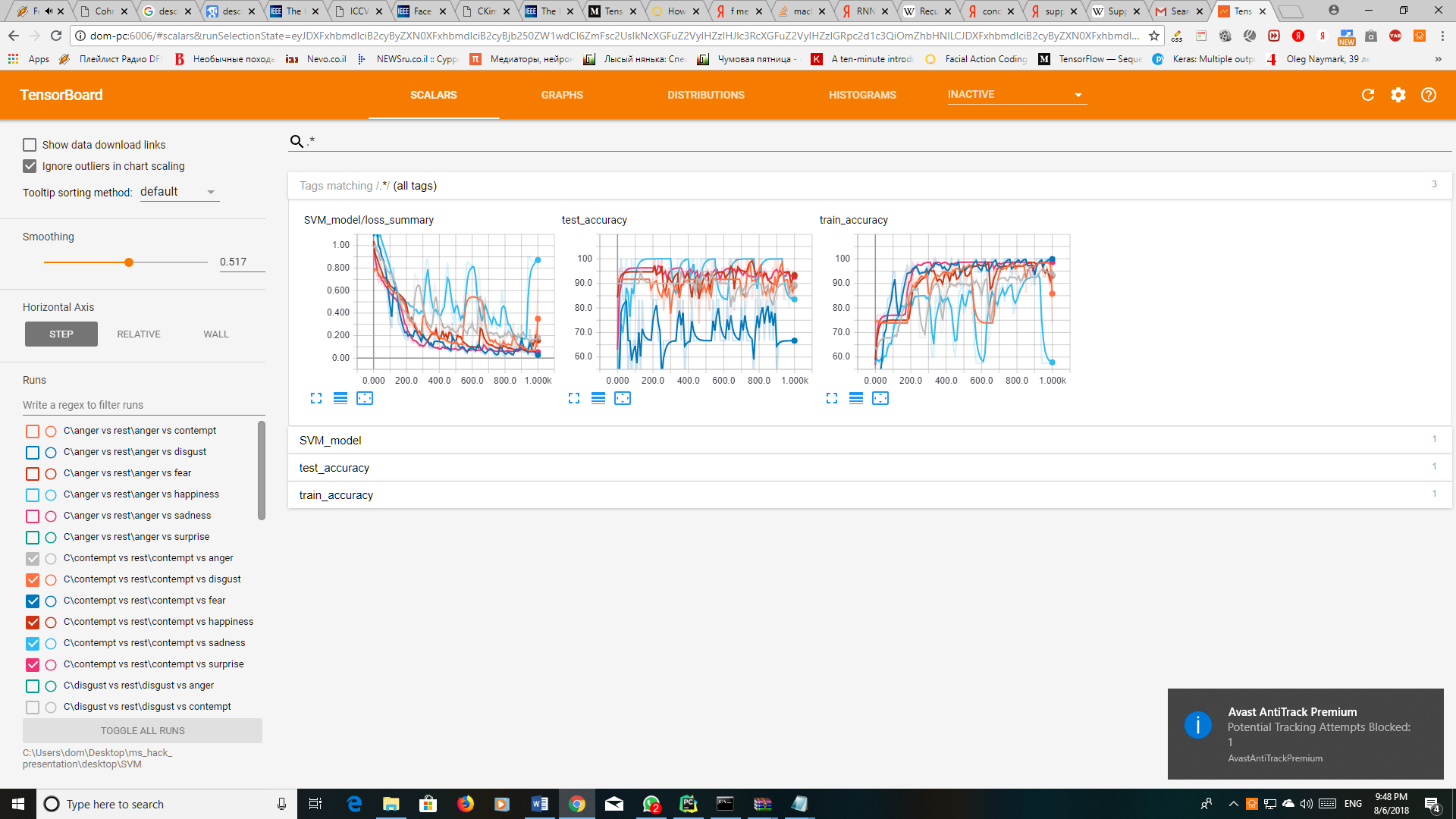
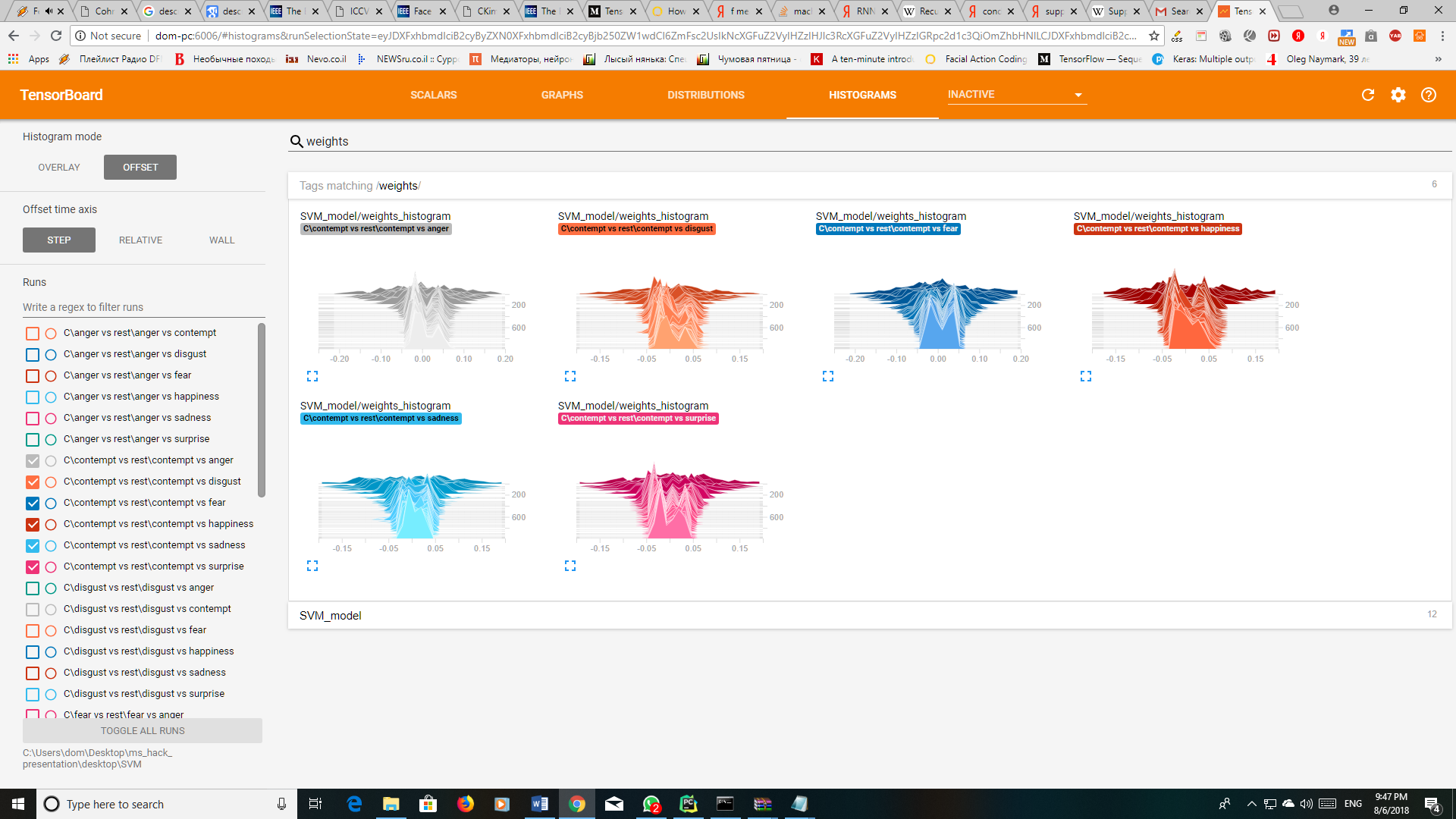
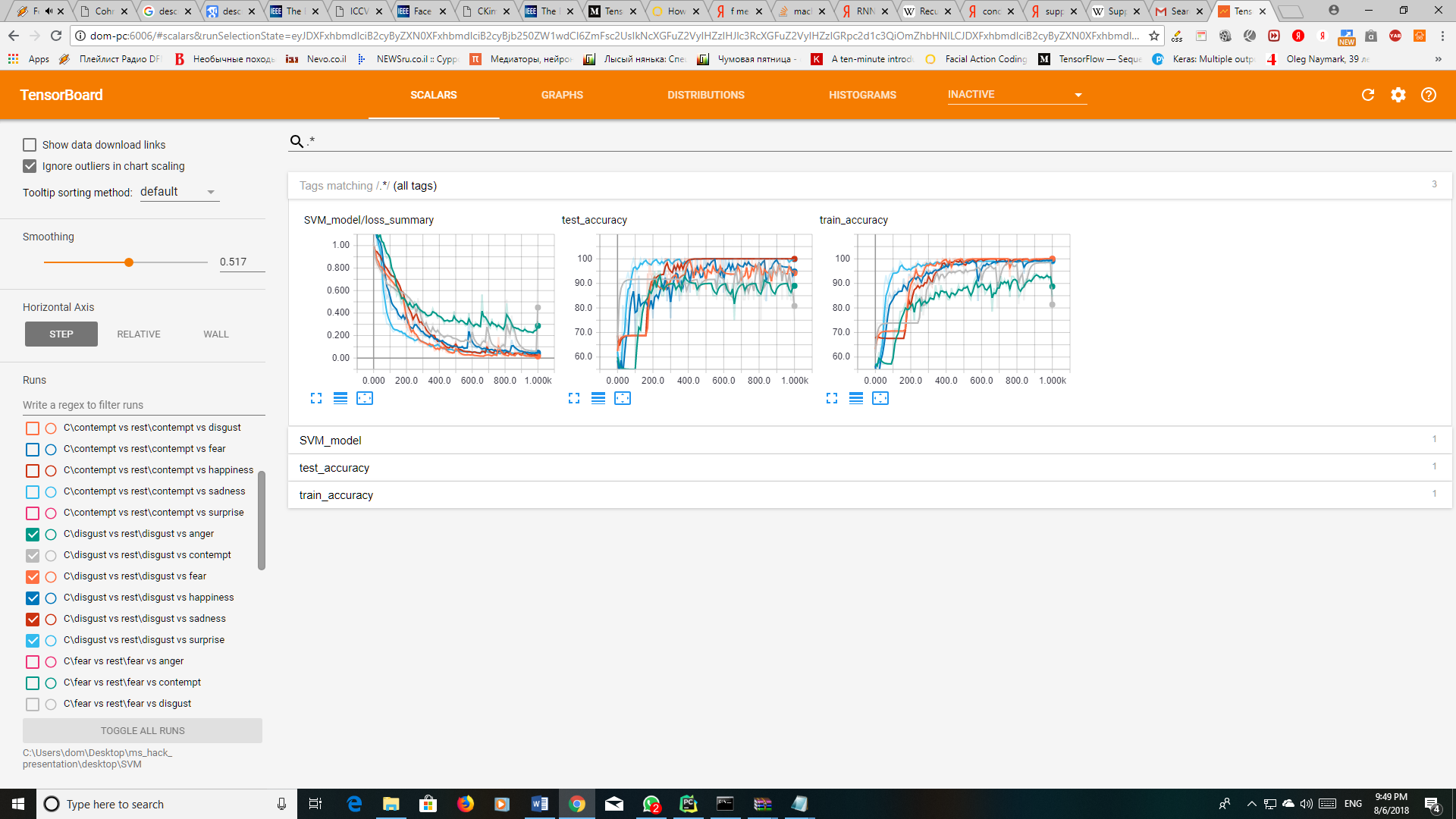
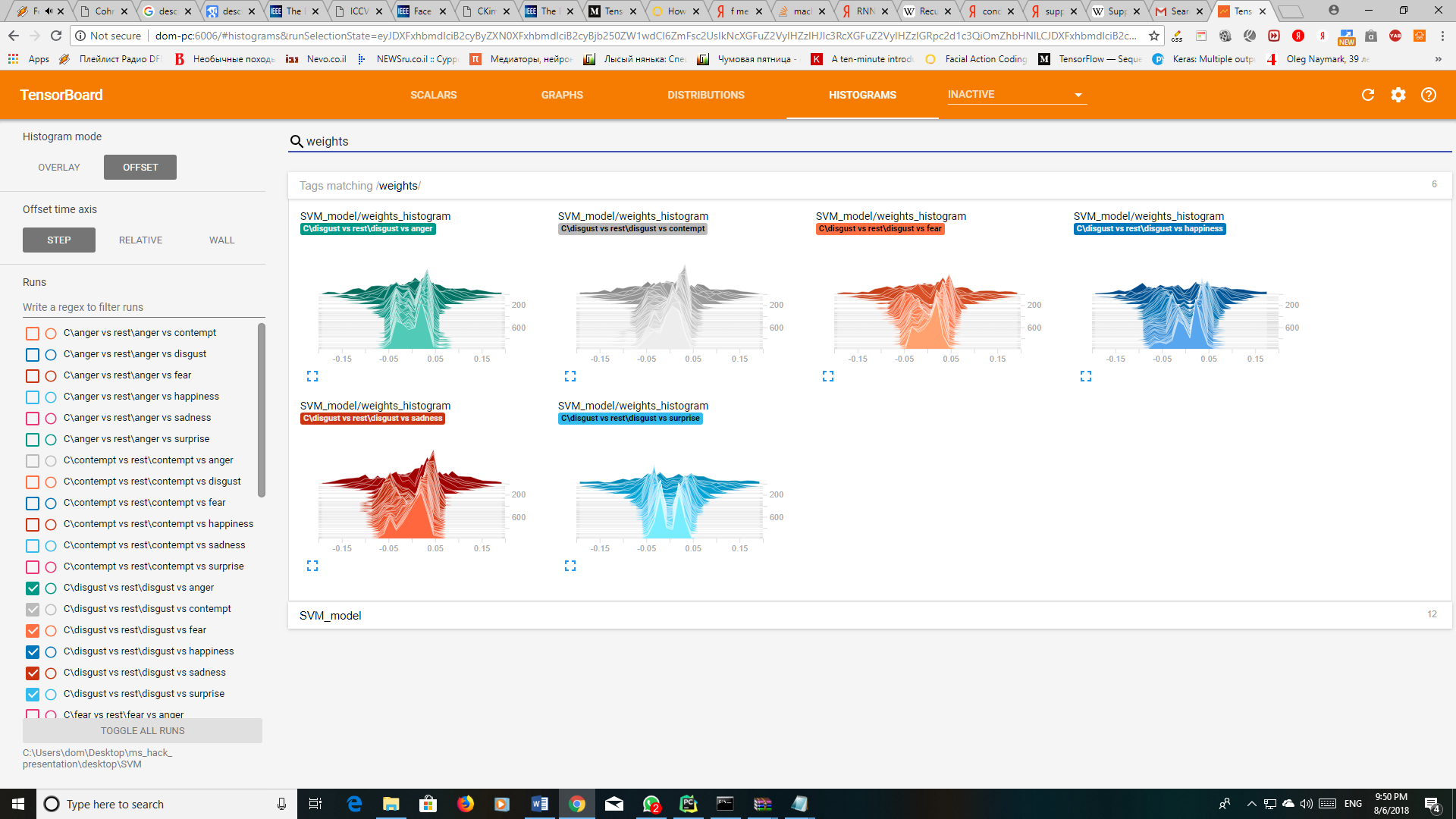
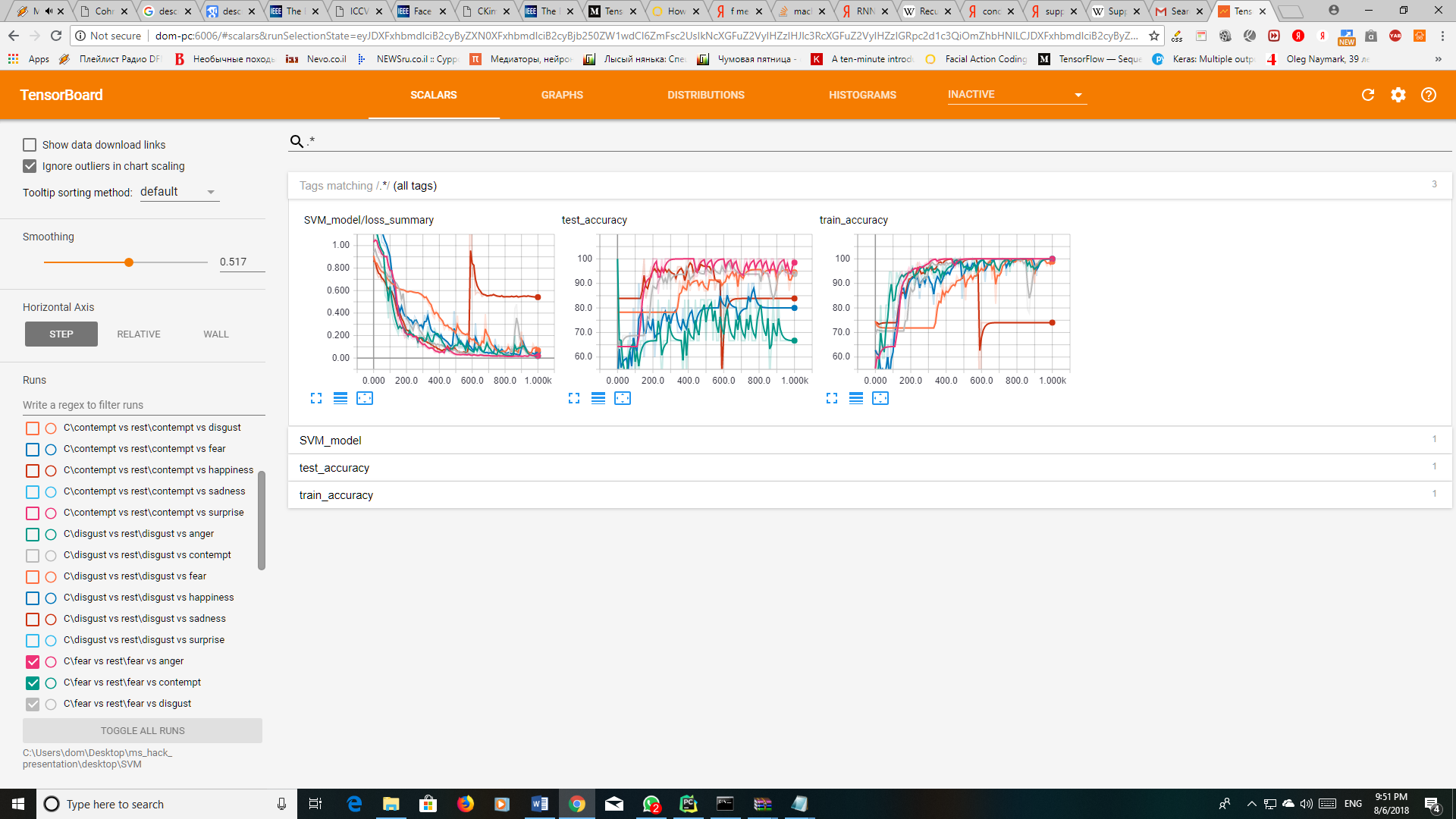
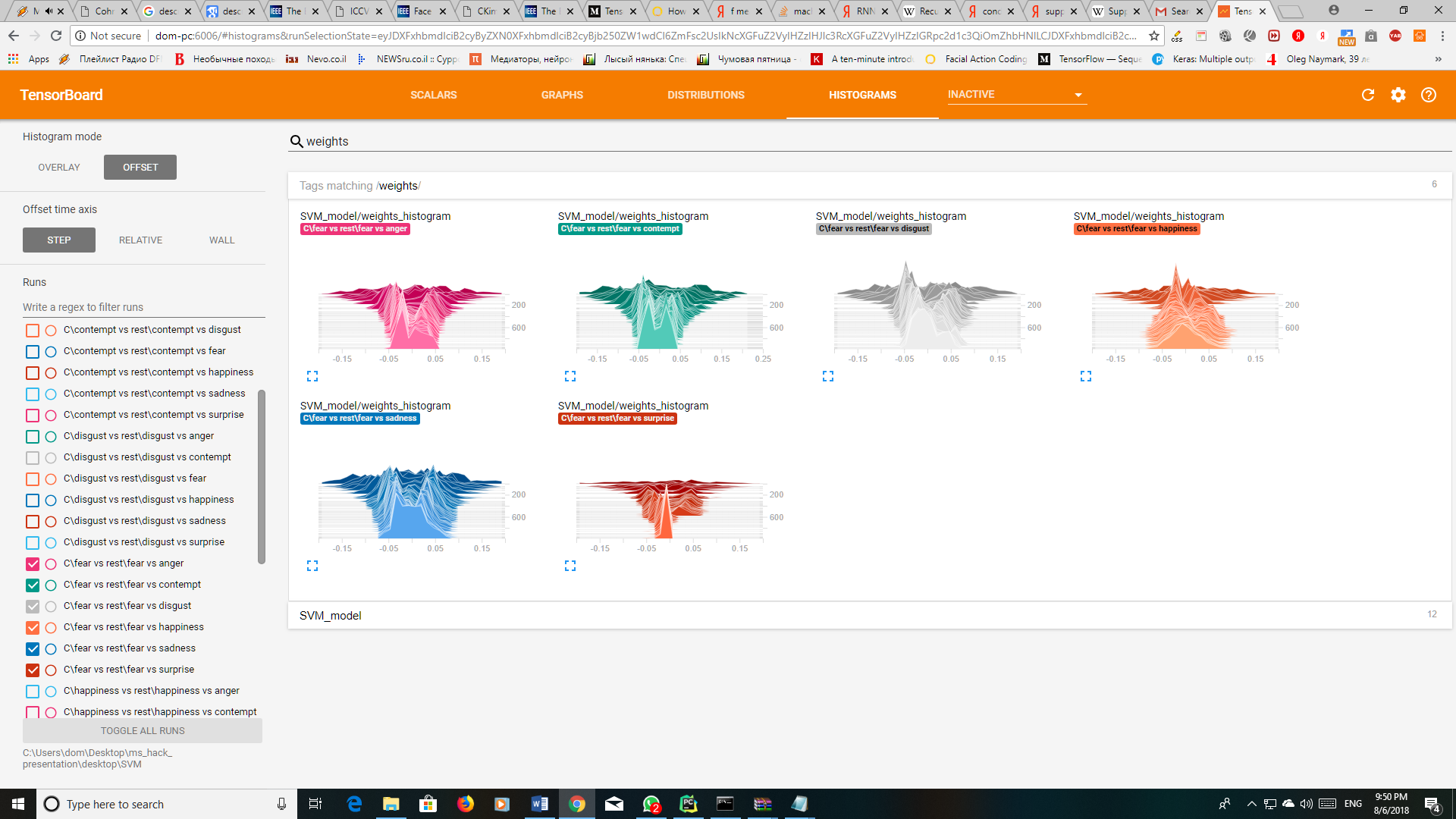
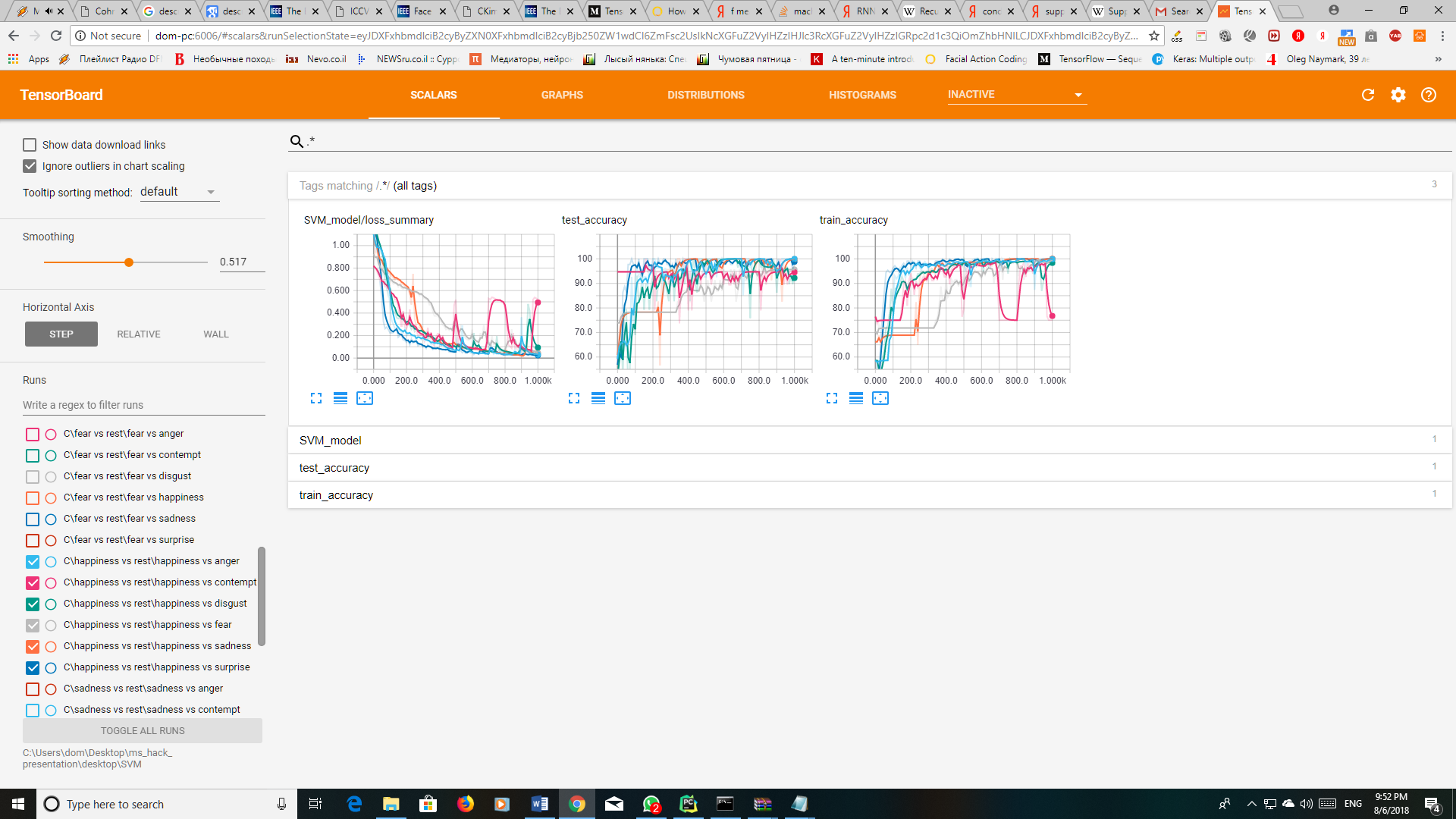
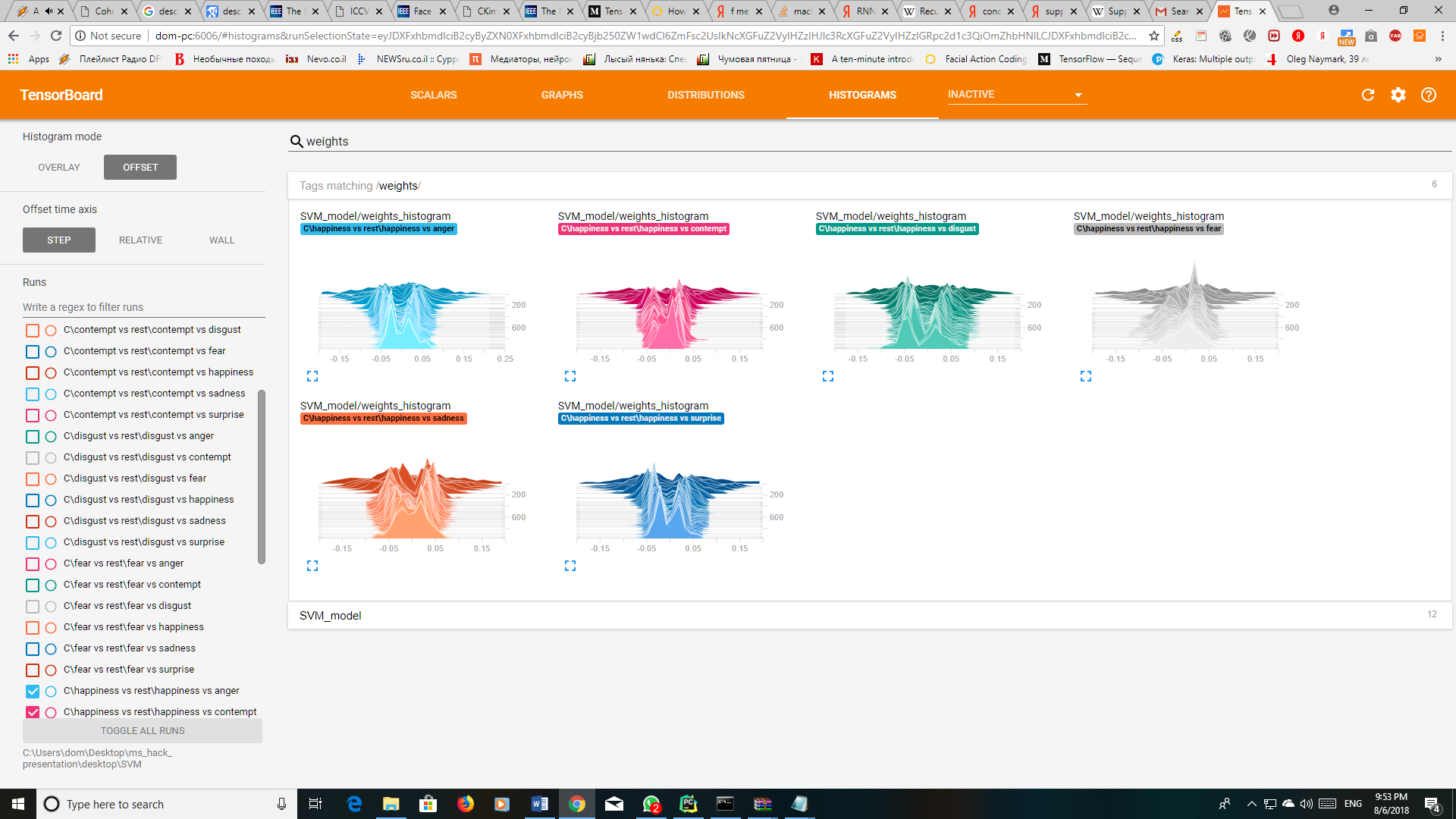
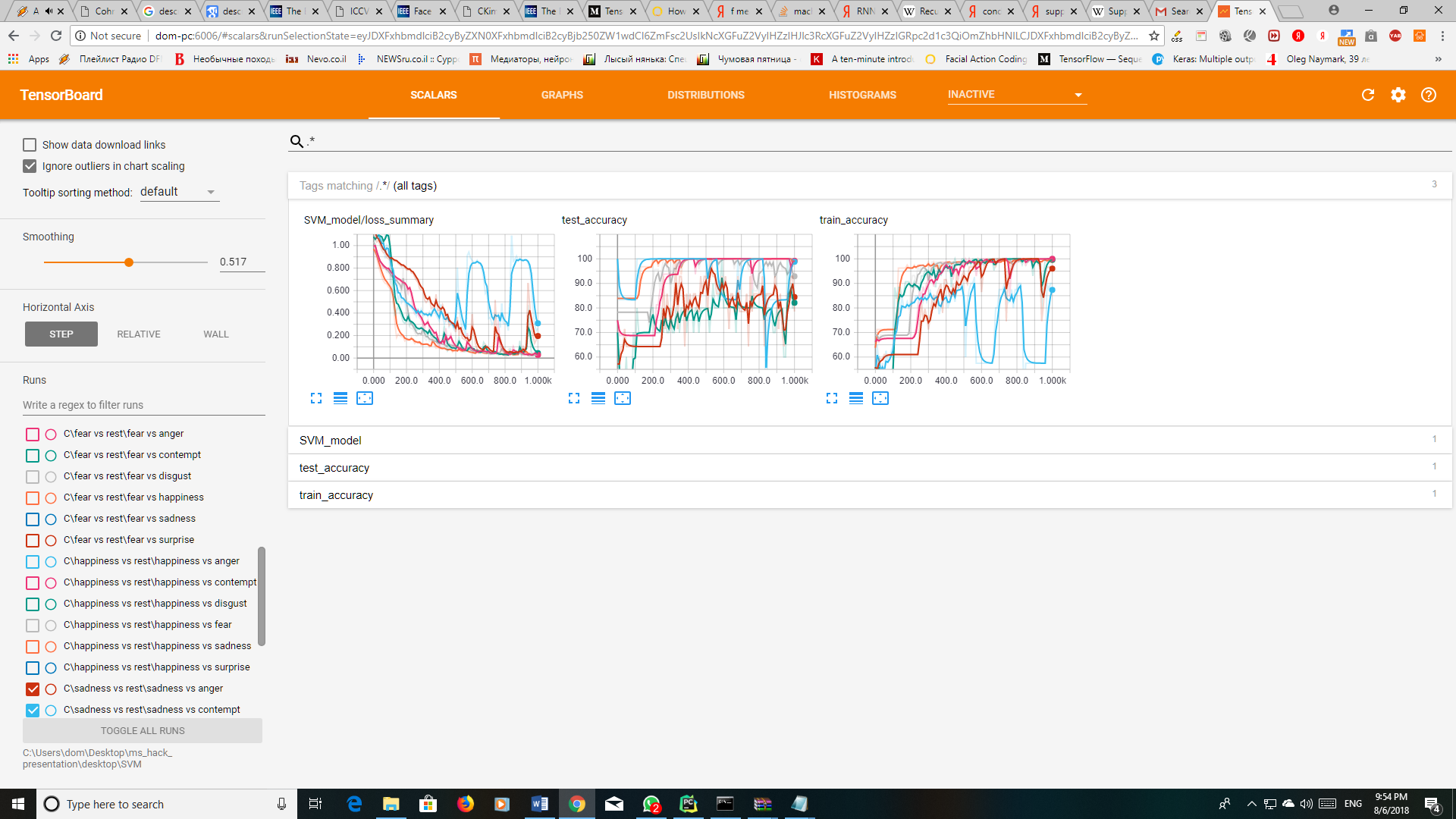
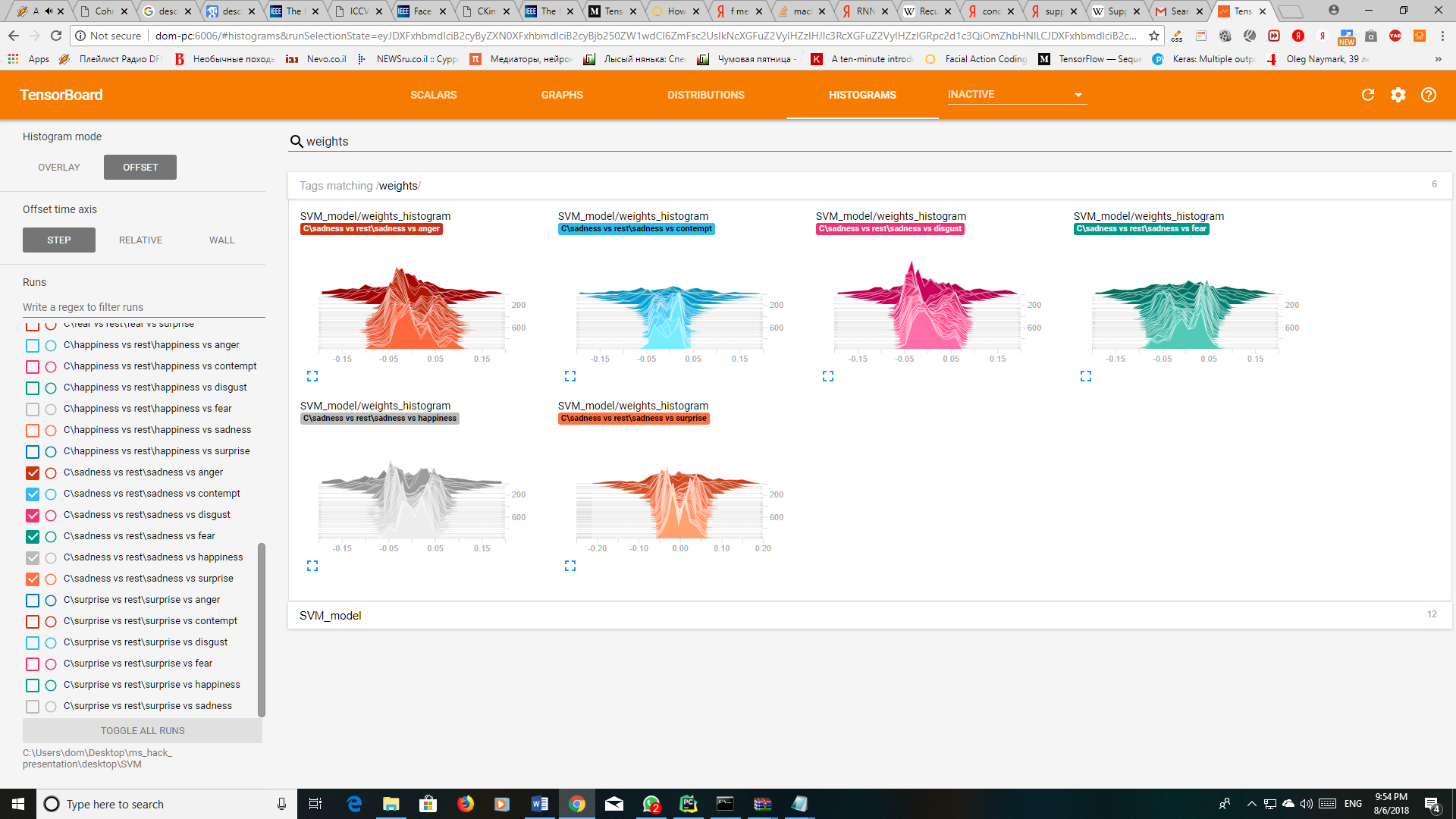
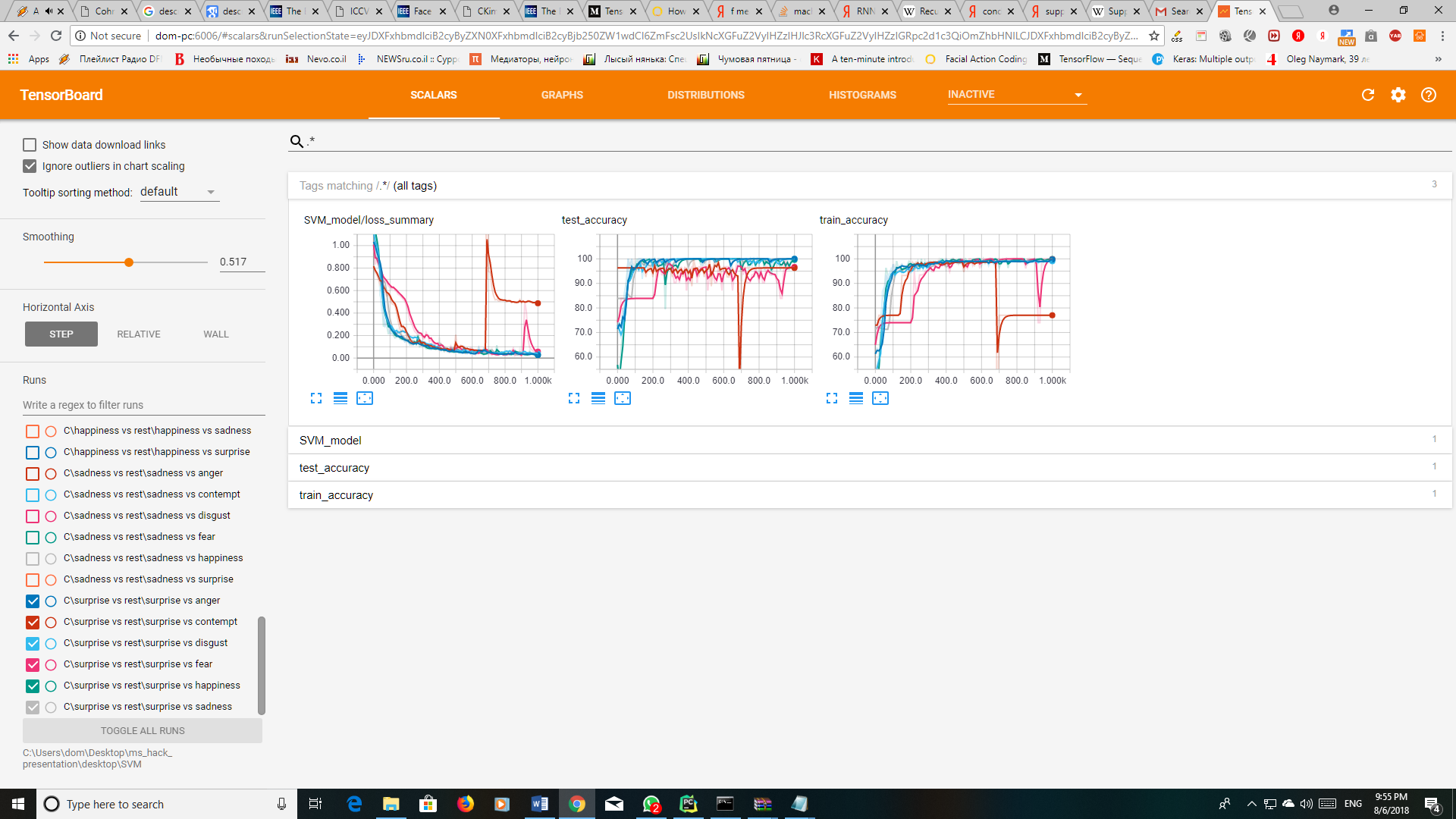
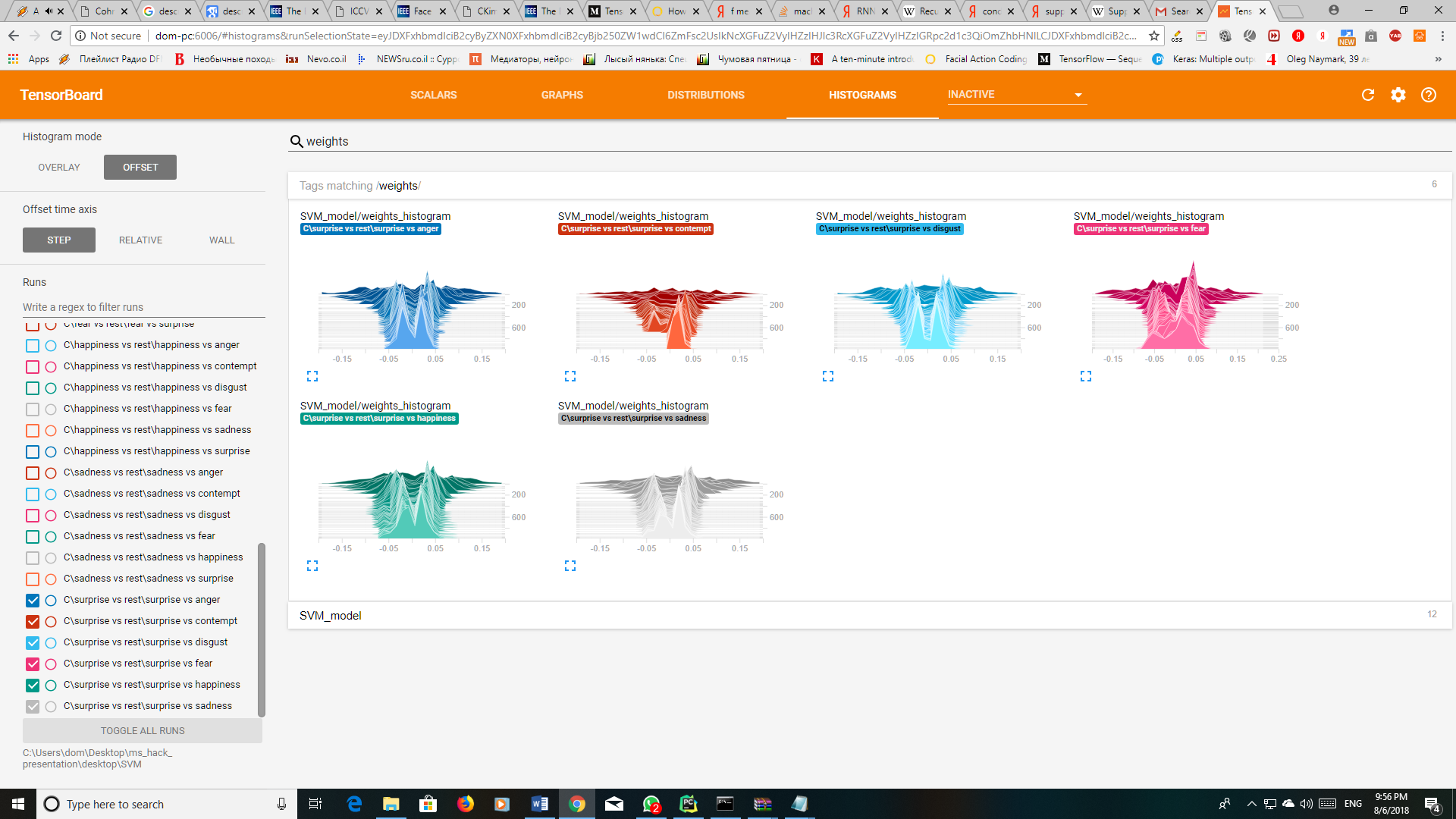
* **AdaBoost algorithm**

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire, who won the 2003 Gödel Prize for their work. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. Unlike neural networks and SVMs, the AdaBoost training process selects only those features known to improve the predictive power of the model, reducing dimensionality and potentially improving execution time as irrelevant features need not be computed.

At this work, the SKlearn library was used to run it, with two Decision Tree Classifiers: SAMME & SAMME.R

* + All results were bad, and we tried a lot of combinations for tuning the system:
  + [SAMME, SAMME.R] x max\_depth in [1,2,3,4,5] x n\_estimators in [300, 600, 900, 1200] x rate in [0.1, 0.5, 1.5, 2.0]. All of them gives about the same result: the average f-measure between 0.08 and 0.38. Difference is the evaluation time: between 40 secs and 16 mins.
  + As AdaBoost is sensitive to noisy data, probably it is the reason.
  + Also, the worst result is always Contempt emotion (about 3% of the data), and the best at Happy and Surprise (each one almost 22% of the total data), therefore the conclusion is that we just don’t have enough data.
  + The last conclusion correlates with the fact that the dimension of each data point (136) is much bigger than number of points at the train:
  + Max\_depth = 2, number of estimators: 600, learning\_rate = 1.5
  + Resulted in accuracy 0.81, time elapsed is 1:24 mins.
* **SVM architecture**In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), support vector machines (SVMs) are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyze data used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis). Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-[probabilistic](https://en.wikipedia.org/wiki/Probabilistic_classification) [binary](https://en.wikipedia.org/wiki/Binary_classifier) [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.  
  Before inputting the data into the SVM model we put it through a RNN to process the sequence (which the SVM cannot handle) into a single matrix.  
  Our SVM model adds another dimension to the dimension of the data points which represents the label. The model then tries to fit a function of a line such that for every x (which is the RNN output, a vector of 256 coordinates) the y coordinate (the label) will have the same sign as the real label of the original sequence inserted into the model and a value greater than or equal to 1 (which in essence represents the distance from the support vector) from the axis of the label.  
  Since our data set is has is not binary labeled we used the one-vs-rest method of converting SVM to a multi-class classifier, for every label we built a classifier for it and another label such that we had 7\*6=42 classifiers. The predicted label of a sequence will be the one that the most classifiers of its binary classifiers outputted +1 for the sequence.

**Results:**

* + **Anger:**
    - **Loss and accuracy:**
    - **Distribution of weights:**
  + **Contempt:**
    - **Loss and accuracy:**
    - **Distribution of weights:**
  + **Disgust:**
    - **Loss and accuracy:**
    - **Distribution of weights:**
  + **Fear:**
    - **Loss and accuracy:**
    - **Distribution of weights:**
  + **Happiness:**
    - **Loss and accuracy:**
    - **Distribution of weights:**
  + **Sadness:**
    - **Loss and accuracy:**
    - **Distribution of weights:**
  + **Surprise:**
    - **Loss and accuracy:**
    - **Distribution of weights:**

**Conclusions**

* The data is labeled with one label when the subject may present multiple emotions.
* Distribute the data more equally (80% and 20% of each class and not of the whole samples).
* It is important to set up the data correctly according to the classifying method used:
  + SoftMax
  + Perceptron Neural Network
  + AdaBoost
  + Recurrent Neural Network (RNN)
  + Support Vector Machine (SVM)
* RNN yielded the best result, possibly due to it being designed learn from sequences more efficiently.
* Some classes didn’t have enough samples to learn from, which decreased our model’s accuracy.
* Machine learning models have a difficulty handling noisy data, unlike deep learning models.
* Machine learning models suffer more from lack of data than deep learning models.

**Perspective for further investigations:**

1. Pretrain the data through additional step: Image -> Landmarks (facial detection)-> Action Units (muscle movements) -> Emotions.
2. Analyze video learning dynamics of changes:
   1. deltas between two neighbor frames
   2. timing factor: how fast did it change.
3. Examine additional methods: CNN, PCA, other algorithms.
4. Try to solve the problem through unsupervised learning.
5. Try and obtain more data.
6. Considering the pose, head movement, angles.
7. Change the data to accommodate to multi-label classifiers.
8. Combine model and use their individual strengths on different parts of the problem.

**Sources:**

**About Facial Action Units:**

<https://imotions.com/blog/facial-action-coding-system/>

<https://en.wikipedia.org/wiki/Facial_Action_Coding_System>

**Data set for the Model Training:**

Cohn-Kanade (CK and CK+) database: <http://www.consortium.ri.cmu.edu/data/ck/>

**Paul Ekman:**

<https://en.wikipedia.org/wiki/Paul_Ekman>

**Ada Boost:**

<https://en.wikipedia.org/wiki/AdaBoost>