# **Automated Smart TV UI Performance Testing with Visual Recognition**

|  |  |
| --- | --- |
| Peng (Peter) Li  Stanford University  peter888@stanford.edu | Shim-Young (Ricky) Lee  Stanford University  sylee1@stanford.edu |

**Abstract**

*In this project, we applied the current convolutional neural network based image classification, and cluster algorithm to automation of Smart TV UI performance testing. Applications such as YouTube specify upper bounds for launch and page transition time that device manufacturers need to meet, and this performance test is currently done manually. Although measuring page transition time is a relatively trivial task for humans, the fact that application displays different texts and images each time (due to its recommendation system) precludes simple solutions such as pixel comparison. At the current stage of development, the convolutional neural network (Inception v3) [3] was shown to be able to classify different types of UI pages extremely well. We will now focus on the detection of movement of displayed images using object localization.*

1. Introduction

Smart TV applications such as YouTube and Netflix have performance requirements. YouTube certification program specifies requirements regarding UI performance for OEM providers, system integrators and smart TV vendors that wish to ship their devices with YouTube application. Among those requirements is page transition performance, which refers to the time the transition between different types of pages (Guide, Menu, Settings etc.) takes. The YouTube certification program provides upper bounds needed to be met for different types of page transition. Although measuring page transition time seems like a trivial task for human eyes, it is currently measured manually, and the goal of this project is the automation of this performance testing procedure using current computer vision techniques.

Deep learning based visual recognition algorithms such as Convolutional Neural Network can be used to recognize different pages and measure application's launch time and page loading time. Sub-page transition, which refers to the movement or change of displayed images or highlighting windows within the same UI page and loading time, can be recognized with object localization.

To automate the measurement of page transition, we retrieve an image at each time step and run image classification algorithm to determine whether the YouTube application has transitioned from displaying one type of page to another. Additionally, we can incorporate some prior knowledge about the UI to eliminate some of the possible outcomes to improve the classification result. We call this “context dependent”, since the classifier relies not only on the input images but also other “context” information. For image (screenshot) retrieval and collection, we will use BlackMagic image capture card. The final outcome is a program that automatically measures the page transition performance.

Intuitively, the evaluation metric should measure how accurately the program classifies the transition (which page to which page) and pinpoints the time of transition, which allows us to measure the page transition performance and ultimately automate the testing procedure. Note that this is not the same as the classification rate (accuracy) of the neural network for collection of screenshots. The details of the metric are parts of the engineering challenges of this project and will be developed as we progress.

2. Related Work

Although the project itself has a very specific application, the core engineering task is a fairly standard image classification. Thus, we plan to review literatures on important neural network architectures that have come out in recent years. Starting from the convolutional neural network architecture used in ImageNet competition by Alex Krizhevsky *et al.* [1], we reviewed Inception architectures [2, 3], and ResNet [4] to name a few.

To detect and measure the sub-page transitions, we are considering the use of various deep learning based object detection schemes [5,6,7] and video classification algorithms [8].

Regarding the YouTube certification program for devices, we refer to “YouTube TV HTML5 Technical Requirements 2017”[9], which specifies different test and criteria needed to be performed and met.

3. Methods

The automated performance testing can be divided into mainly two stages: 1) image classification and 2) object detection and localization. In the first stage, the automated testing program will retrieve an image at each time step as the YouTube application launches and runs, and an image classification algorithm determines what type of page (UI) the application is currently displaying, and the program measures the launch and transition times based on the classification result.

Once the application is loaded and all texts and images are displayed, the white highlighting window slides right and left, sequentially highlighting the video thumbnails in the queue (see figures 2 and 3). To measure the time needed for transition, we use object localization to locate and detect the movement of white window. We decided that a typical image classification scheme is not the most appropriate detection method since the sliding window can be at any position (horizontally) due to its continuous movement from one thumbnail to another. Therefore, it is not easy to determine how many classes there are beforehand, and furthermore, such scheme would depend on how we discretize the time step, making the data gathering process more complex.

3.1 Image Classification

In the first part, we have a typical classification problem. The final program needs to perform image classification at each time step, and the convolutional neural network was trained with 5 different page (UI) types ().

**3.1.1 Model Architecture**

Currently, we are using Inception-v3 classifier [3] provided by the latest version of TensorFlow. We haven’t focused on the careful model(architecture) selection or tuning of hyperparameters but this classifier achieved a good result.

When the application launches, it only displays different pages sequentially, in the same order. In our training, however, we did not use this fact and the classifier chose from 5 possible outcome values each time. Since given the current page, the application can only keep displaying the same page or transition into the next page, we can shrink the number of possible outcomes to two at each stage, or even replace the classifier with simpler binary classifiers. We will test and compare such architectures.

**3.1.2 Simplified CNN Model**

We may later simplify the model and use more basic CNN architecture resembling the model by Alex Krizhevsky *et al.* [1] (the “AlexNet”) if simpler networks can be shown (empirically) to perform equivalently well. The details of architecture (number of layers, filters, FC layers) will be added once we finalize the model.

**3.1.3 Other models experimented**

We also applied Linear Regression, SVM and k-NN for learning and compare purpose.

**3.2 K-Mean’s Cluster Model**

Used balanced K-Mean’s Cluster Model is able to get 95% training time accuracy and 92% test accuracy.

When using the training model to classify the images, the always have sequence of images, apply that sequence information, then the classifier only going to miss few of frames, thus the result still within certain range, say few is 3 (because 0.07 \* 0.07 \* 0.07 = 3.43e-6), the performance impact only 3/60 = 0.05 seconds.

**3.2.1 Validate and Test Accuracy**

The validate or test the model, use balanced labeled data, for each class choose same amount of samples, say 100, and each class selection put in same sequence. Thus the data will be have indices [1:100, 101: 200, 201: 300, 301:400, 401:500], and bellows to real right labels [0, 1, 2, 3, 4]

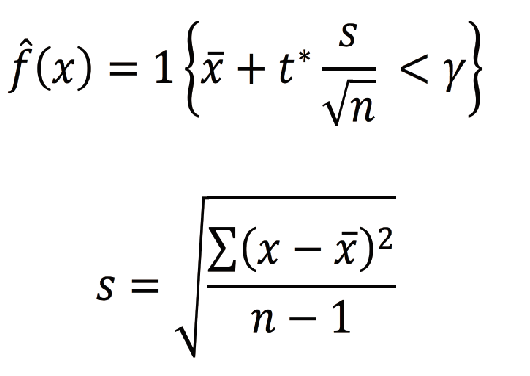
**3.2.2 Majority Element Algorism**

To matching the labels correctness, as a result of K-mean algorism just classifies different classes, but intent to have right label which matching exactly the labels out validate or test data we have. Will use Bayer-Moore majority vote algorism [] to decide the mapping to right labels.

**3.3 YouTube Launch Performance Measurement**

For our targeting problem, we are try to measure the launch performance after we classified each test frames. We need to launch YouTube and do the classification lot of times to get the accurate launch performance.

The final statistic is given by the confidence interval (90%) based on the mean and the standard deviation calculated from the multiple measurements of the launch time. The decision rule for the performance test is whether the upper bound of the confidence interval is less than the required launch time specified by the smart TV applications.



4. Dataset and Features

**4.1 Data Types**

When YouTube launches, it sequentially displays 5 different pages – White Screen, Logo, Spin Loading, Text Loaded, and Image Loaded. White Screen and Logo pages are static pages, lacking dynamic elements. Spin Loading page only has a spinning wheel in the center of the screen.

Shown below are typical images for YouTube application. The pages of class White Screen, Logo, Spin Loading are not shown. The labeled YouTube screenshots can be found in: <https://goo.gl/QGJ3i3>. Sub-page localization data can be found in: <https://goo.gl/PuzVHD>.

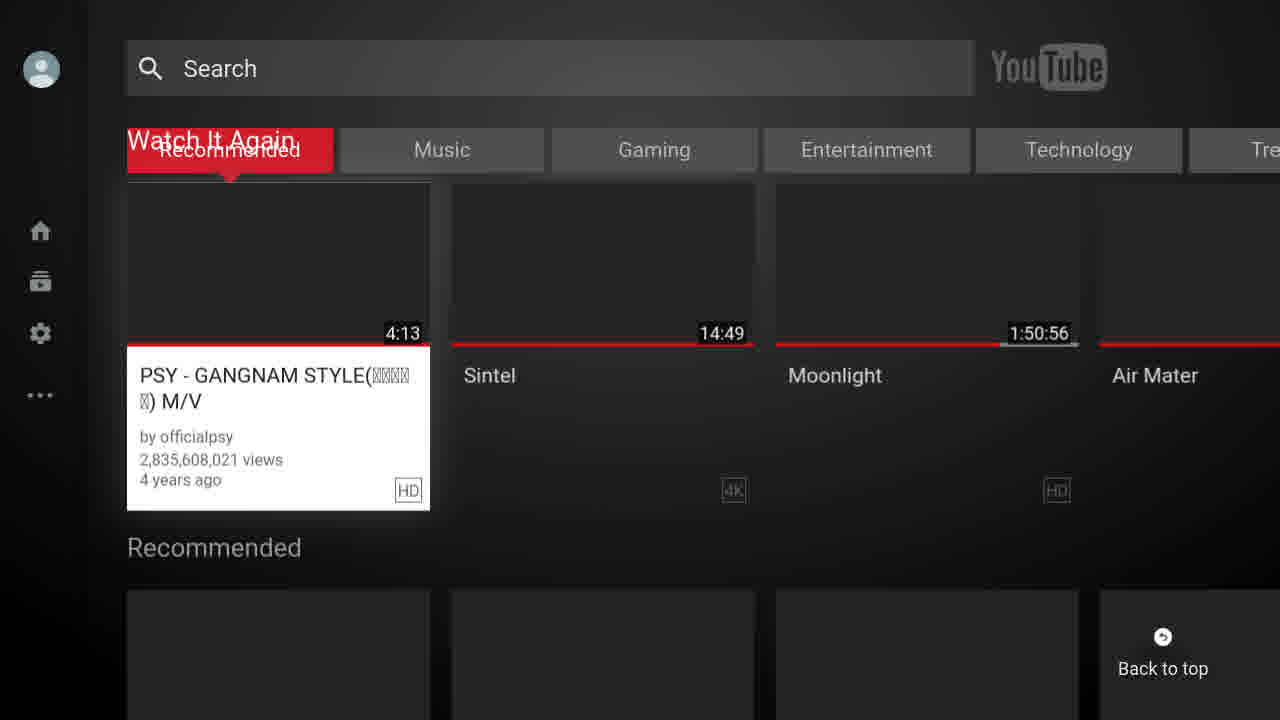


Figure 1: Image of type “Text Loaded”. The text may change for different launches, making simple pixel comparison methods difficult.

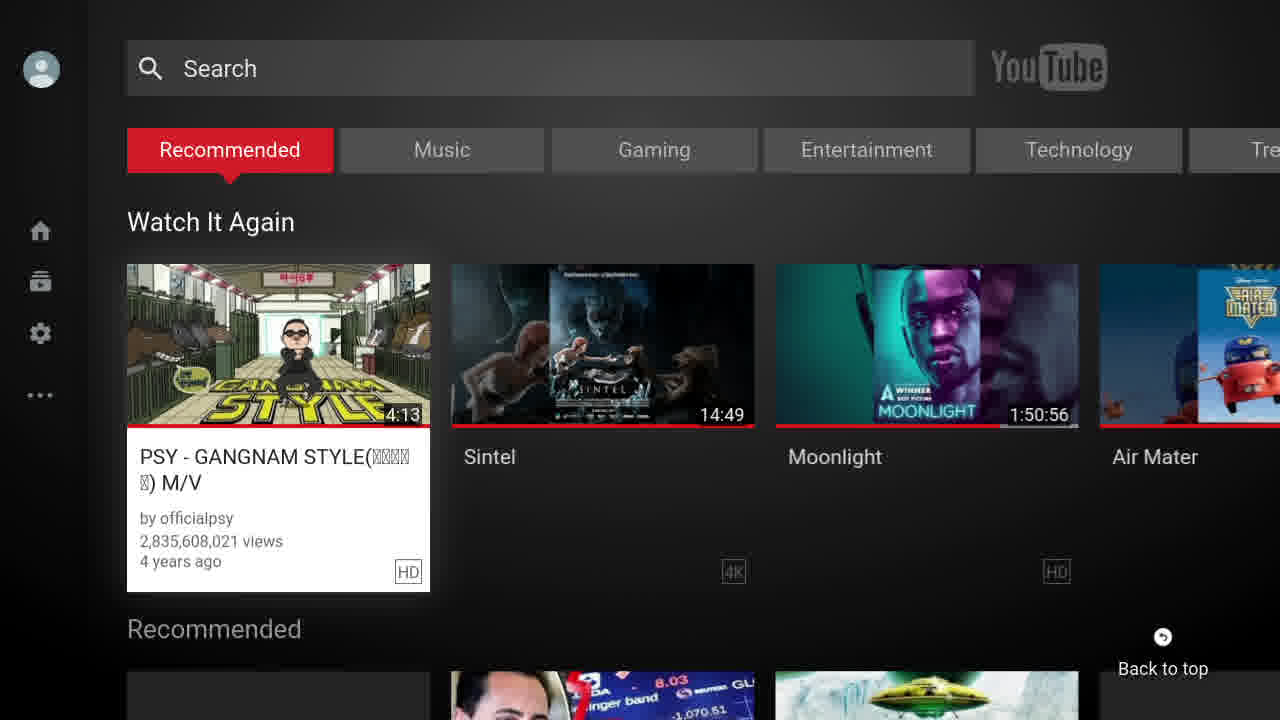


Figure 2: Image of type “Image Loaded”. Again, video thumbnails and text may change for different launches.

**4.2 Data Acquisition and Labeling**

For image retrieval and collection, we used Decklink image capture card from BlackMagic [10], the capture is based on STB's output 720P, and frame rate is 60Hz, capturing 60 screenshots per second. The images were labeled in a semi-automated fashion: since the pages are shown sequentially, we used the average launch time and the average durations for which each page is displayed to roughly divide and save the images in subfolders. Then each folder was manually checked to move the wrongly assigned images to the correct folders.

Each class has over 400 images, and the dataset contains a little more than 4000 labeled images in total. To gather create this dataset, the application was launched more than 1000 times, which took roughly 5 hours. The current size of the dataset is quite small compared to a typical dataset size for training of neural networks, but the result suggests that the current size is sufficient. The number of training samples needed depends on the model choice, so we will gather more data as needed since we can arbitrarily increase the size of our dataset.

**4.3 Pre-process Images**

The captured images are re-sized to 36x64 and 144x256 grey images. In which 36x64 are for experiments, and 144x256 for training purpose. By using the PIL library come with Python.



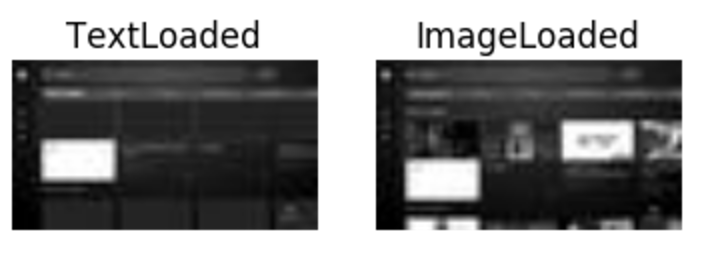


Figure 3: Images after pre-process.

**5. Results**

**5.1 classification results**

Several different convolutional neural network (CNN) architectures as well as other machine learning algorithms were tested, such as Inception V3, 1-convolution network, 1-hidden layer multi-layer perceptron, linear classfier (with softmax), SVM and k-NN.

5.1.1 CNN network

For CNN network, we use one 4x4 filter for the hidden layer, and a full dense layer to classify to 5 classes. And we use hinge loss for the loss function.

The training loss graph converges fast and well.

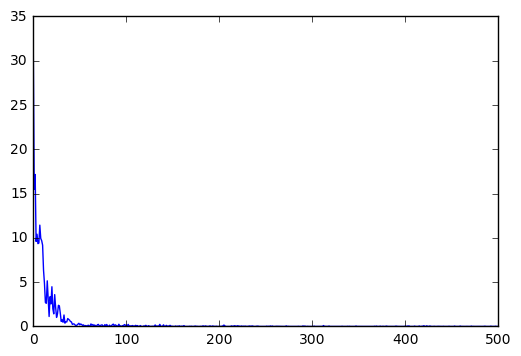


Figure 4: The training time loss graph converges fast and well for CNN.

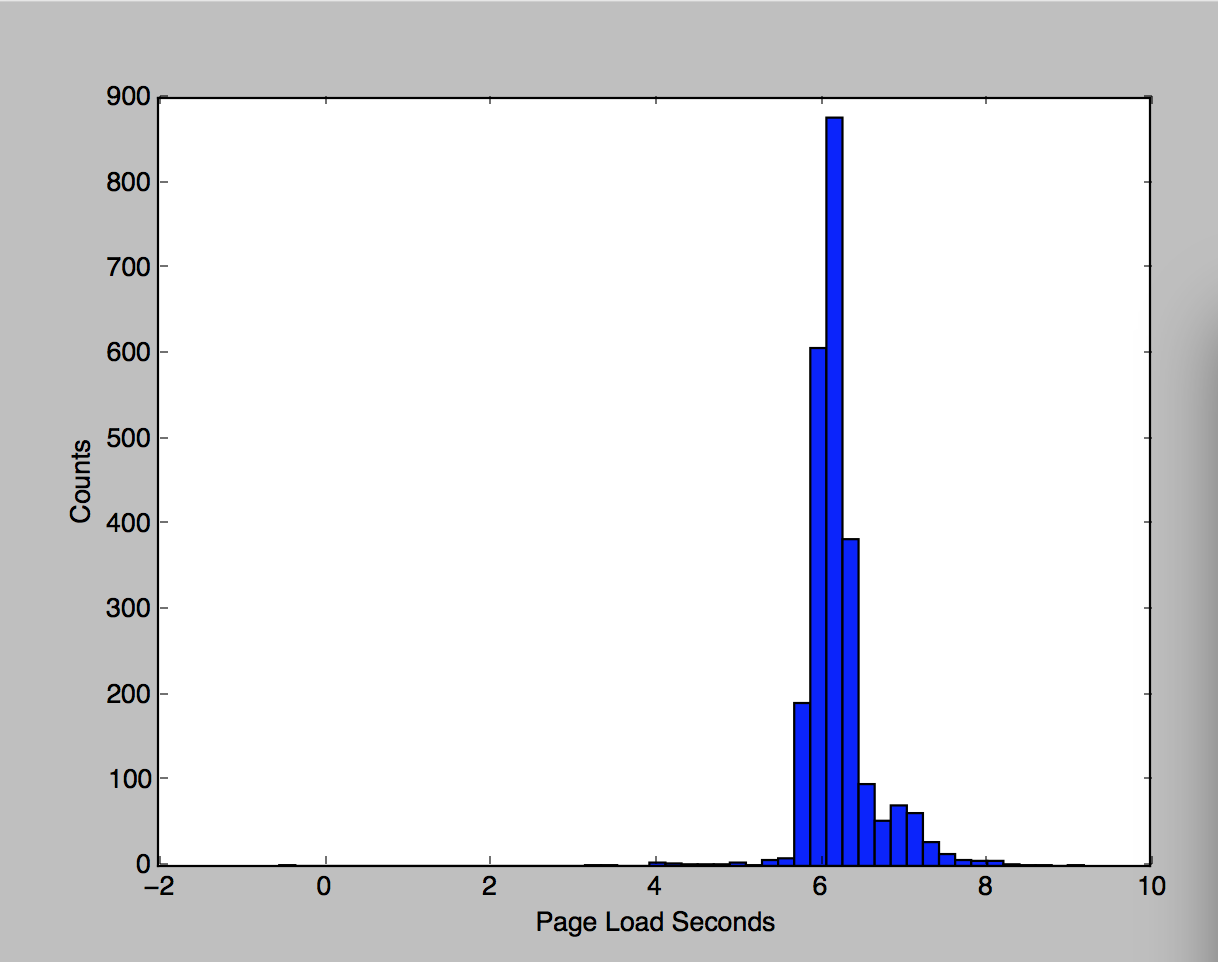


Figure 5: The final launch time histogram

5.1.2 Accuracy and test run time.

All CNN’s, from Inception V3 to the simple 1-convolution network, achieved 100% validation accuracy.

Linear classfier with softmax was the fastest classfier.

Non-CNN classifiers could perform as well as the CNN’s because of the little in-class variance of the images.

|  |  |  |
| --- | --- | --- |
| Model | Test Accuracy | 100 Test Data Execution Time (seconds) |
| Linear | 0.998 | 0.02 |
| K-NN | 0.97 | 38.81 |
| SVM | 0.89 | 0.91 |
| Cluster(K-Mean) | 0.92 | 0.02 |
| Simple-CNN | 1.00 | 0.16 |

Figure 6: Different model test time performance.

5.2 un-supervised learning results.

Although the labeling data for this problem is easy and doable, but thinking about extends this work to other UI performance automation and as the 100% accuracy results easy to get. We also tried k-means to cluster our data. Got 95% training time accuracy and 92% test time accuracy.

We picked up 100 test data for each classes. Use the training k-mean estimator to predict result. Below is the test result.

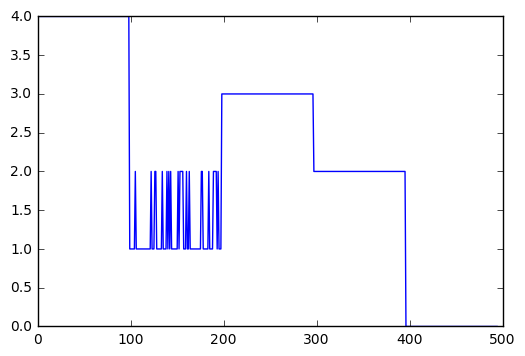


Figure 7: Clusters Test Result for Test Data

5.3 The YouTube launch performance measurement.

|  |  |
| --- | --- |
| To Page | Finished at time |
| White screen | 0.02 |
| logo | 0.63 |
| Spin loading | 3.63 |
| Text loaded | 7.95 |
| Image loaded | 8.13 |

**Figure 8: one example of loading progresses**

**6. Conclusions and Future Work**

As the result section shows, we have successfully applied current deep learning based visual recognition to UI performance Test Automation. Currently, we have implemented the image classification with transfer learning, using the Inception V3 [2] architecture by Szegedy *et al.* and achieved 100% validation and training accuracy, showing that he convolutional neural network can easily deal with a seemingly trivial but tricky classification problem.

In the future, similar pipeline and training procedure can be applied not only to similar UI performance testing of different applications, but also to all UI related test automation that requires similar computer vision capabilities. For example, similar system can be used to iOS and Android UI related test automation, because the details of the system can be adjusted to match iOS and Android UI's image resolutions and test criteria. This can be easily achieved by a pre-processing of training and test image and if necessary, tweaking of model architecture parameters.

When this paper been conduction, iOS11 just announce on WWDC2017, which has the Core Machine Learning Tools for vision tasks. And with Core ML Tools, we can train model in server, and predict the classification results with trained model in iOS11 devices. So, the performance measurement have great chance been done within a single iOS APP.

**References**

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105, 2012.
2. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 1–9, 2015.
3. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. arXiv preprint arXiv:1512.00567, 2015.
4. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.
5. R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.
6. J. Huang V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, K. Murphy, Speed/accuracy trade-offs for modern convolutional object detectors, In *CVPR* 2017
7. J. Dai, Y. Li, K. He, J. Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks, In *NIPS*, 2016
8. A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 1725–1732, 2014.
9. YouTube TV HTML5 Technical Requirements. <https://drive.google.com/file/d/0B0C4aKjz5kL6X3hsTUg5NjJCaWs/view?usp=sharing>
10. Decklink Capture Card from Black Magic Design <https://www.blackmagicdesign.com/products/decklink>
11. Boyer-Moore majority vote algorithm <https://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_majority_vote_algorithm>
12. iOS11 machine learning API <https://developer.apple.com/machine-learning/>