Automatic Labelling of Videos using Deep Neural Network

CSE 847

Project Progress Report

Tarang Chugh
chughtar@msu.edu

Rahul Dey
deyrahul@msu.edu

1. PROBLEM DESCRIPTION

In the last decade, the world wide web has witnessed a massive explosion of multimedia data due to the rise of online media-sharing services such as Youtube, Facebook, etc. While many studies have tackled the problem of analyzing and understanding static images [1], improvements in video understanding research has been lagging due to unavailability of large-scale labeled dataset. However, analysis and understanding of video content on these video-sharing services provide insights to make the interface more engaging and customized to individual needs. Relevant search results, improved video recommendations, and content filtering are some of the many desired properties, which require the understanding of video content. In addition, law enforcement and other kinds of visual surveys can also be aided by understanding the visual feeds.

Google's recent release of the Youtube-8M dataset [2] accompanied with the launch of the Kaggle competition[1] to benchmark the performance on this task, motivated us to take up this arduous task. In this project, we explored several machine learning models that can analyze and understand video content to accurately assign labels. Google Cloud Machine Learning[2] helped us scale up the training process with a series of classifiers on a big multimedia dataset of 1.7 TB.

2. INTRODUCTION

Understanding and processing of multimedia content has been an active research area in the recent years due to their potential applications in a variety of domains. A lot of work has been done in the field of image and speech recognition and understanding. With the progress done in machine learning with images and audios, the focus of industry and academia has been shifting towards the goal of understanding and processing of videos using the methods of machine learning.

Convolutional Neural Networks (CNN) have emerged as a powerful tool for image based automation tasks such as recognition, segmentation, enhancement, encoding etc. It has become the preferred choice for such tasks as compared to other machine learning algorithms such as SVM, k-NN, Multilayer Neural Networks, etc. because of their unique ability to extract useful features from two dimensional inputs. Several standard CNN architectures such as VGG[3], AlexNet [1] and Inception [4] have been successfully applied to a large variety of vision tasks such as object-detection, segmentation, object tracking, image classification, pose estimation, etc. One important observation from such networks is that a network well trained for classification can be successfully applied for a variety of other computer vision tasks too with different datasets.

When it comes to temporal or sequential content such as audios and texts, the properties of Recurrent Neural Networks (RNNs) in learning sequential relationship in data have been proven to be very prominent and powerful in the recent years. They have been applied successfully in many speech and text based tasks such as recognition, translation, sentiment analysis, synthesis, etc. Variants of RNN such as Long Short Term Memory (LSTM) [5] and Gated Recurrent Units (GRUs) [6] have shown to be capable of capturing not only short term, but also long term temporal relationship in sequential data.

In case of videos, traditional CNN architectures suffer due to three main issues. The first issue relates to the large amount of computational resources required to train a CNN for classifying videos. For instance, videos of frame size 178×178, takes weeks to train a CNN using powerful GPUs. Performance of a deep neural network relies on the abundance of data. Thus, in case of videos, the size of data becomes even more, because of their three dimensional (two spatial and one temporal) nature. The second main issue is that CNN doesn't have any inbuilt mechanism to learn temporal patterns in the input data. And lastly, videos are usually of varying time duration. Even if all the input videos are resized to a particular resolution, different durations make them harder to be fed to a predetermined network.

Several researchers have taken different approaches to overcome some of these issues. Karpathy et al. [7] suggested an approach where the input video is divided into two separate streams of data, a *context* stream that learns low resolution features in the frames, and a *fovea* stream that learns high resolution features from the middle portion of the video. Thus, the effective resolution can be reduced substantially. This takes advantage of the camera bias since the object of interest is usually in the center of the frame. Another approach relies on treating an entire slice of video having T number of frames and feeding them into a modified convolutional layer by extending them to be of size $rows \times columns \times 3 \times T$. When successive slices are fed into the CNN, the first layer can find out the difference in them, thus characterizing the global motions in the video. Similar attempts have been made, such as in [8], to extend

[1] https://www.kaggle.com/c/youtube8m
[2] https://cloud.google.com/ml/

CNN for multimedia input by treating space and time as equivalent dimensions of the input, and performing convolutions in both time and space. Recently, Venugopalan et al. [9] modeled frames of videos using pre-trained CNN and sequence of words using a pre-trained RNN on images associated with the sentence captions to translate videos to natural language.

In this work, we work on a dataset that is formed by passing video frames through an Inception network and extracting the frame features which are then compressed using PCA.?? This allows large amounts of multimedia data to be stored in a more compressed but representative way for further processing. Thus the power of CNN can be combined with several other classifiers to train on automatic labeling of videos. The final goal of this project is to automatically recognize objects and activities in videos to improve the performance of auto-labeling, video recommendations, content filtering, retrieval, and indexing, etc.

3. PROPOSED APPROACH

3.1 Dataset

In this project, we utilize the YouTube-8M dataset containing audio-visual features of more than 8 Million multi-labeled YouTube videos (over 500,000 hours of video). These videos are annotated with 4716 class labels, representing visual entities, activities, or main theme of the videos, such as vehicle, forest, game, etc. There are an average of 3.4 class labels/video. Fig. 1 presents the most frequent top-20 class labels in the database. The length of each video is in the range of 120-300 seconds. Table 1 presents the partitions of YouTube-8M dataset into Training, Validation, and Testing splits. Figure 2 presents example videos from the YouTube-8M dataset.

The dataset is provided by YouTube in the Google Cloud as part of the YouTube-8M Video Understanding Challenge[3]. Due to the very large scale of the raw video data, the YouTube-8M dataset provides video-level and frame-level audio-visual features extracted from these videos using Convolutional Neural Networks. For frame-level features, the videos are sampled at 1 frame per second (fps), for a total of 1.8 billion frame-level features.

In order to extract the frame-level visual features, a well-known Convolutional Neural Network architecture, Inception-V3 [4], trained on ImageNet database [10], was utilized. The output from the penultimate layer (bottleneck layer) of the Inception-v3 architecture, a 2048 dimensional vector representation of the input frame, is extracted. Use of PCA and whitening, followed by quantization further reduces it to a 1024 dimensional representation for visual features. A VGG architecture based acoustic model was used to extract the 128-dimensional audio features. In order to compute video-level features, these frame-level features are averaged out.

Table 1: Dataset partitions of YouTube-8M dataset.

Dataset	Training	Validation	Testing	Total
YouTube-8M	5,786,881	1,652,167	825,602	8,264,650

3.2 Environment Setup

[3]https://research.google.com/youtube8m/download.html

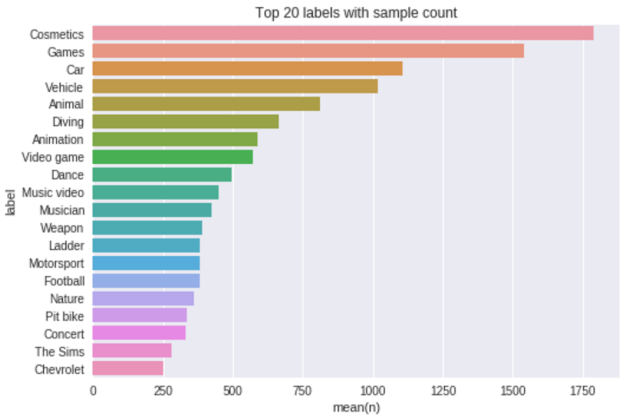


Figure 1: Top-20 frequent class labels in YouTube-8M dataset.

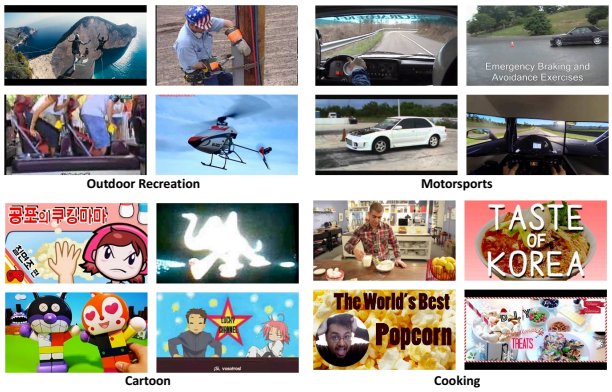


Figure 2: Example videos from YouTube-8M dataset belonging to classes (a) Outdoor Recreation, (b) Motorsports, (c) Cartoon, and (d) Cooking. Image is compiled using the YouTube-8M dataset explorer available at https://research.google.com/youtube8m/explore.html

Google provides a $300 worth credit for competition participants to use their Google Cloud ML Platform. This platform provides an environment for accessing the large training and testing files, writing code in tensorflow, submitting training/testing jobs on their GPU resources and storing and downloading of the model, log and predictions files for submission to the competition.

All our models were implemented using Tensorflow which is a python based open source library for complex numerical computations using tensors and graphs. It has been successfully applied in several machine learning applications because of the many features it offers including parallel deployment of computation to several CPUs or GPUs, data visualization using Tensorboard and so on.

3.3 Models

In view of the above described dataset, we trained several classifiers, including Deep Networks, using frame level audio-visual features. The outputs of these classifiers are confidence levels for each of the training labels. These models have been described below.

3.3.1 Frame Level Logistic Regression

We trained a simple multinomial logistic regression model on the frame level features by assigning all frame level features of a video to the same ground truth as its video. We trained 4800 independent one-vs-all classifiers for each label. The model can be simply represented by

$$y = \sigma(w^T x) \tag{1}$$

where x is a 1152 dimensional feature vector with 1024 visual features concatenated with 128 audio features in a frame and w is the weight vector to be learned from data. We used the cross-entropy loss function to optimize our model. We also used an l_2-regularization scheme to avoid over-fitting with $\lambda = 1e-8$. The base learning rate used for training was 0.01 with a batch size of 1024.

During inference, the outputs of all frame level inputs of a video are averaged to get the final labels for a video. An average pooling is preferred over max-pooling to reduce the effect of outliers associated with individual frames in a video. Thus, if $p(l_m|f_ij)$ denotes the probability of assigning a label l_m to frame f_{ij} of a video v_i, then the final probabilities of assigning the label to the video is given by,

$$p(l_m|v_i) = \frac{1}{N}\sum_{j=1}^{N} p(l_m|f_{ij}) \tag{2}$$

where N is the number of frames in the video v_i.

3.3.2 Video Level Logistic Regression

We also trained a logistic regression model on the video level audio-visual features. The model is similar to the Frame Level Logistic Regression model described above except for two things. The input in this case are the video level audio-visual features that are computed by averaging across all the frame-level features of a video. Secondly, since the average pooling along the frames is done on the input side, we do not need to do average pooling on the output side to get the final labels of a video. The output of the video-level features are taken directly as the predicted labels for the input video. Thus if x_{vi} denotes the frame-level audio-visual features for the $i-$th frame of a video v, and the total number of frames in the video is N_v, then the video level features are given by,

$$x_v = \frac{1}{N_v}\sum_{i=1}^{N_v} x_{vi} \tag{3}$$

3.3.3 Multilayer Neural Network

The logistic regression model is a linear model in the sense that the input features are linearly combined with the learned weights to predict the outputs. It fails to correctly capture non-linearities in the inputs. To overcome this situation, one could implement a multi-layer neural network. We implemented a simple two layer neural network with the following architecture in table 2. This architecture was based on the cross-validation results that we got with different architectures including three and four layer neural networks. Similar to the multinomial logistic regression, the final scores for each label is based on the average of frame level scores for the entire video. The optimum base learning rate was round to be $1e-3$ with exponential decay and a batch size of 256.

Table 2: Neural Network Architecture.

Layer	Input Size	Output Size	Activation
Input	1152	9432	ReLU
Output	9432	4716	ReLU

3.3.4 LSTM Model

Having implemented simple logistic and neural network classifiers, we went onto implement a Recurrent Neural Networks (RNN) model to capture frame-to-frame relationships in a video. As the name suggests, the output at one-time step is fed back into the network in the next time-step in a recurrent way. Thus, it tries to learn from the temporal relations in the data. Since the input dataset already consists of features trained using CNN and quantized after PCA, they represent high-level spatial information from each frame. This amounts to feeding object-level information to the RNN in order to understand the class labels associated with the videos. The idea was to be able to not only see objects in frames but also to understand various activities going on in the video. However, simple RNN models suffer from a problem called vanishing gradients which render them unable to learn long temporal relationships in the data, concentrating on only the more recent inputs. This problem has been addressed by two prominent RNN models - the Long Short Term Memory (LSTM)[5] and the Gated Recurrent Units (GRU)[6]. These models, in addition to the basic RNN unit, use a set of gates that control the flow of information from the current input and the previous output in the computation of the current output. They also typically use a 'forget' gate to control the rate of 'forgetting' past data.

The LSTM model, proposed by Hochreiter et al. is a special kind of Recurrent Neural Network architecture that is capable of modeling long-term dependencies in the inputs. This is achieved by controlling the flow of information in an LSTM cell using input gate, forget gate and output gate. Fig. 3 shows a basic LSTM cell architecture.

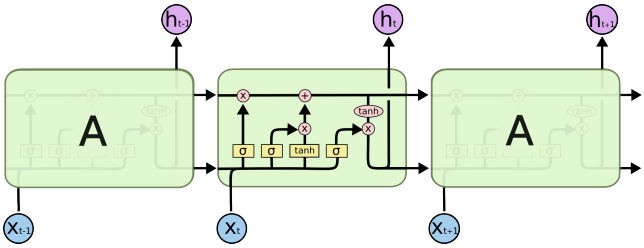


Figure 3: An LSTM Recurrent Cell. Image sourced from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

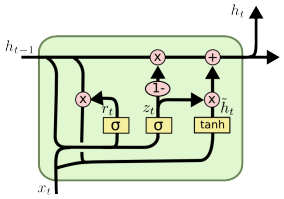
The equations governing the flow of information in an LSTM cell are given by

$$\begin{aligned} i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ \tilde{c}_t &= tanh(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t \end{aligned}$$

where i, f and o denote the input, forget and output gates respectively, $\tilde{c}$ denotes the candidate output, c denotes the out-

put, W, U and b denote the weight parameters to be learned corresponding to the input, the previous hidden state and the bias respectively with the subscript t denoting the time instance. Thus the current input, the previous hidden state and the current output are all given weightage to compute the output in a controlled manner using weight parameters that are themselves learned during the training process.

A variant of the LSTM model is the Gated Recurrent Units (GRU) which reduces the number of gates from three to two. Fig. 4 shows the architecture of a GRU cell having an update gate (similar to the output gate in LSTM) and a reset gate (similar to the forget gate in LSTM) in addition to the output activation.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 4: A GRU cell with the Controlling Equations. Image sourced from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

We experimented with several architectures of the LSTM model by varying the number of layers and cells both and found that a two layer LSTM model with 512 LSTM cells and a sequence length equal to the maximum number of frames in a video, i.e. 300 gave the highest performance. We also implemented a stacked two layer GRU model with 256 GRU cells. In our experiments, we found performance of both LSTM and GRU models to be similar. Therefore, in the rest of the article, we only report results from the LSTM model. In both cases, the base learning rate was found to be optimum at 0.01. The batch size used was 128.

3.3.5 *Deep Bag of Frames (DBoF) Model*

Deep Bag of Frames (DBOF) model, inspired by the classic bag of words approach, is utilized to learn a video-level representation from a set of frame-level features. In this model, the 1024-dimensional frame-level features are first fed into a high N-dimensional fully connected ($N > 1024$) up-projection layer with shared parameters for all frames. The Rectified Linear Unit ($max(0, x)$) activation filter applied in the fully connected layer provides a sparse representation of input frames in high N-dimensional feature space. These N-dimensions are analogous to the N discriminative clusters in bag of words approach.

The sparse representation of the frame-level features are merged to a fixed-length video-level representation using max-pooling. The final classification is performed by applying a fully-connected layer followed by a Softmax layer on the video-level representation. The optimizer used to train the model is Stochastic Gradient Descent with cross-entropy loss. Fig. 5 represents the network architecture of the DBof Model.

3.3.6 *Ensemble of Classifiers*

Finally, we also performed a score level fusion of several of these classifiers. Two ensembles were tested. The first did fusion on the outputs of the DBoF, Logistic Regression and

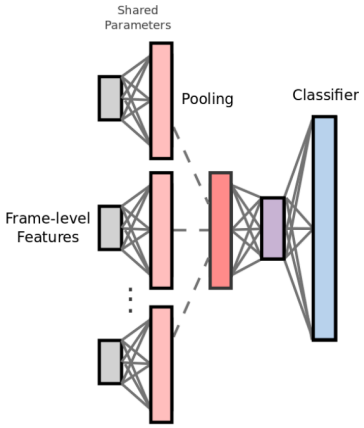


Figure 5: The architecture of Deep Bag of Frames (DBoF) model. Image taken from [2]

Neural Networks classifiers as shown in Fig. 6. The second performed fusion on the DBoF and the Logistic Regression classifiers only as shown in Fig. 7. The models for fusion were selected based on their individual performances.

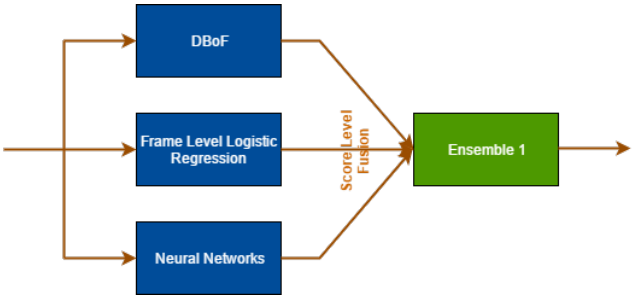


Figure 6: First ensemble classifier doing score level fusion on DBoF, Logistic Regression and Neural Networks classifier.

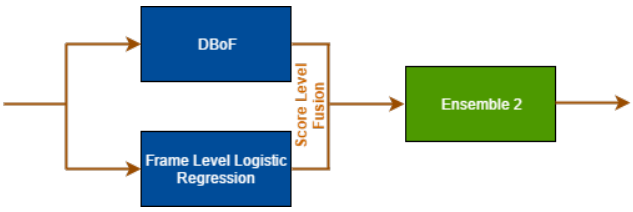


Figure 7: Second ensemble classifier doing score level fusion on DBoF and Logistic Regression classifiers.

All our models were trained using the AdamOptimizer with an exponentially decaying learning rate as shown in Fig. 8.

4. RESULTS

The models take 1-3 days each to converge on Google Cloud ML platform. Two metrics of training performance have been proposed by the Youtube-8M technical report[2].

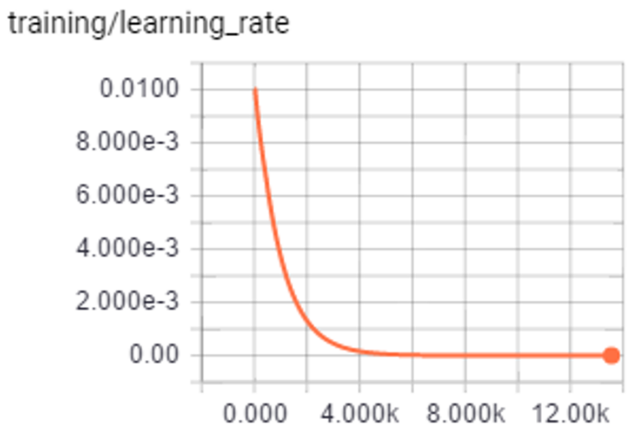


Figure 8: An adaptive learning rate was employed with exponential decay. Snapshot from Tensorboard.

4.1 Hit@k

This is the fraction of test samples that contain at least one of the ground truth labels in the top k predictions. If $rank_{v,l}$ is the rank of label l on video v (with the best scoring label having rank 1), and G_v is the set of ground-truth labels for v, then Hit@k accuracy can be written as:

$$\frac{1}{|V|}\sum_{v\in V}\bigvee_{l\in G_v} I(rank_{v,l}\leq k) \tag{4}$$

where $\bigvee$ is logical OR.

4.2 Precision at equal recall rate (PERR)

The video level annotation precision is measured when the same number of labels per video are retrieved as there are in the ground-truth. With similar notation as for Hit@k, PERR can be written as:

$$\frac{1}{|V:|G_v|>0|}\sum_{v\in V:|G_v|>0}\left[\frac{1}{|G_v|}\sum_{l\in G_v} I(rank_{v,l}\leq |G_v|)\right] \tag{5}$$

Fig. 9 and Fig. 12 show the rank1 training accuracy and training loss of different frame-level models.

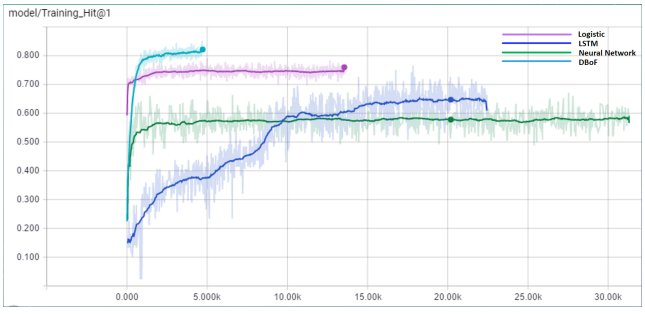


Figure 9: Rank1 Training Accuracy of Frame-Level Classifiers

For a sense of comparison, we also trained a logistic regression model on video level features. Thus the input was a 1024 dimensional feature vector computed by average the frame level visual features of all frames in a video. The rank1 training accuracy is shown in Fig. 13. The results seem to be better than those of frame level features.

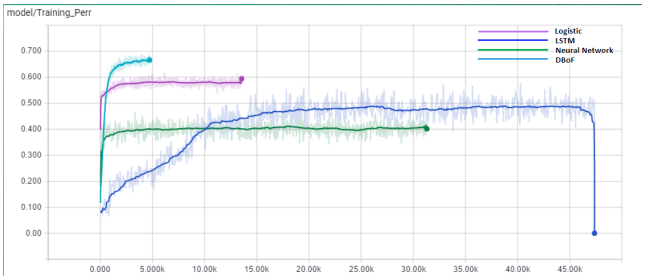


Figure 10: PERR Accuracy of Frame-Level Classifiers

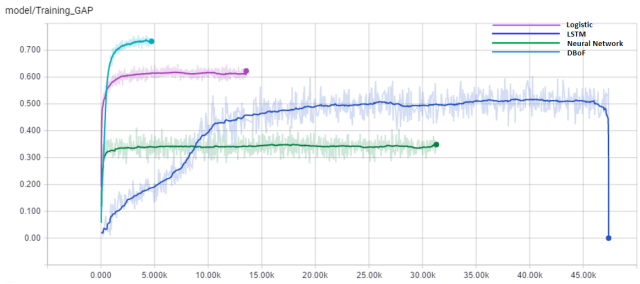


Figure 11: GAP Accuracy of Frame-Level Classifiers

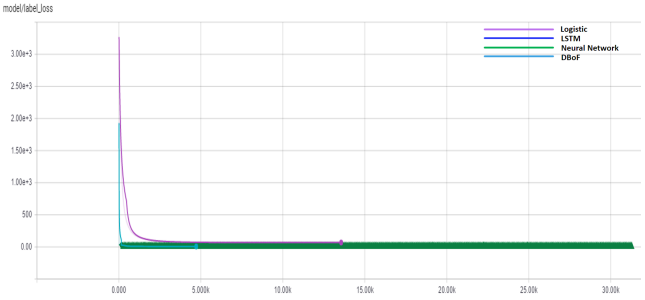


Figure 12: Training Loss Decay of Frame-Level Classifiers

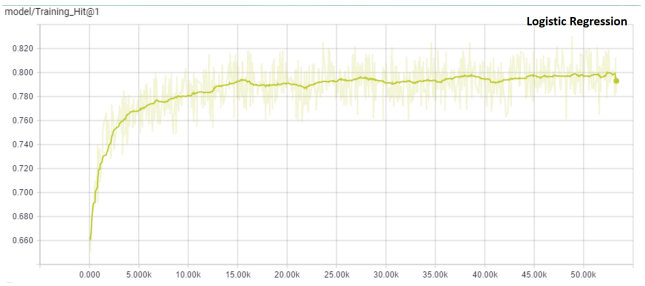


Figure 13: Rank1 Training Accuracy of Video-Level Logistic Regression Model

We submitted our inference results with top 20 predictions and their corresponding confidence score for each test video to Kaggle. These submission are evaluated based on

the Global Average Precision (GAP) at k, where $k = 20$. For N predictions of label/confidence pairs sorted by their confidence score per video, the GAP value was calculated as:

$$GAP = \sum_{i=1}^{N} p(i)\Delta r(i)$$

where N is the total number of final predictions ($N = 20 \times \#Videos$), $p(i)$ is the precision and $r(i)$ is the recall. The GAP scores for each of our frame-level models are shown in table 3.

Table 3: GAP Scores for Frame-Level and Video-Level Models.

Model	Classifier	GAP Score
1	DBoF	**72.72%**
2	Frame Level Logistic Regression	61.30%
3	LSTM	50.00%
4	Neural Networks	34.00%
5	Ensemble (1 + 2 + 4)	55.33%
6	Ensemble (1 + 2)	62.47%
7	Video Level Logistic Regression	70.30%

To our surprise, we find that the performance of a simple Logistic Regression classifier on video-level features was better than all other classifiers on frame-level features. We attribute this observation to the hypothesis that most of the input videos were short and didn't include any activity that could not be deduced by just looking at a single frame. If that is the case, a simple CNN model trained on individual frames would be able to predict labels corresponding to the frames. The video level features reflect the average of all the frames in the video, averaged along the frames. The logistic regression classifier trained on features extracted from CNN can be thought of as a final fully-connected layer at the end of a CNN architecture with the sigmoid activation function. On the other hand, a logistic regression classifier trained on frame-level features is prone to frame-based outliers. Thus, a simple logistic-regression classifier trained on video level features performs on par and sometimes better than most of the classifiers trained on frame-level features while also saving on the size of the dataset and the training time.

On the other hand, among the classifiers trained on frame-level features, the Deep Bag-of-Frames model performed better than all other classifiers. This can be attributed to two reasons. First, the model of DBoF inherently transforms frame-level features into a high-dimensional sparse representation which are then max-pooled into video level features. This amounts to both regularization and a non-linear projection of the input features resulting into better classification performance.

The LSTM model on the other hand does not perform as well as expected. This again can be attributed to two reasons. First, the LSTM model takes much longer to train and thus could not reach training accuracy saturation the way the rest of the models did. Secondly, as mentioned earlier, most of the videos are short. A human, just by looking at a few frames, can predicted most of the labels for the video. Thus, not a lot of sequential information in the video is required to predict correct labels for the video.

In Fig. 16 and table 4 we show a sample of frames from one of the test videos and the corresponding label / confidence value pairs obtained from different classifiers along with the ground truth.

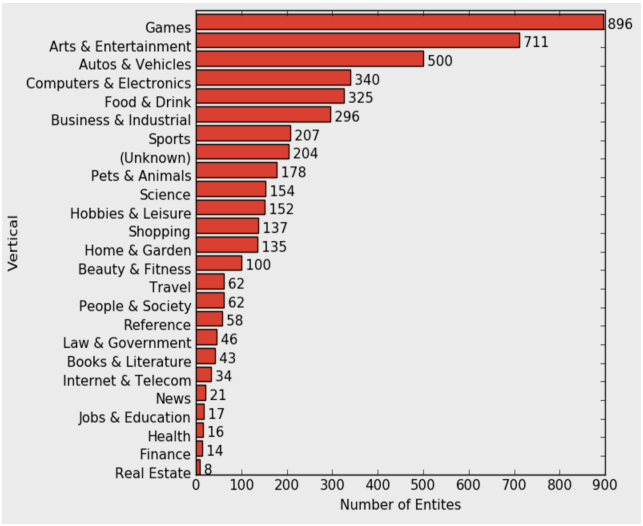


Figure 14: The 25 top-level categories and the number of class labels out of 4716 belonging to each category.

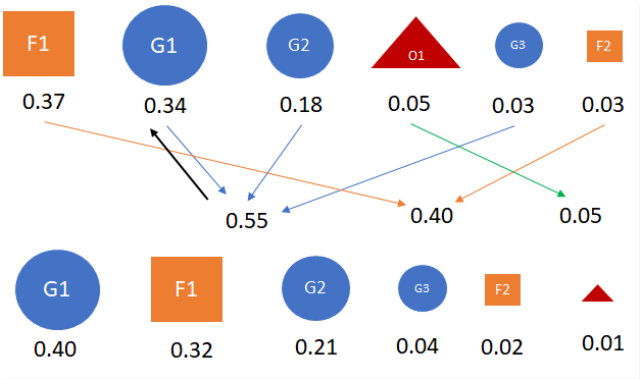


Figure 15: An example of post-processing with labels of the same class supporting each other. For example, here labels of Games (G1, G2 and G3) and Finance (F1 and F2) categories are unlikely to appear together.

4.3 Post-processing using Domain Knowledge

Each of the 4716 class labels belong to one of the top-level 25 categories $C_{i\in\{1,2,\ldots,25\}}$ as shown in Figure 14. For example, class labels car, boat, ship, etc. all belong to the same category "Autos and Vehicles". We utilize this domain knowledge to post-process the predicted class labels in order to improve the performance. We define the labels belonging to the same categories as positive for each other, and labels belonging to categories which are unlikely to appear together, such as Cooking and Vehicles, as negative for each other. Therefore, given the top-20 predictions, the weights can be post processed such that labels which are more likely to appear together are supported to further improve the score, and vice versa. In order to re-weight the

Table 4: Ground truth and the predicted labels for Sample Video 1

Ground Truth	Neural Network	LSTM	Logistic Regression	DBoF Model
Soldier	Game (0.72)	Game (0.95)	Outdoor Recreation (0.31)	Outdoor Recreation (0.79)
Weapon	Weapon (0.42)	Video Game (0.89)	Hunting (0.26)	Soldier (0.49)
Outdoor Recreation	Vehicle (0.35)	Weapon (0.68)	Weapon (0.26)	Weapon (0.43)
-	Video Game (0.31)	Call of Duty (0.47)	Soldier (0.22)	hunting (0.32)
-	Call of Duty (0.20)	PC Game (0.34)	Nature (0.15)	Forest (0.20)

Table 5: Ground truth and the predicted labels for Sample Video 2

Ground Truth	Neural Network	LSTM	Logistic Regression	DBoF Model
Dog	Animal (0.91)	Animal (0.89)	Animal (0.98)	Animal (0.99)
Animal	Animation (0.37)	Pet (0.80)	Pet (0.37)	Pet (0.72)
Shepherd	Vehicle (0.01)	Dog (0.76)	Dog (0.37)	Dog (0.60)
Pet	Nature (0.03)	Puppy (0.31)	Horse (0.36)	Cattle (0.36)
Australian Shepherd	Music Video (0.01)	Nature (0.29)	Livestock (0.12)	Collie (0.19)

class label scores, we first compute the clusters of class labels belonging to each category. The total score S_i for each category C_i is the sum of all class label scores belonging to that category. For a given category C_i, the ratio $\frac{C_i}{\sum_{j=\{1,2,\ldots,25\}} C_j}$ is defined as the support factor for all the class labels belonging to category C_i, and is multiplied with each of the class labels belonging to it. The categories which overall are having less support factor will impact any outlier present in the predictions. Conversely, if many labels belonging to the same category appear together but have lower score, they will support each other and increase the chances of providing a hit at a higher rank by improving the scores of labels that are likely to appear together. Fig. 15 shows this concept of post-processing with an example.

4.4 Challenges Faced and Learnings

In this project, we faced many challenges and also learnt a lot about working on a project with a very large scale database and intensive computational requirements. The first and most significant challenge we faced was the dataset. On one hand the dataset was of tremendous scale, which we could not download in our personal computers. We were restricted to utilize the Google Cloud Platform for accessing the database. The amount of resourced allocated to us on Google Cloud were restricted by the credit amount given to us. We had to judiciously utilize those credits for understanding the dataset, as well for designing the models. On the other hand, the dataset itself did not have any raw pixel information or any motion information across various frames. The pre-extracted features, although highly representative of the frames, did not provide enough information we would need to ourselves apply different techniques learnt in the course.

Second challenge we faced was the long waiting times for training as well as evaluating the models. Given the large scale of dataset, and rationed computation resources, one of the challenges was to plan an optimum schedule to train new models, evaluate the trained models, as well as integrate our ideas of fine-tune the models. Also, while optimizing some of the models, we were faced with issues regarding computational requirement. It was suggested that the LSTM with 1024 cells would work the best, but we could not train a model with 1024 cells due to out-of-memory errors.

While we started as newbies to deep learning, we believe we have now acquired some skills both theoretical and practical, that we can apply in various research projects.

5. CONCLUSIONS AND FUTURE WORK

We trained several models on both frame-level and video-level features. We found the the DBoF and logistic regression models give best predictions with frame-level features while similar performance is achieved with the logistic regression classifier trained on video level features. The performance of RNN models needs to be investigated further to assess if its performance can be improved by training longer and tuning the parameters. We can also try to fine tune the LSTM architecture with cross-validation.

The given features, obtained from the Inception[4] CNN architecture followed by PCA are good representatives of the image features, given the satisfiable performance of several models for video labeling.

We also propose to post-process the outputs to see if the performance can be improved using domain information as suggested earlier.

Lastly, the ensemble model did not perform as expected. This can be attributed to the fact that the individual classifiers were all trained on the same features and same training samples. Thus, there was little variation in the individual classifiers on the ensemble. In future, we can train different classifiers on different sets of features / training samples (bagging / bootstrapping) to see if the performance using an ensemble of them can be improved. Such an ensemble can also include classifiers such as SVM, Decision Trees and so on.

6. WORK DISTRIBUTION

This project was done by Rahul Dey and Tarang Chugh with the primary motive of learning to apply various deep learning and other machine learning models to the exciting task of video understanding. We spent equal efforts in first, getting started with deep learning models such as CNN and RNN, learning to use the Tensorflow library and getting acquainted with Google Cloud ML platform. Later, Tarang focussed more on the data reading, formating and pre-processing from the Tensorflow *tfrecords* data files and designing the DBoF model while Rahul worked on designing other models such as Logistic Regression, Neural Networks,

Figure 16: Frames of the sample video 1 from the validation set are presented. The ground truth labels annotated for this video are: Soldier, Weapon, and Outdoor Recreation.

Figure 17: Frames of the sample video 2 from the validation set are presented. The ground truth labels annotated for this video are: Dog, Snimal, Shepherd, Pet, and Australian Shepherd.

LSTM and GRU. Both of us contributed equally in the training, evaluation and generating inference from all these classifiers, followed by fine tuning the hyper-parameters which was usually done on the basis of the evaluation results.

Several challenges were faced while trying to train and evaluate our models on the Google Cloud ML platform including overcoming GPU Out-of-Memory challenges, utilizing multiple GPU devices and so on while were mostly troubleshooted by Rahul.Tarang also worked on implementing an SVM model on the frame level features. However, a full-fledged SVM model couldn't be trained on the entire dataset. This resulted in poor accuracy of the SVM classifier and hence was not reported in our results.

7. REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[2] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[7] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[8] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[9] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, "Translating videos to natural language using deep recurrent neural networks," *arXiv preprint arXiv:1412.4729*, 2014.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, IEEE, 2009.