Building a Size Constrained Predictive Model for Video Classification

Miha Skalic[1]* and David Austin[2]*

[1]University Pompeu Fabra, Barcelona, Spain
`miha.skalic@upf.edu`
[2]Intel Corporation, Chandler, Arizona

Abstract. Herein we present the solution to the 2nd YouTube-8M video understanding challenge which placed 1st. Competition participants were tasked with building a size constrained video labeling model with a model size of less than 1GB. Our final solution consists of several submodels belonging to Fisher vectors, NetVlad, Deep Bag of Frames and Recurrent neural networks model families. To make the classifier efficient under size constraints we introduced model distillation, partial weights quantization and training with exponential moving average.

Keywords: Deep learning, Multi-label classification, Video processing

1 Introduction

Accelerated by the increase of internet bandwidth, storage space and usage of mobile devices, generation and consumption of video data is on the rise. Paired with recent advances in deep learning [1], specifically image [2] and audio [3] processing the combination opens up new opportunities for better understanding of video content which can then be leveraged for online advertising, video retrieval, video surveillance, etc.

However, usage of deep learning for video processing can be computationally expensive if the model learning starts from raw video and audio [4]. To tackle these limitations YouTube-8M Dataset [5] was generated. In the dataset, samples are represented as a sequence of vectors where each vector, is an embedded representation of a frame and an audio snippet.

Similarly, best performing classifiers are usually computationally expensive meta-predictors that combine several end-to-end classifiers. Meta-predictors or also called ensembles were the best performing models in the first YouTube-8M Video Understanding Challenge[1] [6,7,8,9,10]. Since the ensembles are computationally expensive, in this second iteration of YouTube-8M challenge[2] the participants were asked to build a single Tensorflow [11] model of size less than 1GB.

*Authors contributed equally to this work

[1]`https://www.kaggle.com/c/youtube8m`

[2]`https://www.kaggle.com/c/youtube8m-2018`

1.1 Data

This work is based on the second iteration of YouTube-8M Dataset [5]. Over 6 million samples were split into 3 partitions: training, validation and test set, following approximately 70%, 20%, 10% split. The videos are labeled with 3862 unique tags with an average of 3 tags per video. The dataset does not provide raw videos, instead each frame representation consists of 1024 image and 128 audio features. The image features are extracted from the last ReLU activated layer prior to classification later of the Inception-v3 network [12]. The extracted features are reduced in size using PCA dimensionality reduction and finally quantized for reduced storage cost. Frames are sampled every second and up to 360 frames are included for each video.

1.2 Evaluation

Model predictions were evaluated based on Global Average precision (GAP) score:

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

where N is the number of final predictions, $p(i)$ is the precision, and $r(i)$ is the recall. In the evaluation $N = 20 \times (\mathit{number\ of\ videos})$ was used.

2 Related work

2.1 Model architectures

As the dataset is given as a sequence of vectors corresponding to frames, the main goal of a successful model is to efficiently aggregate the frames. To this end one can use recurrent neural networks, such as long short-term memory (LSTM) [13] or gated recurrent unit (GRU) [14]. Recurrent neural networks can capture the video in temporal fashion, however previous work [15,16] indicates that methods that capture distribution of features and not necessarily temporal ordering can have on a par performance. This group of models includes bag-of-visual-words [17,15], Vector of Locally aggregated Descriptors (VLAD) [18] or Fisher Vector (FV) [19] encoding.

One shortcoming of VLAD and FV methods is that they are not differentiable and thus cannot be trained with backpropagation as part of neural networks. NetVLAD [20] has been proposed as a alternative to VLAD that uses differentiable soft assignments of descriptors to clusters. Similarly, Miech et al. [6]. extended this idea to FV resulting in an approach named FVnet.

This work also includes Deep Bag of Frames (DBoF) network variants originally proposed in [5] and inspired by the success of various classic bag of words representations for video classification.

2.2 Previous solutions

As this works is based on second iteration of YouTube-8M dataset and challenge we drew inspirations from the first challenge. The first year winner [6] in addition to using NetVLAD and FVnet introduced context gating that allows for capturing of dependencies between features as well as capturing prior structure of the output space. The second placed solution [7] proposed a solution they called chaining to capture label interactions, used boosting and more importantly for this work introduced the concept of model distillation [21]. Other competitors [9,8] used various, but mostly recurrent neural network based, aggregation techniques. Our solution also includes in model weight averaging through exponential moving average [22] similar as has been done in [10].

3 Methods for improved performance

3.1 Model distillation

Model distillation [21] is a method for model compression. Initially a bigger teacher network or multiple networks are trained and then a smaller student network is trained to replicate labels predicted by teacher networks(s). Typically, student networks perform better if they are trained to reproduce teacher values than if they were trained on labels directly. Here we applied so called soft-label distillation, similar to [7], where we optimize for a combination of ground truth and predicted labels, minimizing the combination sum of two binary cross entropies:

$$\begin{aligned}\mathcal{L}\left(\boldsymbol{y}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}_{\boldsymbol{t}}\right) =& \lambda\frac{1}{l}\sum_{i=1}^{l} -y_i \log\left(\hat{y}_i\right) + \left(1-y_i\right)\log\left(1-\hat{y}_i\right) \\ &+(1-\lambda)\frac{1}{l}\sum_{i=1}^{l} -\hat{\boldsymbol{y}}_{t,i}\log\left(\hat{y}_i\right) + \left(1-y_{t,i}\right)\log\left(1-\hat{y}_i\right),\end{aligned} \tag{2}$$

where $\boldsymbol{y}$, $\hat{\boldsymbol{y}}_{\boldsymbol{t}}$ and $\hat{\boldsymbol{y}}$ are vectors of ground truth labels, teacher network prediction and student network predictions, respectively. Factor λ determines how much weight is put on minimizing student predicted labels divergence from ground truth labels versus teacher predicted values. In our experiments we have set $\lambda = 0.5$. Lowering λ to 0.4 or raising to 0.6 did not improve performance.

3.2 Partial weights quantization

Previously it has been shown that generally bigger models with larger clusters and hidden layers perform better. For example NetVlad and Fisher vector solutions from Miech et al. [6] were one of the best performing. However, they were individually bigger than 1GB, making them inadequate for the task at hand. On the other hand, model quantization [23] can significantly reduce the model size at the cost of reduced performance. To minimize drop of accuracy we used partial weights quantization, where only variables with more than $17,000$ elements were quantized. In practice this means that fully-connected layer weights were quantized, while batch normalization factors were not.

We used 8 bits quantization for large variables and left default `float32` values for variables with less than $17,000$ elements. At inference time the 8 bit variables were cast back to `float32` values based on stored centroid value. The 256 centroid values were assigned based on min-max uniform quantization.

4 Experiments

4.1 Training Details

Models were trained according to the structure in Fig. 1. The dataset used for training consisted of the prepared training set for the competition, plus all but 800 randomly selected `tfrecord` files from the validation set. The resulting dataset used was a training set of $4,769,202$ training samples and $232,072$ validation samples. We observed a very consistent offset of 0.002 GAP between local validation and the public leaderboard score from the test set which made our validation set large enough to avoid overfitting. Our implementation relied on the Tensorflow based starter code published by the competition sponsors for training, validation, and inference [3].

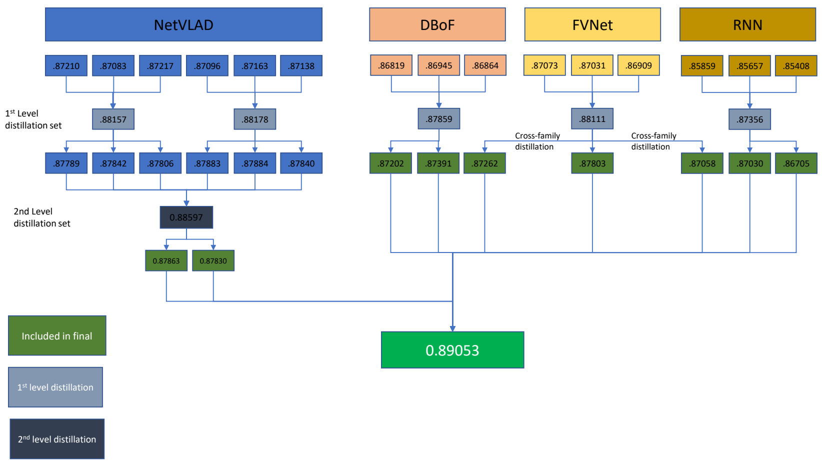

Fig. 1. Overview of our solution.

Each of the four families of models was first trained using hard targets as labels, and the difference between the models was the result of batch size, cluster,

[3] `https://github.com/google/youtube-8m`

and number of hidden layers. Each model was trained using the Adam algorithm with a batch size between $80 - 256$ frames. We did not notice a difference in training time or accuracy between single and multi-GPU training so we limited training to single GPU. For the non-RNN models we sampled 300 frames with replacement during training. Each model combination was evaluated for its ensemble score relative to other model combinations in the same family in order to select the models which would proceed to a distillation round of training. The result of the ensemble selection was a sample of models that had high, mid, and low number of clusters and hidden layers. Batch size did not have a meaningful effect on model performance.

Training of each model consisted of a two step process: first training by the method described thus far, and then secondly by performing an exponential decay average of the weights stored in checkpoint files. Averaging of weights was performed by decreasing the learning rate of a model by a factor of five, and then training the model while storing a shadow copy of the weights that was used for moving averaging.

4.2 Results

The results of averaging weights are summarized in Table 1. For DBoF and RNN architectures the gains from averaging were significant and included for each model, but for bigger models: FVNet and NetVLAD, the gains were either negative or negligible and therefore not used.

Table 1. Weights Averaging Results

Model Family	**GAP change post weight averaging**
RNN	0.00339
FVNet	-0.00186
DBoF	0.00216
NetVLAD	0.00072

After training was performed we performed partial weights quantization as detailed in section 3.2 in order to reduce model size without compromising accuracy. Table 2 shows the impact of the quantization scheme on a FVNet based architecture during validation inference of $232,073$ samples. For a negligible impact on overall GAP score we were able to reduce model size by a factor of 0.75 while incurring a 6.5 percent inference time penalty. The reason for the longer inference time with the quantized model is due to the implementation of casting quantized variables back to `float32` during inference.

Once the initial round of training for each model family was complete, the best combination of three models for a given family were selected to ensemble by equal weighting and the predictions of those models were used to generate soft targets for a distillation dataset. Our implementation resulted in a new training

Table 2. Quantization Performance for Post Distillation FVNet Model.

GAP	time (s)	Model size (MB)
0.87236	476.96	467.5
0.87237	507.79	117

dataset for each family containing the new combination of hard and soft targets as described in section 3.1.

After the distillation dataset was created for each family, we retrained the same model architectures as in the first level of training (pre-distillation). The gains as a result of distillation are summarized in Table 3. The net gain for a given architecture family ranged from 0.00409 for DBoF up to 0.01290 for RNNs. The net gain for a specific given architecture within a given family was consistent and ranged between 0.001 – 0.002.

Table 3. Distillation Performance Gain

Model	Post Distillation	No Distillation	Average Gain	Family Avg Gain
RNN1	0.87058	0.85859	0.01199	0.01290
RNN2	0.8703	0.85657	0.01373	
RNN3	0.86705	0.85408	0.01297	
FVNet1	0.87803	0.87031	0.00772	0.00772
DBoF1	0.87202	0.86819	0.00383	0.00409
DBoF2	0.87391	0.86945	0.00446	
DboF3	0.87262	0.86864	0.00398	
NetVLAD1	0.87789	0.8721	0.00579	0.00675
NetVLAD2	0.87842	0.87083	0.00759	
NetVLAD3	0.87806	0.87237	0.00569	
NetVLAD4	0.87833	0.87096	0.00737	
NetVLAD5	0.87884	0.87163	0.00721	
NetVLAD6	0.8784	0.87156	0.00684	

Because overall GAP score post distillation was highest for the NetVLAD architecture, we chose to perform a second level distillation which was accomplished by ensembling with equal weight the six models from the first level of NetVLAD distillation, and then performing another round of distillation using the same procedure as in the first round.

Results from the second round of distillation were statistically matched to the first round of distillation results, indicating that the ability of each model to learn from its teacher had been saturated. However we observed that the diversity of the second level distillation models were increased relative to the other models in the final overall ensemble and for this reason we chose to keep the second level distillation results. The smallest NetVLAD architectures that could retain the GAP score from the first level distillation were chosen.

Thirteen out of fifteen models trained using distillation were performed by using a distillation dataset created from within the same family of architecture. For example, DboFs distillation models were created as a result of combining models from three DBoF teacher models. For 2 of our 15 models, we trained a model using a distillation set trained from another family. Our rationale for training most models within the same architectural family is that it would be more likely for the student models to learn from the same architectural teachers that generated them. The two models that were trained from another family had nearly identical gains in GAP score to models trained from within the same family. We chose to keep these two models in the final ensemble for purely empirical reasons as measured by the overall ensemble scores.

For all models we made sure that all frames were sampled during inferencing. This resulted in an average GAP improvement of $0.001 - 0.0014$ GAP for most models vs using a random subset for sampling.

4.3 Ensembling

In order to achieve the best overall ensemble, we used our validation hold out set to try various combinations of weighting factors between the models. We used random search to test weighting factors, and noticed a range of 0.00050 in GAP score that could be achieved between the best combination found by random search versus taking a flat or weighted average of models.

Final model sizes and weighted contributions of each model are reported in Table 4.

Table 4. Final Ensemble Matrix

Model Family	**Model size (MB)**	**% of overall model size**	**weight fraction in final ensemble**
RNN1	54	5.13%	0.0063
RNN2	19	1.81%	0.0442
RNN3	19	1.81%	0.1244
FVNet1	369	35.08%	0.202
DBoF1	59	5.61%	0.0956
DBoF2	37	3.52%	0.1088
DBoF3	33	3.14%	0.0951
NetVLAD1	242	23.00%	0.1685
NetVLAD2	220	20.91%	0.1551

5 Conclusions

We have addressed the problem of large-scale video tagging under model size constrains. Models from four different families were used and for all them we have shown that they benefit from distillation in addition to classical ensembling and in model weights averaging. Additionally it was demonstrated that partial weight quantization is an efficient method to reduce model size down to almost a quarter the original size without significant drop in performance. Code used in this challenge, as well as full models architectures and learning parameters, are available at `http://github.com/miha-skalic/youtube8mchallange`, released under Apache License 2.0.

References

1. LeCun, Y.A., Bengio, Y., Hinton, G.E.: Deep learning. Nature (2015)
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. Advances In Neural Information Processing Systems (2012)
3. Graves, a., Mohamed, A., Hinton, G.: Speech recognition with deep recurrent neural networks. Icassp (2013)
4. Ng, J.Y.H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (2015)
5. Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., Vijayanarasimhan, S.: Youtube-8m: A large-scale video classification benchmark. CoRR **abs/1609.08675** (2016)
6. Miech, A., Laptev, I., Sivic, J.: Learnable pooling with context gating for video classification. CoRR **abs/1706.06905** (2017)
7. Wang, H., Zhang, T., Wu, J.: The monkeytyping solution to the youtube-8m video understanding challenge. CoRR **abs/1706.05150** (2017)
8. Li, F., Gan, C., Liu, X., Bian, Y., Long, X., Li, Y., Li, Z., Zhou, J., Wen, S.: Temporal modeling approaches for large-scale youtube-8m video understanding. CoRR **abs/1707.04555** (2017)
9. Chen, S., Wang, X., Tang, Y., Chen, X., Wu, Z., Jiang, Y.: Aggregating frame-level features for large-scale video classification. CoRR **abs/1707.00803** (2017)
10. Skalic, M., Pekalski, M., Pan, X.E.: Deep learning methods for efficient large scale video labeling. CoRR **abs/1706.04572** (2017)
11. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015) Software available from tensorflow.org.
12. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2016)
13. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation (1997)
14. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR **abs/1406.1078** (2014)

15. Laptev, I., Marszałek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: 26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR. (2008)
16. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: Proceedings of the IEEE International Conference on Computer Vision. (2013)
17. Wang, H., Ullah, M.M., Klaser, A., Laptev, I., Schmid, C.: Evaluation of local spatio-temporal features for action recognition. BMVC 2009 - British Machine Vision Conference (2009)
18. Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (2010)
19. Perronnin, F., Dance, C.: Fisher kernels on visual vocabularies for image categorization. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (2007)
20. Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018)
21. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop. (2015)
22. Ruppert, D.: Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering (2018)
23. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. CoRR **abs/1510.00149** (2015)