

Learnable pooling with Context Gating for video classification

Antoine Miech, Ivan Laptev and Josef Sivic
<https://github.com/antoine77340/LOUPE>

Abstract—Current methods for video analysis often extract frame-level features using pre-trained convolutional neural networks (CNNs). Such features are then aggregated over time e.g., by simple temporal averaging or more sophisticated recurrent neural networks such as long short-term memory (LSTM) or gated recurrent units (GRU). In this work we revise existing video representations and study alternative methods for temporal aggregation. We first explore clustering-based aggregation layers and propose a two-stream architecture aggregating audio and visual features. We then introduce a learnable non-linear unit, named Context Gating, aiming to model interdependencies among network activations. Our experimental results show the advantage of both improvements for the task of video classification. In particular, we evaluate our method on the large-scale multi-modal Youtube-8M v2 dataset and outperform all other methods in theYoutube 8M Large-Scale Video Understanding challenge.

Index Terms—Machine learning, Computer vision, Neural networks, Video analysis.

1 INTRODUCTION

Understanding and recognizing video content is a major challenge for numerous applications including surveillance, personal assistance, smart homes, autonomous driving, stock footage search and sports video analysis. In this work, we address the problem of multi-label video classification for user-generated videos on the Internet. The analysis of such data involves several challenges. Internet videos have a great variability in terms of content and quality (see Figure 1). Moreover, user-generated labels are typically incomplete, ambiguous and may contain errors.

Current approaches for video analysis typically represent videos by features extracted from consecutive frames, followed by feature aggregation over time. Example methods for feature extraction include deep convolutional neural networks (CNNs) pre-trained on static images [1], [2], [3], [4]. Representations of motion and appearance can be obtained from CNNs pre-trained for video frames and short video clips [5], [6], as well as hand-crafted video features [7], [8], [9]. Other more advanced models employ hierarchical spatio-temporal convolutional architectures [5], [10], [11], [12], [13], [14] to both extract and temporally aggregate video features at the same time.

Common methods for temporal feature aggregation include simple averaging or maximum pooling as well as more sophisticated pooling techniques such as VLAD [15] or more recently recurrent models (LSTM [16] and GRU [17]). These techniques, however, may be suboptimal. Indeed, simple techniques such as average or maximum pooling may become inaccurate for long sequences. Recurrent models are frequently used for temporal aggregation of variable-length sequences [18], [19] and often outperform simpler aggregation methods, however, their training remains cumbersome. As we show in Section 5, training recurrent



Groundtruth: Barbecue - Grilling - Machine - Food - Wood - Cooking
Top 6 scores: Food (97.5%) - Wood (74.9%) - Barbecue (60.0%)
- Cooking (50.1%) - Barbecue grill (27.9%) - Table (27.4%)



Groundtruth: Tree - Christmas Tree - Christmas Decoration - Christmas
Top 6 scores: Christmas (87.7%) - Christmas decoration (40.1%) - Origami
(23.0%) - Paper (15.2%) - Tree (13.9%) - Christmas Tree (7.4%)

Fig. 1: Two example videos from theYoutube-8M V2 dataset together with the ground truth and top predicted labels. Predictions colored as green are labels from the groundtruth annotation.

models requires relatively large amount of data. Moreover, recurrent models can be sub-optimal for processing of long video sequences during GPU training. It is also not clear if current models for sequential aggregation are well-adapted for video representation. Indeed, our experiments with training recurrent models using temporally-ordered and randomly-ordered video frames show similar results.

Another research direction is to exploit traditional orderless aggregation techniques based on clustering approaches such as Bag-of-visual-words [20], [21], Vector of Locally aggregated

• A. Miech, I. Laptev and J. Sivic are with Inria, WILLOW, Departement d'Informatique de l'École Normale Supérieure, PSL Research University, ENS/INRIA/CNRS UMR 8548, Paris, France
E-mail: {antoine.miech, ivan.laptev, josef.sivic}@inria.fr
• J. Sivic is also with Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague.

Descriptors (VLAD) [15] or Fisher Vectors [22]. It has been recently shown that integrating VLAD as a differentiable module in a neural network can significantly improve the aggregated representation for the task of place retrieval [23]. This has motivated us to integrate and enhance such clustering-based aggregation techniques for the task of video representation and classification.

Contributions. In this work we make the following contributions: (i) we introduce a new state-of-the-art architecture aggregating video and audio features for video classification, (ii) we introduce the Context Gating layer, an efficient non-linear unit for modeling interdependencies among network activations, and (iii) we experimentally demonstrate benefits of clustering-based aggregation techniques over LSTM and GRU approaches for the task of video classification.

Results. We evaluate our method on the large-scale multi-modal Youtube-8M V2 dataset containing about 8M videos and 4716 unique tags. We use pre-extracted visual and audio features provided with the dataset [19] and demonstrate improvements obtained with the Context Gating as well as by the combination of learnable poolings. Our method obtains top performance, out of more than 650 teams, in the Youtube-8M Large-Scale Video Understanding challenge¹. Compared to the common recurrent models, our models are faster to train and require less training data. Figure 1 illustrates some qualitative results of our method.

2 RELATED WORK

This work is related to previous methods for video feature extraction, aggregation and gating reviewed below.

2.1 Feature extraction

Successful hand-crafted representations [7], [8], [9] are based on local histograms of image and motion gradient orientations extracted along dense trajectories [9], [24]. More recent methods extract deep convolutional neural network activations computed from individual frames or blocks of frames using spatial [6], [25], [26], [27] or spatio-temporal [5], [10], [11], [12], [13], [14] convolutions. Convolutional neural networks can be also applied separately on the appearance channel and the pre-computed motion field channel resulting in the, so called, two-stream representations [6], [11], [14], [26], [28]. As our work is motivated by the Youtube-8M large-scale video understanding challenge [19], we will assume for the rest of the paper that features are provided (more details are provided in Section 5). This work mainly focuses on the temporal aggregation of given features.

2.2 Feature aggregation

Video features are typically extracted from individual frames or short video clips. The remaining question is: how to aggregate video features over the entire and potentially long video? One way to achieve this is to employ recurrent neural networks, such as long short-term memory (LSTM) [16] or gated recurrent unit (GRU) [17]), on top of the extracted frame-level features to capture the temporal structure of video into a single representation [18], [29], [30], [31], [32]. Hierarchical spatio-temporal convolution architectures [5], [10], [11], [12], [13], [14] can also be viewed

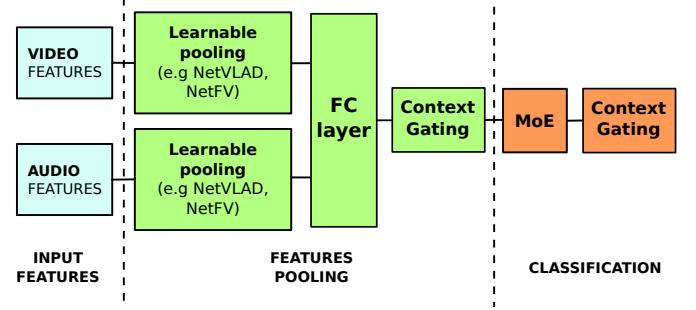


Fig. 2: Overview of our network architecture for video classification (the “Late Concat” variant). FC denotes a Fully-Connected layer. MoE denotes the Mixture-of-Experts classifier [19].

as a way to both extract and aggregate temporal features at the same time. Other methods capture only the *distribution* of features in the video, not explicitly modeling their temporal ordering. The simplest form of this approach is the average or maximum pooling of video features [33] over time. Other commonly used methods include bag-of-visual-words [20], [21], Vector of Locally aggregated Descriptors (VLAD) [15] or Fisher Vector [22] encoding. Application of these techniques to video include [7], [8], [9], [34], [35]. Typically, these methods [31], [36] rely on an unsupervised learning of the codebook. However, the codebook can also be learned in a discriminative manner [34], [37], [38] or the entire encoding module can be included within the convolutional neural network architecture and trained in the end-to-end manner [23]. This type of end-to-end trainable orderless aggregation has been recently applied to video frames in [26]. Here we extend this work by aggregating visual and audio inputs, and also investigate multiple orderless aggregations.

2.3 Gating

Gating mechanisms allow multiplicative interaction between a given input feature X and a gate vector with values in between 0 and 1. They are commonly used in recurrent neural network models such as LSTM [16] and GRU [17] but have so far not been exploited in conjunction with other non-temporal aggregation strategies such as Fisher Vectors (FV), Vector of Locally Aggregated Descriptors (VLAD) or bag-of-visual-words (BoW). Our work aims to fill this gap and designs a video classification architecture combining non-temporal aggregation with gating mechanisms. One of the motivations for this choice is the recent Gated Linear Unit (GLU) [39], which has demonstrated significant improvements in natural language processing tasks.

Our gating mechanism initially reported in [40] is also related to the parallel work on Squeeze-and-Excitation architectures [41], that has suggested gated blocks for image classification tasks and have demonstrated excellent performance on the ILSVRC 2017 image classification challenge.

3 NETWORK ARCHITECTURE

Our architecture for video classification is illustrated in Figure 2 and contains three main modules. First, the input features are extracted from video and audio signals. Next, the pooling module aggregates the extracted features into a single compact (e.g. 1024-dimensional) representation for the entire video. This

1. <https://www.kaggle.com/c/youtube8m>

pooling module has a two-stream architecture treating visual and audio features separately. The aggregated representation is then enhanced by the Context Gating layer (section 3.1). Finally, the classification module takes the resulting representation as input and outputs scores for a pre-defined set of labels. The classification module adopts the Mixture-of-Experts [42] classifier as described in [19], followed by another Context Gating layer.

3.1 Context Gating

The Context Gating (CG) module transforms the input feature representation X into a new representation Y as

$$Y = \sigma(WX + b) \circ X, \quad (1)$$

where $X \in \mathbb{R}^n$ is the input feature vector, σ is the element-wise sigmoid activation and \circ is the element-wise multiplication. $W \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$ are trainable parameters. The vector of weights $\sigma(WX + b) \in [0, 1]$ represents a set of learned gates applied to the individual dimensions of the input feature X .

The motivation behind this transformation is two-fold. First, we wish to introduce non-linear interactions among activations of the input representation. Second, we wish to recalibrate the strengths of different activations of the input representation through a self-gating mechanism. The form of the Context Gating layer is inspired by the Gated Linear Unit (GLU) introduced recently for language modeling [39] that considers a more complex class of transformations given by $\sigma(W_1X + b_1) \circ (W_2X + b_2)$, with two sets of learnable parameters W_1 , b_1 and W_2 , b_2 . Compared to the the Gated Linear Unit [39], our Context Gating in (1) (i) reduces the number of learned parameters as only one set of weights is learnt, and (ii) re-weights directly the input vector X (instead of its linear transformation) and hence is suitable for situations where X has a specific meaning, such the score of a class label, that is preserved by the layer. As shown in Figure 2, we use Context Gating in the feature pooling and classification modules. First, we use CG to transform the feature vector before passing it to the classification module. Second, we use CG after the classification layer to capture the prior structure of the output label space. Details are provided below.

3.2 Relation to residual connections

Residual connections have been introduced in [1]. They demonstrate faster and better training of deep convolutional neural networks as well as better performance for a variety of tasks. Residual connections can be formulated as

$$Y = f(WX + b) + X, \quad (2)$$

where X are the input features, (W, b) the learnable parameters of the linear mapping (or it can be a convolution), f is a non-linearity (typically Rectifier Linear Unit as expressed in [1]). One advantage of residual connections is the possibility of gradient propagation directly into X during training, avoiding the vanishing gradient problem. To show this, the gradient of the residual connection can be written as:

$$\nabla Y = \nabla(f(WX + b)) + \nabla X. \quad (3)$$

One can notice that the gradient ∇Y is the sum of the gradient of the previous layer ∇X and the gradient $\nabla(f(WX + b))$. The

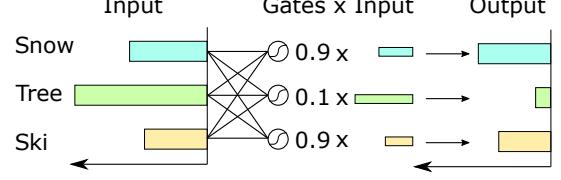


Fig. 3: Illustration of Context Gating that down-weights visual activations for Tree in a skiing scene.

vanishing gradient problem is overcome thanks to the term ∇X , which allows the gradient to backpropagate directly from Y to X without decreasing in the norm. A similar effect is observed with Context Gating which has the following gradient equation:

$$\nabla Y = \nabla(\sigma(WX + b)) \circ X + \sigma(WX + b) \circ \nabla X. \quad (4)$$

In this case, the term ∇X is weighted by activations $\sigma(WX + b)$. Hence, for dimensions where $\sigma(WX + b)$ are close to 1, gradients are directly propagated from Y to X . In contrast, for values close to 0 the gradient propagation is vanished. This property is valuable as it allows to stack several non-linear layers and avoid vanishing gradient problems.

3.3 Motivation for Context Gating

Our goal is to predict human-generated tags for a video. Such tags typically represent only a subset of objects and events which are most relevant to the context of the video. To mimic this behavior and to suppress irrelevant labels, we introduce the Context Gating module both to re-weight the features and the output labels of our architecture.

Capturing dependencies among features. Context Gating can help creating dependencies between visual activations. Take an example of a skiing video showing a skiing person, snow and trees. While network activations for the *Tree* features might be high, trees might be less important in the *context* of skiing where people are more likely to comment about the snow and skiing rather than the forest. Context Gating can learn to down-weight visual activations for *Tree* when it co-occurs with visual activations for *Ski* and *Snow* as illustrated in Figure 3.

Capturing prior structure of the output space. Context Gating can also create dependencies among output class scores when applied to the classification layer of the network. This makes it possible to learn a prior structure on the output probability space, which can be useful in modeling *biases in label annotations*.

4 LEARNABLE POOLING METHODS

Within our video classification architecture described above, we investigate several types of learnable pooling models, which we describe next. Previous successful approaches [18], [19] employed recurrent neural networks such as LSTM or GRU for the encoding of the sequential features. We chose to focus on non-recurrent aggregation techniques. This is motivated by several factors: first, recurrent models are computationally demanding for long temporal sequences as it is not possible to parallelize the sequential computation. Moreover, it is not clear if treating the aggregation problem as a sequence modeling problem is necessary. As we show in our experiments, there is almost no change in performance if we shuffle the frames in a random order as almost all of the

relevant signal relies on the static visual cues. All we actually need to do is to find a way to efficiently *remember* all of the relevant visual cues. We will first review the NetVLAD [23] aggregation module and then explain how we can exploit the same idea to imitate Fisher Vector and Bag-of-visual-Words aggregation scheme.

4.1 NetVLAD aggregation

The NetVLAD [23] architecture has been proposed for place recognition to reproduce the VLAD encoding [15], but in a differentiable manner, where the clusters are tuned via backpropagation instead of using k-means clustering. It was then extended to action recognition in video [26]. The main idea behind NetVLAD is to write the descriptor x_i hard assignment to the cluster k as a soft assignment:

$$a_k(x_i) = \frac{e^{w_k^\top x_i + b_k}}{\sum_{j=1}^K e^{w_j^\top x_i + b_j}} \quad (5)$$

where $(w_j)_j$ and $(b_j)_j$ are learnable parameters. In other words, the soft assignment $a_k(x_i)$ of descriptor x_i to cluster k measures on a scale from 0 to 1 how close the descriptor x_i is to cluster k . In the hard assignment way, $a_k(x_i)$ would be equal to 1 if x_i closest cluster is cluster k and 0 otherwise. For the rest of the paper, $a_k(x_i)$ will define soft assignment of descriptor x_i to cluster k . If we write $c_j, j \in [1, K]$ the j -th learnable cluster, the NetVLAD descriptor can be written as

$$VLAD(j, k) = \sum_{i=1}^N a_k(x_i)(x_i(j) - c_k(j)), \quad (6)$$

which computes the weighted sum of residuals $x_i - c_k$ of descriptors x_i from learnable anchor point c_k in cluster k .

4.2 Beyond NetVLAD aggregation

By exploiting the same cluster soft-assignment idea, we can also imitate similar operations as the traditional Bag-of-visual-words [20], [21] and Fisher Vectors [22] in a differentiable manner.

For bag-of-visual-words (BOW) encoding, we use soft-assignment of descriptors to visual word clusters [23], [43] to obtain a differentiable representation. The differentiable BOW representation can be written as:

$$BOW(k) = \sum_{i=1}^N a_k(x_i). \quad (7)$$

Notice that the exact bag-of-visual-words formulation is reproduced if we replace the soft assignment values by its hard assignment equivalent. This formulation is closely related to the Neural BoF formulation [44], but differs in the way of computing the soft assignment. In detail, [44] performs a softmax operation over the computed L2 distances between the descriptors and the cluster centers, whereas we use soft-assignment given by eq. (5) where parameters w are learnable without explicit relation to computing L2 distance to cluster centers. It also relates to [45] that uses a recurrent neural network to perform the aggregation. The advantage of BOW aggregation over NetVLAD is that it aggregates a list of feature descriptors into a much more compact representation, given a fixed number of clusters. The drawback is that significantly more clusters are needed to obtain a rich representation of the aggregated descriptors.

Inspired by Fisher Vector [22] encoding, we also experimented with modifying the NetVLAD architecture to allow learning of second order feature statistics within the clusters. We will denote this representation as NetFV (for Net Fisher Vectors) as it aims at imitating the standard Fisher Vector encoding [22]. Reusing the previously established soft assignment notation, we can write the NetFV representation as

$$FV1(j, k) = \sum_{i=1}^N a_k(x_i) \left(\frac{x_i(j) - c_k(j)}{\sigma_k(j)} \right), \quad (8)$$

$$FV2(j, k) = \sum_{i=1}^N a_k(x_i) \left(\left(\frac{x_i(j) - c_k(j)}{\sigma_k(j)} \right)^2 - 1 \right), \quad (9)$$

where $FV1$ is capturing the first-order statistics, $FV2$ is capturing the second-order statistics, $c_k, k \in [1, K]$ are the learnable clusters and $\sigma_k, k \in [1, K]$ are the clusters' diagonal covariances. To define $\sigma_k, k \in [1, K]$ as positive, we first randomly initialize their value with a Gaussian noise with unit mean and small variance and then take the square of the values during training so that they stays positive. In the same manner as NetVLAD, c_k and σ_k are learnt independently from the parameters of the soft-assignment a_k . This formulation differs from [38], [46] as we are not exactly reproducing the original Fisher Vectors. Indeed the parameters $a_k(x_i)$, c_k and σ_k are decoupled from each other. As opposed to [38], [46], these parameters are not related to a Gaussian Mixture Model but instead are trained in a discriminative manner.

Finally, we have also investigated a simplification of the original NetVLAD architecture that averages the actual descriptors instead of residuals, as first proposed by [47]. We call this variant NetRVLAD (for Residual-less VLAD). This simplification requires less parameters and computing operations (about half in both cases). The NetRVLAD descriptor can be written as

$$RVLAD(j, k) = \sum_{i=1}^N a_k(x_i)x_i(j). \quad (10)$$

More information about our Tensorflow [48] implementation of these different aggregation models can be found at: <https://github.com/antoine77340/LOUPE>

5 EXPERIMENTS

This section evaluates alternative architectures for video aggregation and presents results on the Youtube-8M [19] dataset.

5.1 Youtube-8M Dataset

The Youtube-8M dataset [19] is composed of approximately 8 millions videos. Because of the large scale of the dataset, visual and audio features are pre-extracted and provided with the dataset. Each video is labeled with one or multiple tags referring to the main topic of the video. Figure 5 illustrates examples of videos with their annotations. The original dataset is divided into training, validation and test subsets with 70%, 20% and 10% of videos, respectively. In this work we keep around 20K videos for the validation, the remaining samples from the original training and validation subsets are used for training. This choice was made to obtain a larger training set and to decrease the validation time. We have noticed that the performance on our validation set was comparable (0.2%-0.3% higher) to the test performance evaluated on the Kaggle platform. As we have no access to

Method	GAP
Average pooling + Logistic Regression	71.4%
Average pooling + MoE + CG	74.1%
LSTM (2 Layers)	81.7%
GRU (2 Layers)	82.0%
BoW (4096 Clusters)	81.6%
NetFV (128 Clusters)	82.2%
NetRVLAD (256 Clusters)	82.3%
NetVLAD (256 Clusters)	82.4%
Gated BoW (4096 Clusters)	82.0%
Gated NetFV (128 Clusters)	83.0%
Gated NetRVLAD (256 Clusters)	83.1%
Gated NetVLAD (256 Clusters)	83.2%

TABLE 1: Performance comparison for individual aggregation schemes. Clustering-based methods are compared with and without Context Gating.

the test labels, most results in this section are reported for our validation set. We report evaluation using the Global Average Precision (GAP) metric at top 20 as used in the Youtube-8M Kaggle competition (more details about the metric can be found at: <https://www.kaggle.com/c/youtube8m#evaluation>).

5.2 Implementation details

In the Youtube 8M competition dataset [19] video and audio features are provided for every second of the input video. The visual features consist of ReLU activations of the last fully-connected layer from a publicly available² Inception network trained on Imagenet. The audio features are extracted from a CNN architecture trained for audio classification [49]. PCA and whitening are then applied to reduce the dimension to 1024 for the visual features and 128 for the audio features. More details on feature extraction are available in [19].

All of our models are trained using the Adam algorithm [50] and mini-batches with data from around 100 videos. The learning rate is initially set to 0.0002 and is then decreased exponentially with the factor of 0.8 every 4M samples. We use gradient clipping and batch normalization [51] before each non-linear layer.

For the clustering-based pooling models, i.e. BoW, NetVLAD, NetRVLAD and NetFV, we randomly sample N features with replacement from each video. N is fixed for all videos at training and testing. As opposed to the original version of NetVLAD [23], we did not pre-train the codebook with a k-means initialization as we did not notice any improvement by doing so. For training of recurrent models, i.e. LSTM and GRU, we process features in the temporal order. We have also experimented with the random sampling of frames for LSTM and GRU which performs surprisingly similarly.

All our models are trained with the cross entropy loss. Our implementation uses the TensorFlow framework [48]. Each training is performed on a single NVIDIA TITAN X (12Gb) GPU.

5.3 Model evaluation

We evaluate the performance of individual models in Table 1. To enable a fair comparison, all pooled representations have the same size of 1024 dimensions. The “Gated” versions for the clustering-based pooling methods include CG layers as described

After pooling	After MoE	GAP
-	-	82.2%
Gated Linear Unit	-	82.4%
Context Gating	-	82.7%
Gated Linear Unit	Context Gating	82.7%
Context Gating	Context Gating	83.0%

TABLE 2: Context Gating ablation study. There is no GLU layer after MoE as GLU does not output probabilities.

Method	Early Concat	Late Concat
NetVLAD	81.9%	82.4%
NetFV	81.2%	82.2%
GRU	82.2%	82.1%
LSTM	81.7%	81.1%

TABLE 3: Evaluation of audio-video fusion methods (Early and Late Concat).

in Section 3.1. Using CG layers together with GRU and LSTM has decreased the performance in our experiments.

From Table 1 we can observe a significant increase of performance provided by all learnt aggregation schemes compared to the Average pooling baselines. Interestingly, the NetVLAD and NetFV representations based on the temporally-shuffled feature pooling outperforms the temporal models (GRU and LSTM). Finally, we can note a consistent increase in performance provided by the Context Gating for all clustering-based pooling methods.

5.4 Context Gating ablation study

Table 2 reports an ablation study evaluating the effect of Context Gating on the NetVLAD aggregation with 128 clusters. The addition of CG layers in the feature pooling and classification modules gives a significant increase in GAP. We have observed a similar behavior for NetVLAD with 256 clusters. We also experimented with replacing the Context Gating by the GLU [39] after pooling. To make the comparison fair, we added a Context Gating layer just after the MoE. Despite being less complex than GLU, we observe that CG also performs better. We note that the improvement of 0.8% provided by CG is similar to the improvement of the best non-gated model (NetVLAD) over LSTM in Table 1.

5.5 Video-Audio fusion

In addition to the late fusion of audio and video streams (Late Concat) described in Section 3, we have also experimented with a simple concatenation of original audio and video features into a single vector, followed by the pooling and classification modules in a “single stream manner” (Early Concat). Results in Table 3 illustrate the effect of the two fusion schemes for different pooling methods. The two-stream audio-visual architecture with the late fusion improves performance for the clustering-based pooling methods (NetVLAD and NetFV). On the other hand, the early fusion scheme seems to work better for GRU and LSTM aggregations. We have also experimented with replacing the concatenation fusion of audio-video features by their outer product. We found this did not work well compared to the concatenation mainly due to the high dimensionality of the resulting output. To alleviate this issue, we tried to reduce the output dimension using the multi-modal compact bilinear pooling approach [52] but found the resulting models underfitting the data.

2. https://www.tensorflow.org/tutorials/image_recognition

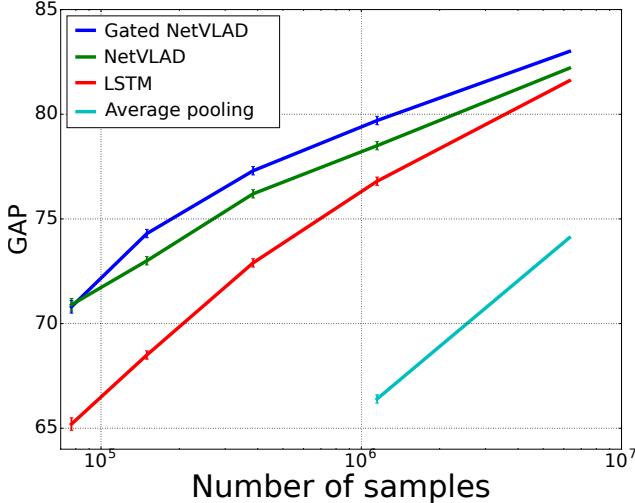


Fig. 4: The GAP performance of the different main models when varying the dataset size.

5.6 Generalization

One valuable feature of the Youtobe-8M dataset is the large scale of annotated data (almost 10 millions videos). More common annotated video datasets usually have sizes several orders of magnitude lower, ranging from 10k to 100k samples. With the large-scale dataset at hand we evaluate the influence of the amount of training data on the performance of different models. To this end, we experimented with training different models: Gated NetVLAD, NetVLAD, LSTM and average pooling based model on multiple randomly sampled subsets of the Youtobe 8M dataset. We have experimented with subsets of 70K, 150K, 380K and 1150K samples. For each subset size, we have trained models using three non-overlapping training subsets and measured the variance in performance. Figure 4 illustrates the GAP performance of each model when varying the training size. The error bars represent the variance observed when training the models on the three different training subsets. We have observed low and consistent GAP variance for different models and training sizes. Despite the LSTM model has less parameters (around 40M) compared to NetVLAD (around 160M) and Gated NetVLAD (around 180M), NetVLAD and Gated NetVLAD models demonstrate better generalization than LSTM when trained from a lower number of samples. The Context Gating module still helps generalizing better the basic NetVLAD based architecture when having sufficient number of samples (at least 100k samples). We did not show results with smaller dataset sizes as the results for all models were drastically dropping down. This is mainly due to the fact that the task is a multi-label prediction problem with a large pool of roughly 5000 labels. As these labels have a long-tail distribution, decreasing the dataset size to less than 30k samples would leave many labels with no single training example. Thus, it would not be clear if the drop of performance is due to the aggregation technique or a lack of training samples for rare classes.

5.7 Ensembling

We explore the complementarity of different models and consider their combination through ensembling. Our ensemble consists of several independently trained models. The ensembling

Approach	Ensemble size	GAP
Ours (Full)	25	85.0
Ours (Light)	7	84.7
Wang <i>et al.</i> [53]	75	84.6
Li <i>et al.</i> [54]	57	84.5
Chen <i>et al.</i> [55]	134	84.2
Skalic <i>et al.</i> [56]	75	84.2

TABLE 4: Ensemble model sizes of the top ranked teams (out of 655) from the Youtobe 8M kaggle competition.

averages label prediction scores of selected models. We have observed the increased effect of ensembling when combining diverse models. To choose models, we follow a simple greedy approach: we start with the best performing model and choose the next model by maximizing the GAP of the ensemble on the validation set. Our final ensemble used in the Youtobe 8M challenge contains 25 models. A seven models ensemble is enough to reach the first place with a GAP on the private test set of 84.688. These seven models correspond to: Gated NetVLAD (256 clusters), Gated NetFV (128 clusters), Gated BoW (4096 Clusters), BoW (8000 Clusters), Gated NetRVLAD (256 Clusters), GRU (2 layers, hidden size: 1200) and LSTM (2 layers, hidden size: 1024). Our code to reproduce this ensemble is available at: <https://github.com/antoine77340/Youtobe-8M-WILLOW>. To obtain more diverse models for the final 25 ensemble, we also added all the non-Gated models, varied the number of clusters or varied the size of the pooled representation.

Table 4 shows the ensemble size of the other top ranked approaches, out of 655 teams, from the Youtobe-8M kaggle challenge. Besides showing the best performance at the competition, we also designed a smaller set of models that ensemble more efficiently than others. Indeed, we need much less models in our ensemble than the other top performing approaches. Full ranking can be found at: <https://www.kaggle.com/c/youtube8m/leaderboard>.

6 CONCLUSIONS

We have addressed the problem of large-scale video tagging and explored trainable variants of classical pooling methods (BoW, VLAD, FV) for the temporal aggregation of audio and visual features. In this context we have observed NetVLAD, NetFV and BoW to outperform more common temporal models such as LSTM and GRU. We have also introduced the Context Gating mechanism and have shown its benefit for the trainable versions of BoW, VLAD and FV. The ensemble of our individual models has been shown to improve the performance further, enabling our method to win the Youtobe 8M Large-Scale Video Understanding challenge. Our TensorFlow toolbox LOUPE is available for download from [57] and includes implementations of the Context Gating as well as learnable pooling modules used in this work.

ACKNOWLEDGMENTS

The authors would like to thank Jean-Baptiste Alayrac and Relja Arandjelović for valuable discussions as well as the Google team for providing the Youtobe-8M Tensorflow Starter Code. This work has also been partly supported by ERC grants ACTIVIA (no. 307574) and LEAP (no. 336845), CIFAR Learning in Machines & Brains program, ESIF, OP Research, development and education Project IMPACT No. CZ.02.1.01/0.0/0.0/15_003/0000468 and a Google Research Award.



Groundtruth: Dish - Cuisine - Food - Sauce

Top 4 scores: Food (99.7%) - Cooking (78.4%) - Cuisine (69.9%)
- Dish (44.6%)



Groundtruth: Gadget - Iphone 3G - Mobile Phone - Smartphone
- Computer Monitor - Telephone

Top 6 scores: Mobile phone (99.9%) - Smartphone (99.7%) - Gadget (89.3%)
- Telephone (49.0%) - Camera (5.2%) - Microsoft Lumia (3.3%)



Groundtruth: Paper - Food - Art

Top 4 scores: Paper (61.6%) - Art (51.9%) - Hat (13.5%) - Paint (10.2%)



Groundtruth: Radio-controlled aircraft - Vehicle - North America P-51 Mustang
- Airplane - Model Aircraft - Landing - Radio-controlled model

Top 7 scores: Vehicle (100%) - Airplane (99.9%) - Radio-controlled model (99.6%) - Model aircraft (98.8%) - Aircraft (98.7%) - Radio-controlled aircraft (94.0%) - Motorsport (55.3%)



Groundtruth: Car - Vehicle - Sport Utility Vehicle - Dacia Duster - Renault
- Four Wheel Drive

Top 6 scores: Car (100%) - Vehicle (100%) - Sport Utility Vehicle (97.0%)
- Dacia Duster (30.1%) - Fiat Automobiles (30.0%) - Volkswagen Beetles (12.0%)



Groundtruth: Barbecue - Grilling - Machine - Food - Wood - Cooking

Top 4 scores: Food (97.5%) - Wood (74.9%) - Barbecue (60.0%)
- Cooking (50.1%) - Barbecue grill (27.9%) - Table (27.4%)



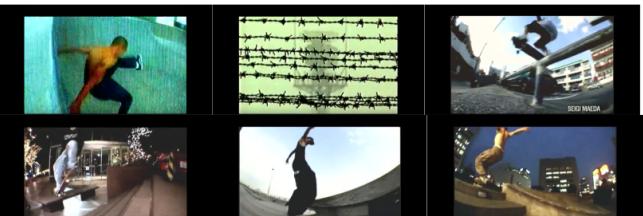
Groundtruth: Festival - Musician - Parade - Marching Band

Top 9 scores: Parade (99.7%) - Musical Ensemble (99.7%) - Marching Band (98.9%) - Musician (65.9%) - Festival (59.7%) - University (9.1%) - School (9.1%) - Military Band (8.8%) - Stadium (8.7%)



Groundtruth: Concert - Musician - Musical Ensemble - Drummer - Orchestra

Top 5 scores: Concert (99.6%) - Musician (99.6%) - Musical Ensemble (92.8%)
- Orchestra (89.0%) - Drummer (80.0%)



Groundtruth: Skateboard - Skateboarding - Outdoor recreation

Top 3 scores: Skateboarding (100%) - Skateboard (98.2%) - Outdoor recreation (97.0%)



Groundtruth: Tree - Christmas Tree - Christmas Decoration - Christmas

Top 6 scores: Christmas (87.7%) - Christmas decoration (40.1%) - Origami (23.0%) - Paper (15.2%) - Tree (13.9%) - Christmas Tree (7.4%)

Fig. 5: Qualitative results from our best single model (Gated NetVLAD). We show both groundtruth labels (in green) from the Youtube 8M dataset and the top predictions of the Gated NetVLAD model.

REFERENCES

- [1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *CVPR*, 2016.
- [2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.
- [3] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *ICLR*, 2015.
- [4] C. Szegedy, S. Ioffe, and V. Vanhoucke, “Inception-v4, inception-resnet and the impact of residual connections on learning,” *arXiv:1602.07261v1*, 2016.
- [5] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in *ICCV*, 2015.
- [6] C. Feichtenhofer, A. Pinz, and A. Zisserman, “Convolutional two-stream network fusion for video action recognition,” in *CVPR*, 2016.
- [7] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, “Learning realistic human actions from movies,” in *CVPR*, 2008.
- [8] C. Schüldt, I. Laptev, and B. Caputo, “Recognizing human actions: a local svm approach,” in *ICPR*, 2004.
- [9] H. Wang and C. Schmid, “Action Recognition with Improved Trajectories,” in *ICCV*, 2013.
- [10] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, “Sequential deep learning for human action recognition,” *Human Behavior Understanding*, pp. 29–39, 2011.
- [11] J. Carreira and A. Zisserman, “Quo vadis, action recognition? a new model and the kinetics dataset,” in *CVPR*, 2017.
- [12] C. Feichtenhofer, A. Pinz, and R. P. Wildes, “Spatiotemporal multiplier networks for video action recognition,” in *CVPR*, 2017.
- [13] S. Ji, W. Xu, M. Yang, and K. Yu, “3D Convolutional Neural Networks for Human Action Recognition,” in *PAMI*, 2013.
- [14] G. Varol, I. Laptev, and C. Schmid, “Long-term Temporal Convolutions for Action Recognition,” *PAMI*, 2017.
- [15] H. Jegou, M. Douze, C. Schmid, and P. Perez, “Aggregating local descriptors into a compact image representation,” in *CVPR*, 2010.
- [16] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” in *Neural Computing*, 1997.
- [17] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, “On the Properties of Neural Machine Translation: Encoder-Decoder Approaches,” *arXiv preprint arXiv:1409.1259*, 2014.
- [18] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” *arXiv preprint arXiv:1411.4389*, 2014.
- [19] 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.
- [20] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” in *ECCV Workshop*, 2004.
- [21] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos,” in *ICCV*, 2003.
- [22] F. Perronnin and C. Dance, “Fisher kernels on visual vocabularies for image categorization,” in *CVPR*, 2007.
- [23] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, “NetVLAD: CNN architecture for weakly supervised place recognition,” in *CVPR*, 2016.
- [24] C. R. de Souza, A. Gaidon, E. Vig, and A. M. López, “Sympathy for the details: Dense trajectories and hybrid classification architectures for action recognition,” in *ECCV*, 2016.
- [25] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *CVPR*, 2014, pp. 1725–1732.
- [26] R. Girdhar, D. Ramaman, A. Gupta, J. Sivic, and B. Russell, “Actionvlad: Learning spatio-temporal aggregation for action classification,” in *CVPR*, 2017.
- [27] L. Wang, Y. Qiao, and X. Tang, “Action recognition with trajectory-pooled deep-convolutional descriptors,” in *CVPR*, 2015, pp. 4305–4314.
- [28] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *ICLR*, 2014, pp. 568–576.
- [29] F. Basura, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars, “Modeling video evolution for action recognition,” in *CVPR*, 2015.
- [30] M. Ibrahim, S. Muralidharan, Z. Deng, A. Vahdat, and M. Greg, “A Hierarchical Deep Temporal Model for Group Activity Recognition,” in *CVPR*, 2016.
- [31] G. Lev, G. Sadeh, B. Klein, and L. Wolf, “Rnn fisher vectors for action recognition and image annotation,” in *ECCV*, 2016.
- [32] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *CVPR*, 2015.
- [33] L. Wang, Y. Xiong, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, “Temporal segment networks: Towards good practices for deep action recognition,” in *ECCV*, 2016.
- [34] X. Peng, L. Wang, Y. Qiao, and Q. Peng, “Boosting VLAD with Supervised Dictionary Learning and High-Order Statistics,” in *ECCV*, 2014.
- [35] Z. Xu, Y. Yang, and A. G. Hauptmann, “A Discriminative CNN Video Representation for Event Detection,” in *CVPR*, 2015.
- [36] F. Perronnin and D. Larlus, “Fisher Vectors Meet Neural Networks: A Hybrid Classification Architecture,” in *CVPR*, 2015.
- [37] X. Peng, C. Zou, Y. Qiao, and Q. Peng, “Action recognition with stacked fisher vectors,” in *ECCV*, 2014.
- [38] V. Sydorov, M. Sakurada, and C. H. Lampert, “Deep fisher kernels and end to end learning of the Fisher kernel GMM parameters,” in *CVPR*, 2014.
- [39] Y. N. Dauphin, F. Angela, M. Auli, and D. Grangier, “Language modeling with gated convolutional networks,” in *arXiv preprint arXiv:1612.08083*, 2016.
- [40] A. Miech, I. Laptev, and J. Sivic, “Learnable pooling with context gating for video classification,” *arXiv preprint arXiv:1706.06905*, 2017.
- [41] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” *arXiv preprint arXiv:1709.01507*, 2017.
- [42] M. I. Jordan, “Hierarchical mixtures of experts and the em algorithm,” *Neural Computation*, 1994.
- [43] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Lost in quantization: Improving particular object retrieval in large scale image databases,” in *CVPR*, 2008.
- [44] N. Passalis and A. Tefas, “Learning neural bag-of-features for large scale image retrieval,” *IEEE Trans. Cybernetics*, 2017.
- [45] A. Richard and J. Gall, “A bag-of-words equivalent recurrent neural network for action recognition,” in *BMVC*, 2015.
- [46] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep fisher networks for large-scale image classification,” in *NIPS*, 2013.
- [47] M. Douze, J. Revaud, C. Schmid, and H. Jegou, “Stable hyper-pooling and query expansion for event detection,” in *ICCV*, 2013.
- [48] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” *arXiv preprint arXiv:1603.04467*, 2015.
- [49] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, “CNN architectures for large-scale audio classification,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.
- [50] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.
- [51] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.
- [52] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, “Compact bilinear pooling,” in *CVPR*, 2016.
- [53] H.-D. Wang, T. Zhang, and J. Wu, “The monkeytyping solution to the youtube-8m video understanding challenge,” *arXiv preprint arXiv:1706.05150*, 2017.
- [54] F. Li, C. Gan, X. Liu, Y. Bian, X. Long, Y. Li, Z. Li, J. Zhou, and S. Wen, “Temporal modeling approaches for large-scale youtube-8m video understanding,” *arXiv preprint arXiv:1707.04555*, 2017.
- [55] S. Chen, X. Wang, Y. Tang, X. Chen, Z. Wu, and Y.-G. Jiang, “Aggregating frame-level features for large-scale video classification,” *arXiv preprint arXiv:1707.00803*, 2017.
- [56] M. Skalic, M. Pekalski, and X. E. Pan, “Deep learning methods for efficient large scale video labeling,” *arXiv preprint arXiv:1706.04572*, 2017.
- [57] A. Miech, “LOUPE tensorflow toolbox for learnable pooling module,” <https://github.com/antoine77340/LOUPE>, 2017.