***A Project Report On***

**Youtube-8M Challenge**

(A large-scale video classification)

*Submitted By*

**Aakash Shah (axs165231)**

**Rajan Jhaveri (rjj160330)**

**Hardik Punjabi (hvp160030)**

**Varun Dani (vxd162230)**

**CS6375.004**

**Machine Learning**

**ABSTRACT**

In this project, we are trying to classify a large-scale video dataset. We are trying to classify the video classification dataset provided by “GOOGLE” which is labelled using “YOTUBE VIDEO ANNOTATION SYSTEM”. It is composed of approximately 8 Million videos -500 K hours of video-annotated with 4716 visual entities.

For the model building and classification task we are using the TensorFlow framework which is built on top of python. TensorFlow converts the dataset into collections of tensors. Using some techniques that will be described later in the report we are trying to classify the video data and label it. The goal is to build a classification model and tune it using appropriate parameters and train it on the video classification dataset which is of 1.7 TB in size. After the model is trained the testing dataset is fed to the model which consists of 700k video data and the model should provide the confidence level for all 4716 labels provided, from which the top 20 labels will be output and the model’s performance will be measured on these 20 labels.

Initial start-up code for the project is provided by google which gives us a performance of 69% on certain parameters. The main goal of the project is to select the best model and tune it using the best parameters possible.

**Introduction:**

The project topic is a current online competition being hosted by [kaggle.com](https://www.kaggle.com/c/youtube8m). The goal of the competition is to predict the labels of YoutTube videos. We are provided extracted frame level and video level features by google. The feature data and detailed feature information can be found on the <https://research.google.com/youtube8m> webpage. The training dataset in this competition contains videos and labels that are publicly available on YouTube, while the test data is not publicly available. The test data also has anonymized video IDs. We are supposed to find the best classification model for this data set and predict the confidence level of all 4716 labels provided, out of which top 20 labels and their confidence levels are the output and the performance of the model is measured by comparing these 20 labels and their confidence levels to the actual labels and their confidence levels of the video-data that is part of the testing dataset. The main difficulty or objective of this challenge is to find the best parameters for the classification model.

The training dataset consists of 31GB of video level data and 1.7TB of frame level data. Due to the large volume of dataset using the local machine for training of the classification model is almost impossible. For this reason, we have used Google-Cloud to train and test the model.

**Dataset Description:**

“Youtube 8M Challenge” project is a part of the Kaggle competition and the dataset we are using for it is provided by Google and can be found at :

Video Level Dataset

<http://us.data.yt8m.org/1/video_level/train/index.html>

<http://us.data.yt8m.org/1/video_level/validate/index.html>

<http://us.data.yt8m.org/1/video_level/test/index.html>

Frame Level Dataset

<http://us.data.yt8m.org/1/frame_level/train/index.html>

<http://us.data.yt8m.org/1/frame_level/validate/index.html>

<http://us.data.yt8m.org/1/frame_level/test/index.html>

This dataset consists of 2 different types of data that describe a video.

1) Video Level

2) Frame Level

The Video level dataset provides aggregated values of the audio and video features per second. The dataset can be accessed directly on Google Cloud or can be download to local machine. Total size of Video level dataset is 31GB. Files are in TFRecords format. Each video has

1. "video\_id": unique id for the video, in train set it is a Youtube video id, and in test/validation they are anonymized
2. "labels": list of labels of that video
3. "mean\_rgb": float array of length 1024
4. "mean\_audio": float array of length 128

The Frame level dataset explains the whole video frame by frame. It can be thought of as a description of every frame of the video. All the videos are taken as 60 frames per second. The dataset can be accessed directly on Google Cloud or can be download to local machine. Total size of the dataset is 1.71TB. Files are in TFRecords format. Each video has

1. "video\_id": unique id for the video, in train set it is a YouTube video id, and in test/validation they are anonymized.
2. "labels": list of labels of that video.
3. Each **frame** has "rgb": float array of length 1024,
4. Each **frame** has "audio": float array of length 128

**train.csv, validation.csv** - the training and validation sets ground truth are available on Google Cloud: [train\_labels.csv](http://us.data.yt8m.org/1/ground_truth_labels/train_labels.csv) and [validate\_labels.csv](http://us.data.yt8m.org/1/ground_truth_labels/validate_labels.csv), or downloadable with gsutil:gs:<//us.data.yt8m.org/1/ground_truth_labels/train_labels.csv>, gs:<//us.data.yt8m.org/1/ground_truth_labels/validate_labels.csv>

* **VideoId** - the id of the video
* **Labels** - the correct labels of the video (space delimited)

**label\_names.csv** - a mapping between label\_id and label\_name (available on Kaggle)

**sample\_submission.csv** - a sample submission file in the correct format (available on Kaggle)

* **VideoId** - the id of the video
* **LabelConfidencePair** - space delimited predictions and their probabilities. For example, 1 0.6 2 0.4 means label 1 with 0.6 probability, and label 2 with 0.4 probability for this video.

The dataset provided by google is pre-processed.

**Techniques Used:**

Techniques used for training the model are as follows:

* Techniques used for training on video-level data:

1) Logistic Model:

Linear projection of the output features into the label space, followed by a sigmoid function to convert logit values to probabilities.

2) Moe Model:

A per-class softmax distribution over a configurable number of logistic classifiers. One of the classifiers in the mixture is not trained, and always predicts 0.

* Techniques used for training the model on the Frame level data:

1>LSTMModel:

Processes the features for each frame using a multi-layered LSTM neural net. The final internal state of the LSTM is input to a video-level model for classification. Note that you will need to change the learning rate to 0.001 when using this model. It is a type of Recurrent Neural Network.

2>DbofModel:

Projects the features for each frame into a higher dimensional 'clustering' space, pools across frames in that space, and then uses a video-level model to classify the now aggregated features.

* FrameLevelLogisticModel:

Equivalent to 'LogisticModel', but performs average-pooling on the fly over frame-level features rather than using pre-aggregated features.

**Performance Metrics Used:**

**Performance metrics Used areas follows:**

**1>Hit@1:**

In this performance metric, we check if the first label predicted by the classification model is same as the actual label of the video or not, if the label is same as the actual video’s first label then we assign a value of 1 or 0 otherwise. We sum over all the videos and divide it with the total number of videos. This values gives us the percentage of videos in which our model predicts the first label of the video data correctly. The formula for that can be explained as follows:

**2>PERR:**

The full form of PERR is precision equal recall rate. It is also known as the breakeven point where the value of precision is equal to the value of recall. Precision measure considers true positives and false positives. Recall measure considers true positives and false negatives. The precision measure fails to consider false negatives and the recall measure fails to consider false positives. The perr value solves this problem and considers true positives, true negatives and false positives. The higher values of perr proves that your model is better. The formula for precision and accuracy are as follows:

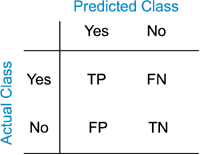


Figure 1:Confusion Matrix

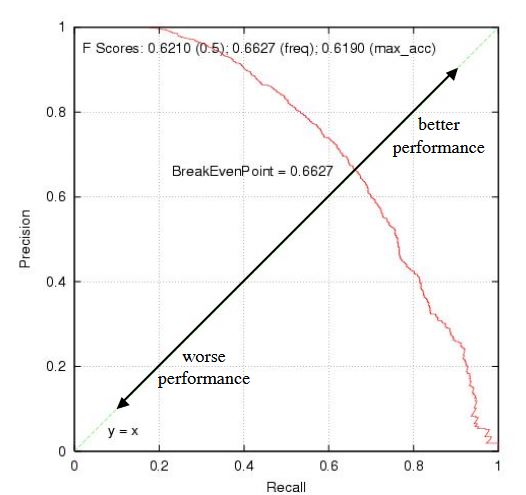
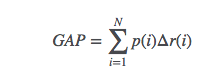


Figure 2:Precision Vs Recall

**3>GAP (Global Average Precision)**

We evaluate the submissions by the Global Average Precision (GAP) at k, where k=20. For each video, we predict a list of labels and their corresponding confidence scores. The evaluation takes the predicted labels that have the highest k confidence scores for each video, then treats each prediction and the confidence score as an individual data point in a long list of global predictions, to compute the Average Precision across all the predictions and all the videos.

If a submission has N predictions (label/confidence pairs) sorted by its confidence score, then the Global Average Precision is computed as:

****

**1**:GAP Value

where N is the number of final predictions (if there are 20 predictions for each video, then N = 20 \* #Videos), p(i) is the precision, and r(i) is the recall.

**Graph of Model Computations:**

One of the great feature TensorFlow is that it allows building the graph of model computations which gives the visual representation of how the model performed the computations this allows us to identify the problems with the model if any and better understand the working of each model. The model training is performed video level as well as frame level dataset. Below are the graph generated for model on video-level and frame level model training.

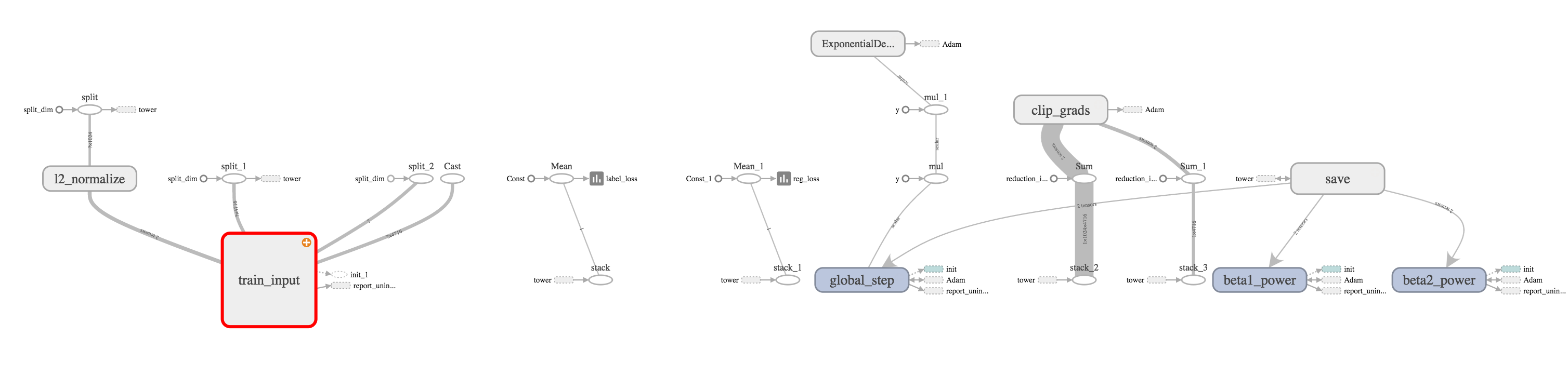
1> Graphical Representation of MOE model using Adam optimizer applied to video-level data

Figure 3:Model Computation(Video Level) Part:I

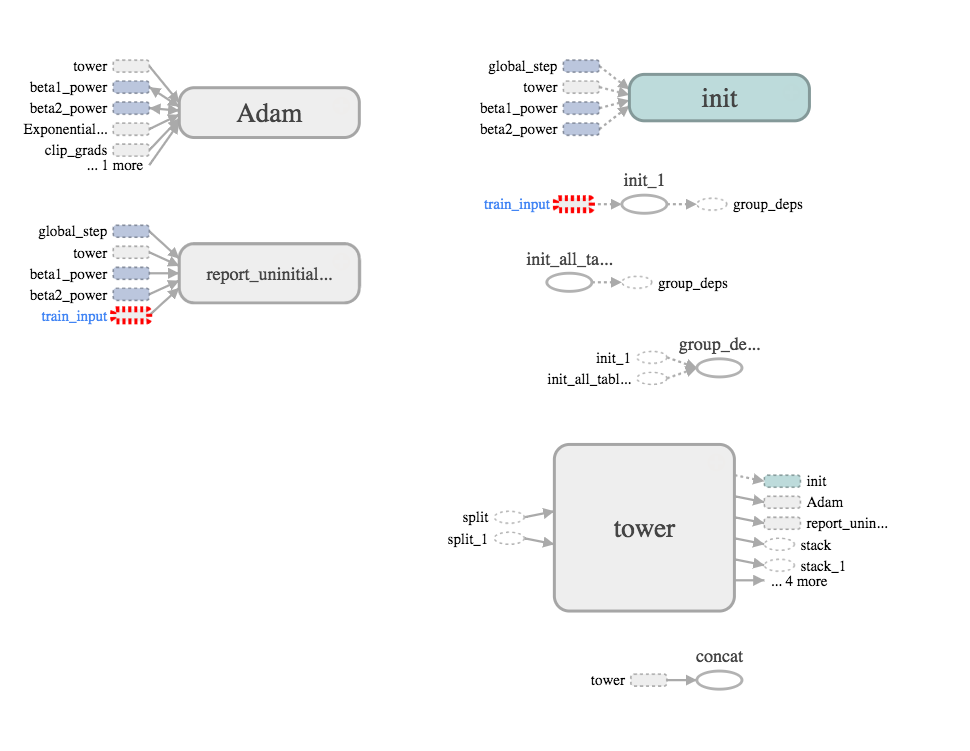


Figure 4:Model Computation (Video Level) Part:II

2> Graphical representation of Logistic Model using Adam Optimizer applied to frame level dataset:

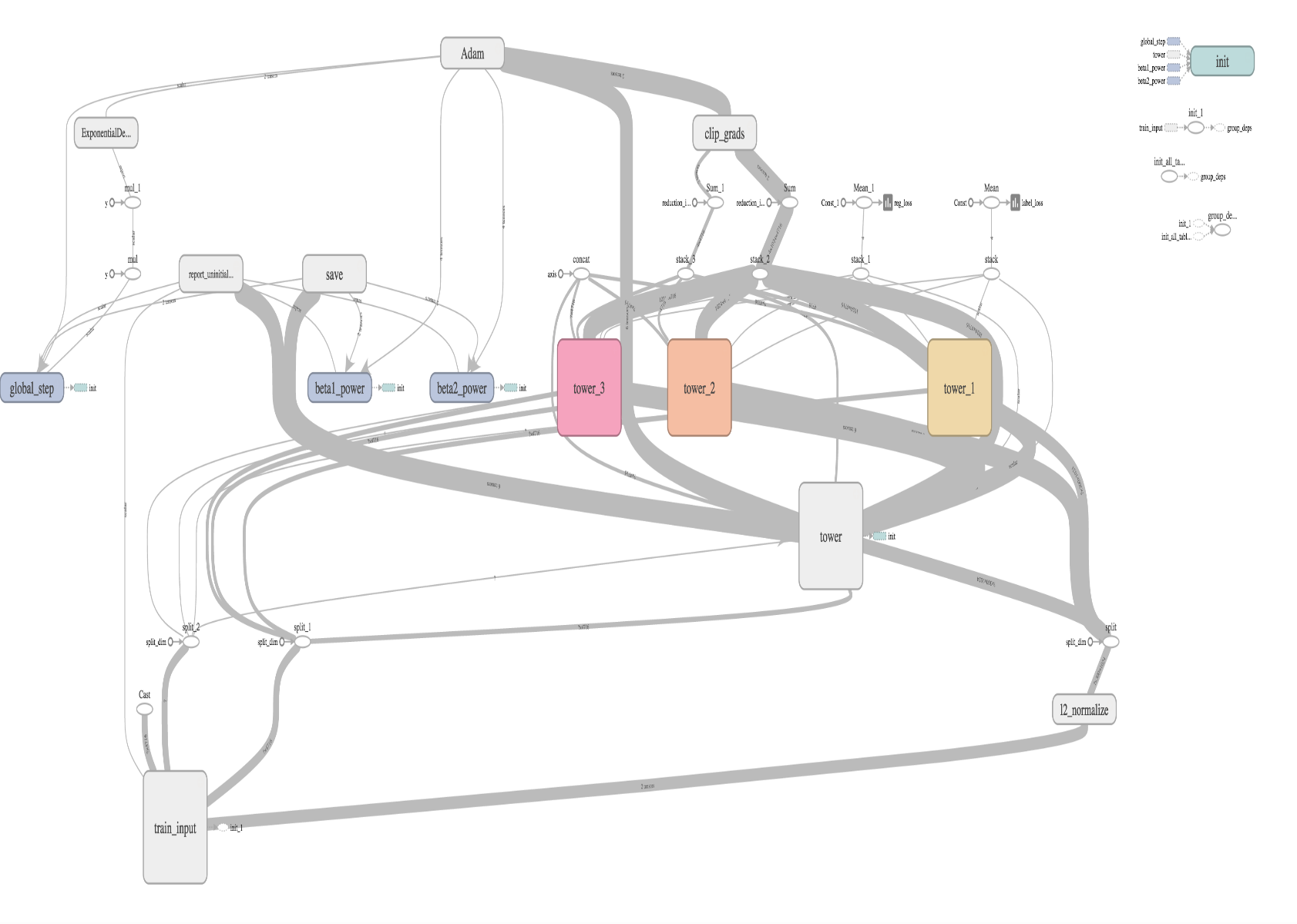


Figure 5:Model Computation (Frame Level)

**Graphical Representation of Performance:**

As the main difficulty of the project was to find the best model and the best parameters for those models, building graphs makes the task of decision making easy. Different graphs plotted using TensorFlow’s in-built functionality called Tf.graph() allows us to visualize the effects of different parameters on the model which in turn allows to select the best parameters for the model and comparing the performance of different models allows to select the best model. We have trained the model using video-level and frame level dataset.

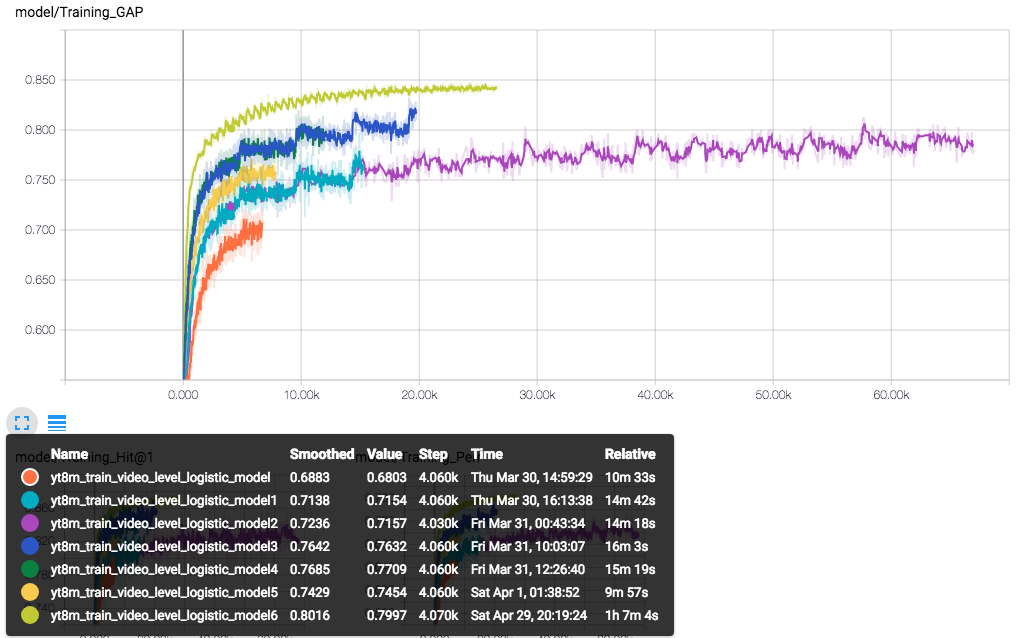
* The graphs generated during training the model using video level dataset are as follows:

(Here each step is equal to 1000 iterations)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model Used:** | **Batch Size** | **Training**  **GAP** | **Validation**  **GAP** | **Epoch** | **Learning**  **Rate** | **Decay** | **Results**  **GAP** |
| MOE model | 1024 | 0.78 | 0.723 | 14 | 0.01 | 0.95 | 0.72258 |
| Logistic | 2048 | 0.73 | 0.697 | 12 | 0.015 | 0.97 | 0.69624 |
| MOE model | 1024 | 0.80 | 0.722 | 15 | 0.012 | 0.94 | 0.7213 |
| MOE model | 1024 | 0.79 | 0.741 | 15 | 0.012 | 0.94 | 0.7416 |
| MOE model | 1024 | 0.82 | 0.781 | 12 | 0.012 | 0.96 | 0.78051 |
| MOE  model | 2048 | 0.8314 | 0.781 | 200 | 0.025 | 0.9 | 0.78143 |

Table 1:Video Level Models

* 1. Graphical representation GAP (Global Average Precision) for different models:



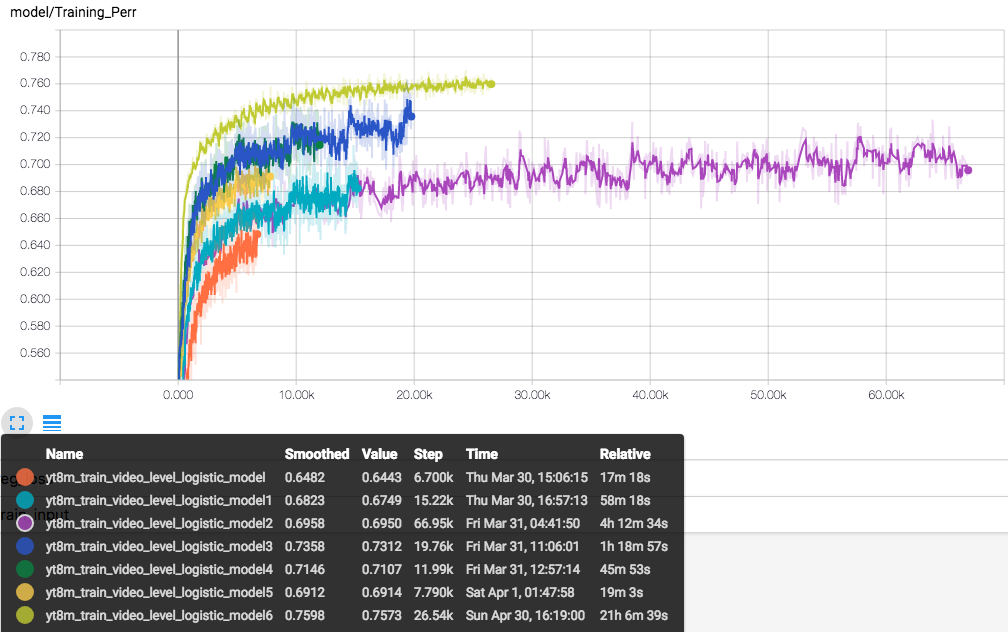
Graph 1:GAP vs Steps

2> The graph showing Hit@1 for different models using different parameters:



Graph 2:Hit@1 VS steps

3>Graph showing PERR for different models using different parameters:



Graph 3:PERR vs Steps

4>Learning rate decay graph for video level data



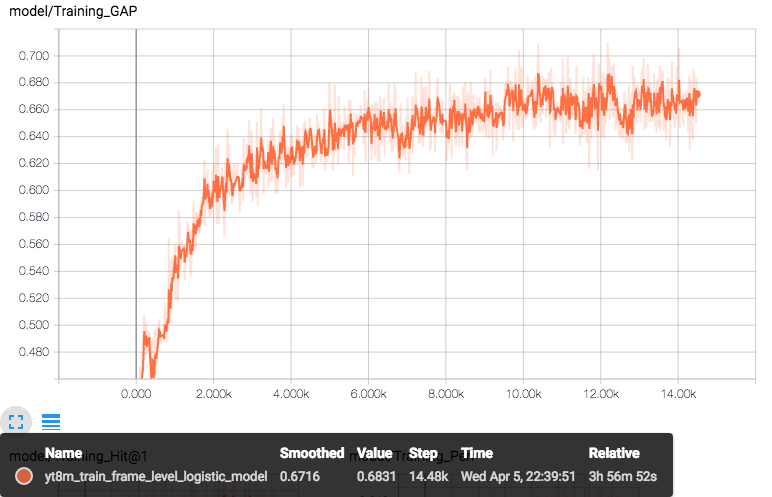
Graph 4:Learning rate VS Steps

* The graphs generated during training the model on frame level dataset are as follows:

(Due to the large volume of frame level dataset it is costly and time consuming to run the model on frame level dataset. So only one instance of such run is presented)

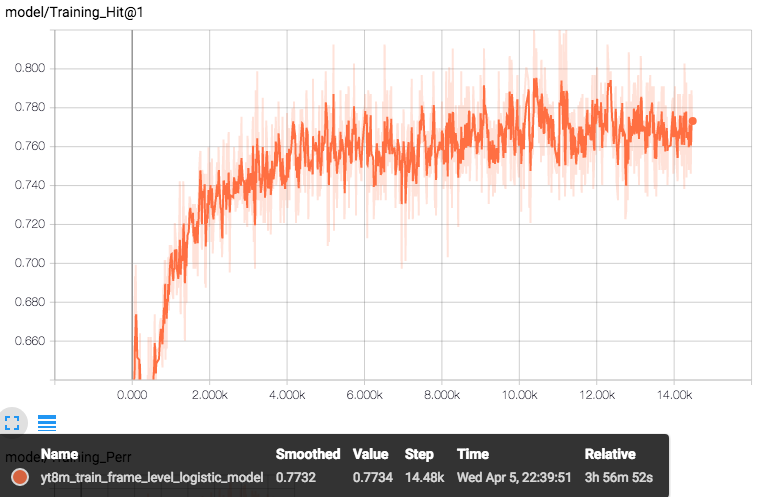
(Here each step means 1000 iterations)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model Used** | **Training**  **GAP** | **Validation**  **GAP** | **Epoch** | **Learning Rate** | **Decay** | **Results** |
| Logisitc | 1024 | 0.7114 | 0.6852 | 20 | 0.012 | 0.67143 |

1>Graphical representation of GAP on frame level datset:

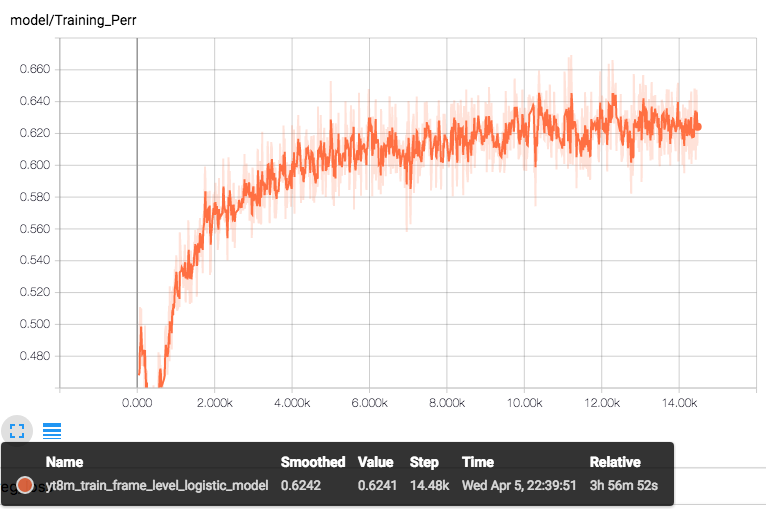
Graph 5:GAP VS Steps

2>Graphical representation of Hit@1 on frame level dataset:



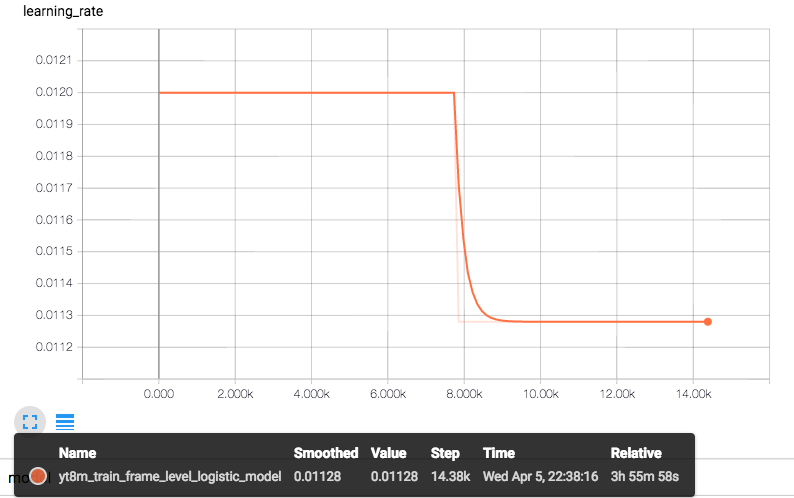
Graph 6:Hit@1 VS Steps

3>Graphical representation of PERR on frame level dataset:



Graph 7:PERR VS Steps

4>Graphical Representation of learning rate decay on frame level dataset:



Graph 8:Learning rate VS Steps

**Final Results:**

We ran the models multiple times on the testing data, each time using different parameters and type of model. The best result that we obtained used the following model and configuration:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model Used:** | **Batch Size** | **Training**  **GAP** | **Validation**  **GAP** | **Epoch** | **Learning**  **Rate** | **Decay** | **Results**  **GAP** |
| MOE  model | 2048 | 0.8314 | 0.781 | 200 | 0.025 | 0.9 | 0.78143 |

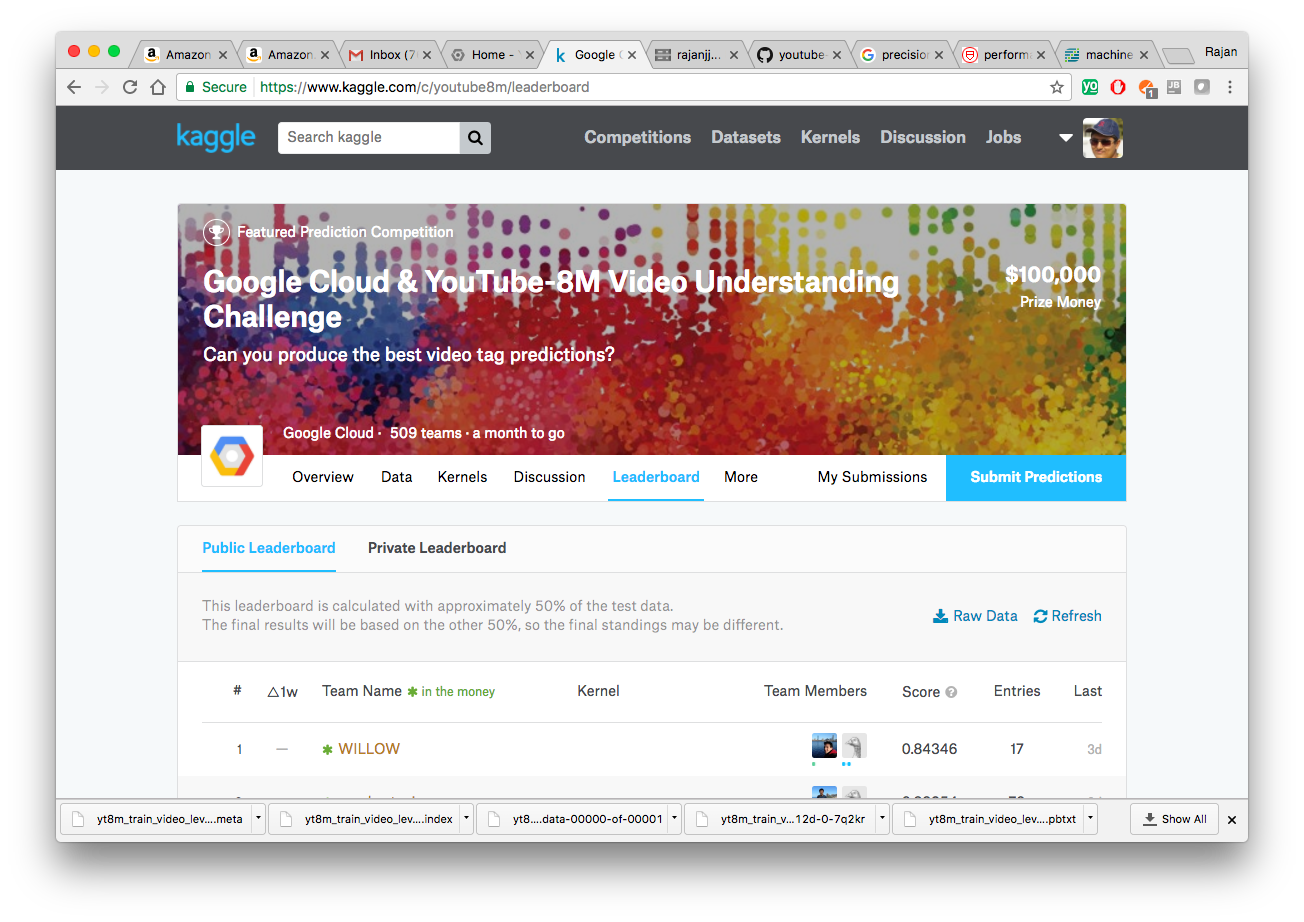
To show our progress and competition we are attaching the screenshot of leader-board of kaggle’s “Youtube 8M challenege” 

Figure 6:Total Teams:509

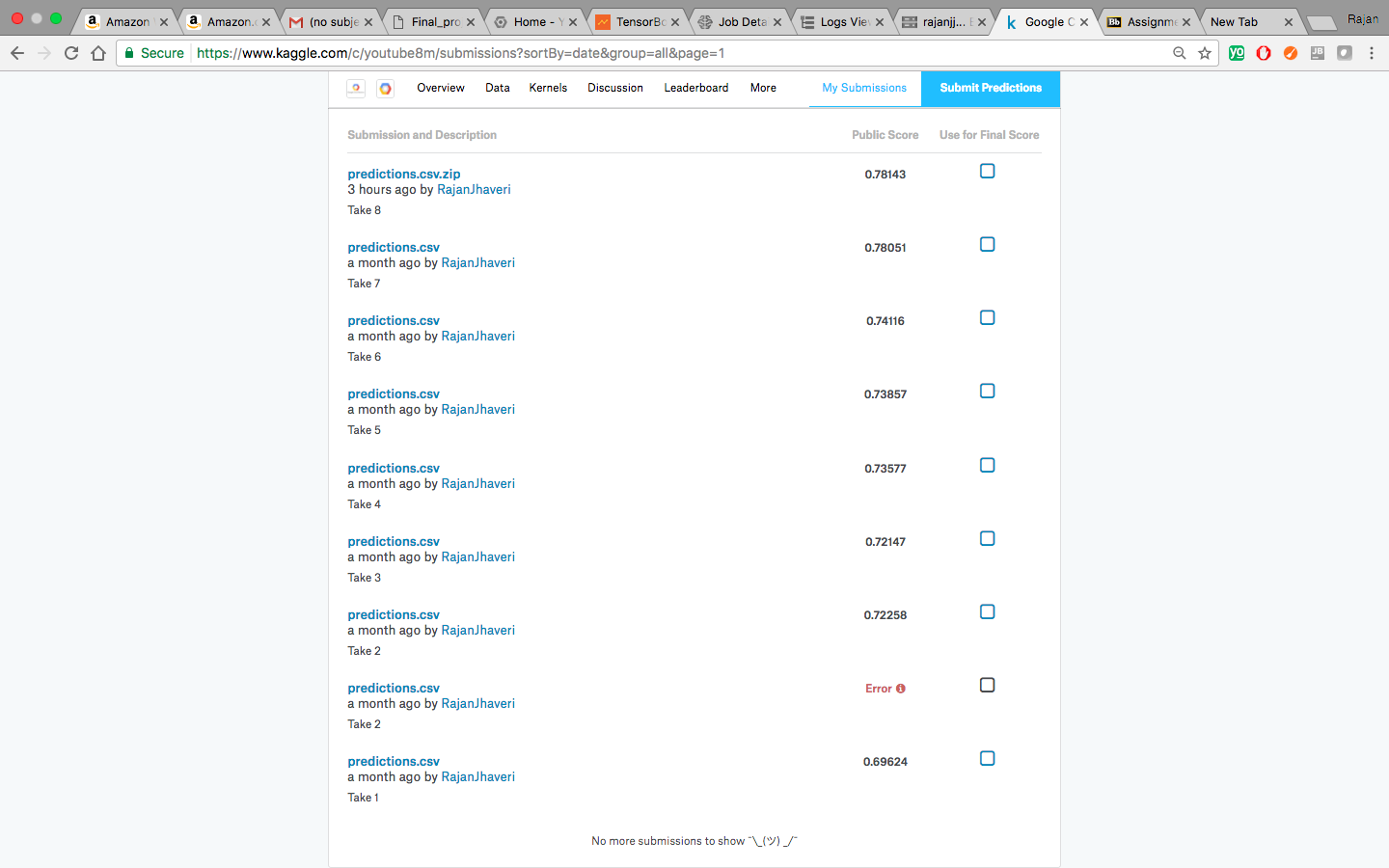


Figure 7:Total Submissions

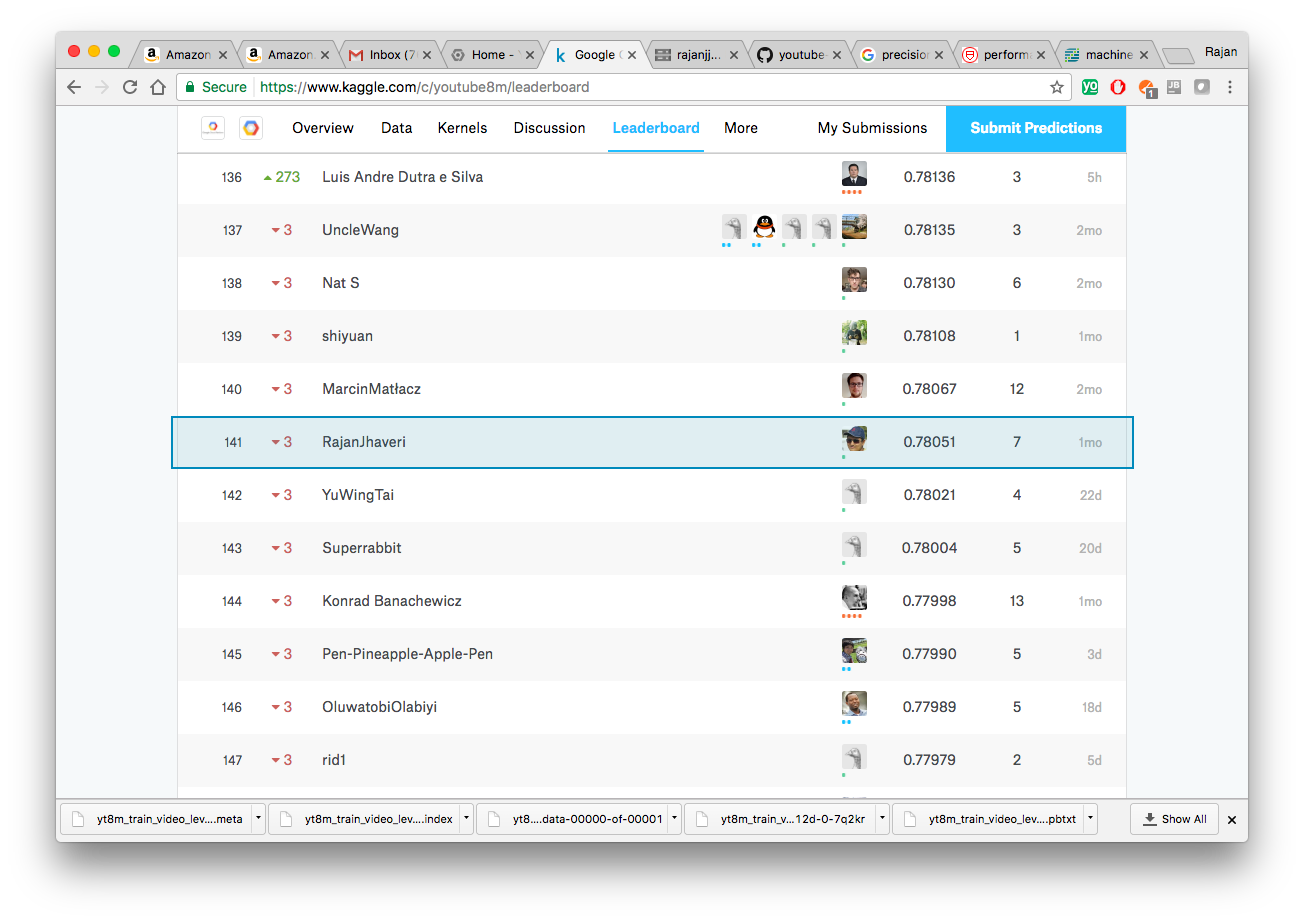


Figure 8:Intermediate result and position

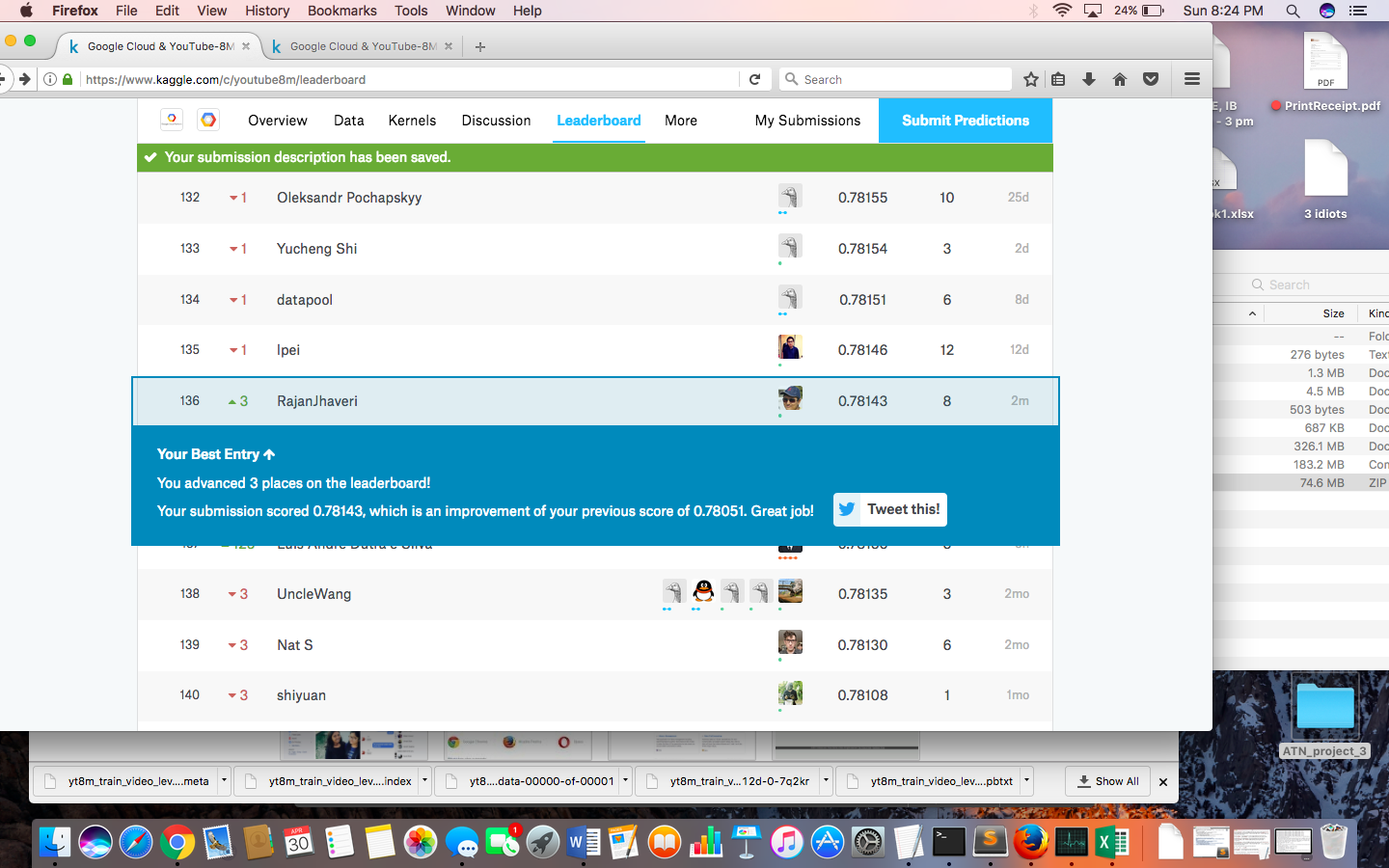


Figure 9:Final Result and Position

**Future Scope:**

As of now, we have implemented and tried many permutations of parameters to get the best results possible. We currently have a GAP value of 0.78143 and we rank 136 out of 509 teams participating in the competition. The teams participating in this competition are teams with some of the best people in machine learning. Our best results are based on Video Level data.

We tried the frame level data but could not get a very good GAP value but we understand that the best results can be achieved by utilizing the massive dataset of 1.7 TB which is a very rich dataset for understanding and better training of the model. We are still trying to implement the Recurrent Neural Net model also known as LSTM (Long Short Term Model) .

Hopefully we can achieve an accuracy much higher than what we are able to achieve as of now. From the stage we are at, even an increment of 0.05 in GAP value is considered to be a good achievement. The competition goes on till June 2nd and we hope we can end up in the top 25-50 teams which would be a great achievement.

References:

1)    Description,Data & Understanding of Challenge: <https://www.kaggle.com/c/youtube8m>

2)    Evaluation Strategy  :  <https://www.kaggle.com/c/youtube8m#evaluation>

3)    Help for Google Cloud : <https://www.kaggle.com/c/youtube8m#getting-started-with-google-cloud>

4)    All Starter Code : <https://github.com/google/youtube-8m>

5)    Learning and Using of Tensorflow : <https://www.tensorflow.org>

6)    Discussion and Know How of the subject Matter : <https://www.kaggle.com/c/youtube8m/discussion>

7)    Kernels : <https://www.kaggle.com/c/youtube8m/kernels>