NeXtVLAD: An Efficient Neural Network to Aggregate Frame-level Features for Large-scale Video Classification

Rongcheng Lin, Jing Xiao, Jianping Fan

University of North Carolina at Charlotte

Abstract. This paper introduces a fast and efficient network architecture, NeXtVLAD, to aggregate frame-level features into a compact feature vector for video classification. Briefly speaking, the basic idea is to decompose a high-dimensional feature into a group of relatively low-dimensional vectors with attention before applying NetVLAD[1] aggregation over time. The approach turns out to be both effective and parameter efficient in aggregating temporal information. In the 2nd Youtube-8M video understanding challenge, a single NeXtVLAD model with less than 80M parameters achieves a GAP score of 0.87846 in private leaderboard. A mixture of 3 NeXtVLAD models results in 0.88722, which is ranked 5th over 394 teams. The code is publicly available at https://github.com/linrongc/youtube-8m

Keywords: Neural Network, VLAD, Video Classification, Youtube-8M

1 Introduction

Due to the prevalence of digital cameras and smart phones, exponentially increasing number of videos are recorded, uploaded, watched and shared. To automatically recognize and understand the video content has become a critical and challenging problem in many real world applications, including video-based search, recommendation and intelligent robots etc. To accelerate the pace of research in video content analysis, Google AI launched the second Youtube-8M video understanding challenge, aiming to learn video representation under limited budget constraints. Because of the unprecedent scale and diversity of the Youtube-8M dataset[2], frame-level visual and audio features extracted by pre-trained convolutional neural networks (CNNs) are provided. How to effectively and efficiently aggregate such features into a compact video-level representation is the main challenge.

NetVLAD, which was originally used to aggregate spatial representation for task of place recognition[1], was found to be more effective and faster than common temporal models, such as LSTM[3] and GRU[4], for the task of temporal aggregation of visual and audio features[5]. One of the main drawbacks of NetVLAD is that the encoded features are of high dimension. A non-trivial classification model based on those features would need hundreds of millions of parameters. For instance, a NetVLAD network with 128 clusters will encode a feature of 2048 dimension as an vector of 262144 dimension. A subsequent fully-connected layer with 2048-dimensional outputs will result in about 537M parameters. The parameter inefficiency would make the model harder to be optimized and easier to be overfitting.

To handle the parameter inefficiency problem, inspired by the work of ResNeXt[6], we proposed a novel neural network architecture, NeXtVLAD. Different from NetVLAD, the input features are decomposed into a group of relatively lower-dimensional vectors with attention before they are encoded and aggregated over time. The underlying assumption is that one video frame may contains multiple objects and decomposing the frame-level features before encoding would be beneficial for models to produce a more concise video representation. Experimental results on Youtube-8M dataset shows that our proposed model are both more effective and parameter efficient than the original NetVLAD model. Moreover, the NeXtVLAD model can converge faster and more resistant to overfitting.

2 Related Works

2.1 Feature Aggregation

Before the era of deep neural networks, researchers proposed many encoding methods, including BoW (Bag of visual Words)[7], FV (Fisher Vector)[8] and VLAD (Vector of Locally Aggregated Descriptors)[9] etc., to aggregate local image descriptors into a global compact vector, aiming to improve the performance of large scale visual object retrieval. The aggregation methods are also applied to the video classification problem in some early works[10][11]. Recently, [1] proposed a differential module, NetVLAD, to integrate VLAD into current neural networks and showed significant improvement for the task of place retrieval. The architecture was then proved to very effective in aggregating spatial and temporal information in videos[5][12].

2.2 Video Classification with Deep Neural Networks

Recently, with the availability of large-scale video data sets[13][14][2] and mass computation power of GPUs, deep neural networks have achieve remarkable advances in the field of video classification[15][16][17][18]. These approaches can be roughly summarized into following categories:

1). Spatiotemporal Convolutional Networks[13][17][18], which mainly relying on convolution and pooling to aggregate temporal information along with spatial information.
2). Two Stream Networks[16][19][20][21], which utilize stacked optical flow to recognize human motions in addition to the context frame images.
3). Recurrent Spatial Networks[15][22], which applied Recurrent Neural Networks, including LSTM or GRU to model temporal information in videos.
4). Other approaches[23][24][25][26].

3 Network Architecture

We will first review the NetVLAD aggregation model before we dive into the details of our proposed NeXtVLAD model for video classification.

3.1 NetVLAD Aggregation Network for Video Classification

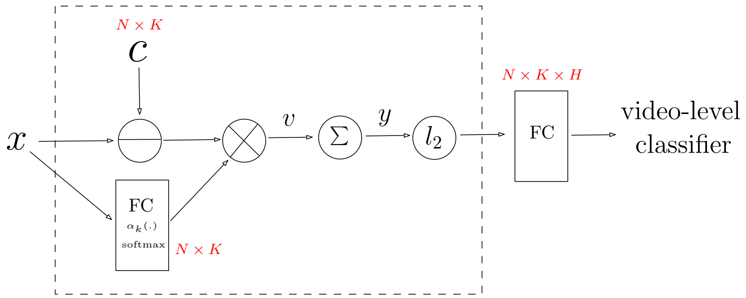

Fig. 1. Schema of NetVLAD model for video classification. Formulas in red denote the number of parameters (ignoring biases or batch normalization). FC means fully-connected layer.

Considering a video with M frames, N-dimensional frame-level descriptors x are extracted by a pre-trained spatial CNN. In NetVLAD aggregation of K clusters, each frame-level descriptor is firstly encoded to be a feature vector of $N \times K$ dimension using the following equation:

$$\begin{gathered} v_{ijk} = \alpha_k(x_i)(x_{ij} - c_{kj}) \\ i \in \{1, ..., M\}, j \in \{1, ..., N\}, k \in \{1, ..., K\} \end{gathered} \tag{1}$$

where c_k is the N-dimensional anchor point of cluster k and $\alpha_k(x_i)$ is a soft assignment function of x_i to cluster k, which measures the proximity of x_i and cluster k. The proximity function is modeled using a single fully-connected layer with softmax activation,

$$\alpha_k(x_i) = \frac{e^{w_k^T x_i + b_k}}{\sum_{s=1}^{K} e^{w_s^T x_i + s_t}}. \tag{2}$$

Secondly, a video-level descriptor y can be obtained by aggregating all the frame-level features,

$$y_{jk} = \sum_{i}^{M} v_{ijk} \tag{3}$$

and intra-normalization is applied to suppress bursts[27]. Finally, the constructed video-level descriptor y is reduced to an H-dimensional hidden vector via a fully-connected layer before being fed into the final video-level classifier.

As shown in Figure 1, the parameter number of NetVLAD model before video-level classification is about

$$N \times K \times (H + 2), \tag{4}$$

where the dimension reduction layer (second fully-connected layer) accounts for the majority of total parameters. For instance, a NetVLAD model with $N = 1024$, $K = 128$ and $H = 2048$ contains more than $268M$ parameters.

3.2 NeXtVLAD Aggregation Network

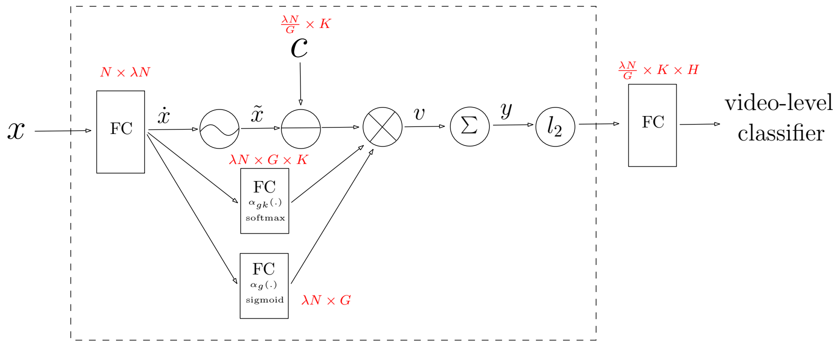

Fig. 2. Schema of a NeXtVLAD network for video classification. Formulas in red denote the number of parameters (ignoring biases or batch normalization). FC represents a fully-connected layer. The wave operation means a reshape transformation.

In a NeXtVLAD aggregation network, the input vector x_i is firstly expanded as $\dot{x}_i$ with a dimension of λN via a linear fully-connected layer, where λ is a width multiplier we set to be 2 in all of our experiments. Then a reshape operation is applied to transform $\dot{x}$ with a shape of $(M, \lambda N)$ to $\tilde{x}$ with a shape of $(M, G, \lambda N/G)$, in which G is the size of groups. The process is equivalent to splitting $\dot{x}_i$ into G lower-dimensional feature vectors $\left\{ \tilde{x}_i^g \middle| g \in \{1, ..., G\} \right\}$, each of which is subsequently represented as a mixture of residuals from cluster anchor points c_k in the same lower-dimensional space:

$$\begin{aligned} v_{ijk}^g &= \alpha_g(\dot{x}_i)\alpha_{gk}(\dot{x}_i)(\tilde{x}_{ij}^g - c_{kj}) \\ g \in \{1, ..., G\}, i &\in \{1, ..., M\}, j \in \{1, ..., \frac{\lambda N}{G}\}, k \in \{1, ..., K\}, \end{aligned} \tag{5}$$

where the proximity measurement of the decomposed vector $\tilde{x}_i^g$ to cluster k consists of two parts:

$$\alpha_{gk}(\dot{x}_i) = \frac{e^{w_{gk}^T \dot{x}_i + b_{gk}}}{\sum_{s=1}^K e^{w_{gs}^T \dot{x}_i + b_{gs}}}, \tag{6}$$

$$\alpha_g(\dot{x}_i) = \sigma(w_g^T \dot{x}_i + b_g), \tag{7}$$

in which $\sigma(.)$ is a sigmoid function with output scale from 0 to 1. The first part $\alpha_{gk}(\dot{x}_i)$ measures the soft assignment of $\tilde{x}_i^g$ to cluster k, while the second part $\alpha_g(\dot{x}_i)$ can be regarded as an attention function over groups.

Then, a video-level descriptor will be achieved via aggregating the encoded vectors over time and groups:

$$y_{jk} = \sum_{i,g} v_{ijk}^g, \tag{8}$$

after which we apply an intra-normalization operation, a dimension reduction fully-connected layer and a video-level classifier as same as those of the NetVLAD aggregation network.

As noted in Figure 2, because the dimension of video-level descriptors y_{jk} is reduced by G times than NetVLAD, the number of parameters shrinks. Specifically, the total number of parameters are:

$$\lambda N(N + G + K(G + \frac{H+1}{G})). \tag{9}$$

Since G is much smaller than H and N, roughly speaking, the parameter number of NeXtVLAD is about $\frac{G}{\lambda}$ times smaller than that of NetVLAD. For instance, a NeXtVLAD network with $\lambda = 2$, $G = 8$, $N = 1024$, $K = 128$ and $H = 2048$ only contains $71M+$ parameters, which is about 4 times smaller than that of NetVLAD, $268M+$.

3.3 NeXtVLAD Model and SE Context Gating

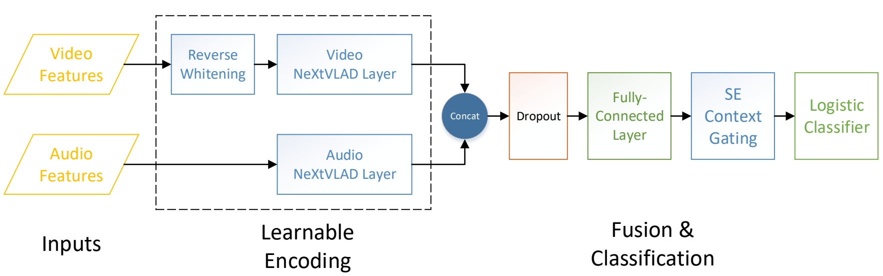

Fig. 3. Overview of the NeXtVLAD model designed for Youtube-8M video classification.

The basic model we used for 2nd Youtube-8M challenge has the similar architecture with the winner solution[5] of first Youtube-8M challenge. Video and audio features are encoded and aggregated separately with a two-stream architecture. The aggregated representation is enhanced by a SE Context Gating module, aiming to modeling the dependency among labels. At last, a logistic classifier with sigmoid activation is adopted for video-level multi-label classification.

Inspired by the work of Squeeze-and-Excitation networks[28], as shown in Figure 4, the SE Context Gating consists of 2 fully-connected layers with less parameters than the original Context Gating introduced in [5]. The total number of parameters are:

$$\frac{2F^2}{r} \tag{10}$$

where r denotes the reduction ratio that is set to be 8 or 16 in our experiments. During the competition, we find that reversing the whitening process,

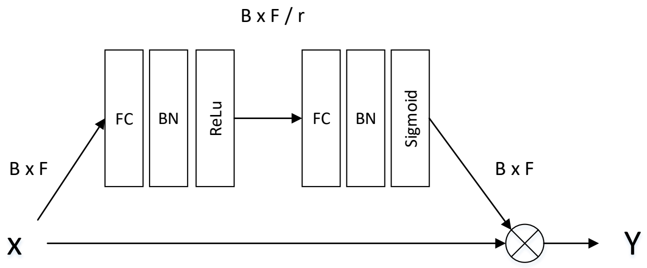

Fig. 4. The schema of the SE Context Gating. FC denotes fully-connected and BN denotes batch normalization. B represents the batch size and F means the feature size of x.

which is applied after PCA dimension reduction in frame feature extraction, is beneficial for the generalization performance of NeXtVLAD model. The possible reason is that whitening after PCA will distort the feature space by eliminating the different contribution between feature dimensions with regard to distance measurements, which could be critical for the encoder to find better anchor points and soft assignments for each input feature. Since the Eigen values $\{e_j | j \in \{1, ..., N\}\}$ of PCA transformation is released by the Google team, we are able to reverse the whitening process by:

$$\hat{x}_j = x_j * \sqrt{e_j} \tag{11}$$

where x and $\hat{x}$ are the input and reversed vector respectively.

3.4 Knowledge Distillation with On-the-fly Naive Ensemble

Knowledge distillation[29][30][31] was designed to transfer the generalization ability of the cumbersome teacher model to a relatively simpler student network by using prediction from teacher model as an additional "soft target" during training. During the competition, we tried the network architecture introduced in [32] to distill knowledge from a on-the-fly mixture prediction to each submodel.

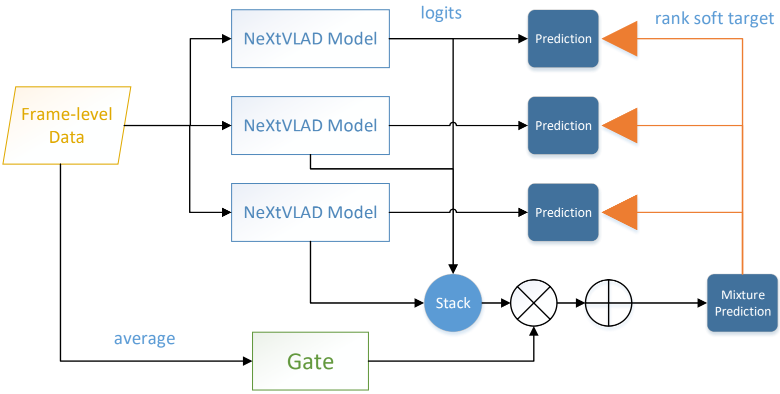

Fig. 5. Overview of a mixture of 3 NeXtVLAD models with on-the-fly knowledge distillation. The orange arrows indicate the distillation of knowledge from mixture predictions to the submodels.

As shown in Figure 5, the logits of the mixture predictions z^e is a weighted sum of logits $\{z^m | m \in \{1, 2, 3\}\}$ from the 3 corresponding submodels:

$$z^e = \sum_{m=1}^{3} a_m(\bar{x}) * z^m \tag{12}$$

where $a_m(.)$ represents the gating network,

$$a_m(\bar{x}) = \frac{e^{w_m^T \bar{x} + b_m}}{\sum_s^3 e^{w_s^T \bar{x} + b_s}} \tag{13}$$

and $\bar{x}$ represents the frame mean of input features x. The knowledge of the mixture prediction is distill to each submodel via minimizing the KullbackLeibler divergence written as:

$$\mathcal{L}_{kl}^{m,e} = \sum_{c=1}^{C} p^e(c) \log \frac{p^e(c)}{p^m(c)}, \tag{14}$$

where C is the total number of class labels and $p(.)$ represents the rank soft prediction:

$$p^m(c) = \frac{exp(z_c^m/T)}{\sum_{s=1}^{C} exp(z_s^m/T)},\ p^e(c) = \frac{exp(z_c^e/T)}{\sum_{s=1}^{C} exp(z_s^e/T)}. \tag{15}$$

T is a temperature which can adjust the relative importance of logits. As suggested in [29], larger T will increase the importance of logits with smaller values and encourage models to share more knowledge about the learned similarity measurements of the task space. The final loss of the model is:

$$\mathcal{L} = \sum_{m=1}^{3} \mathcal{L}_{bce}^m + \mathcal{L}_{bce}^e + T^2 * \sum_{m=1}^{3} \mathcal{L}_{kl}^{m,e} \tag{16}$$

where $\mathcal{L}_{bce}^m$ ($\mathcal{L}_{bce}^e$) means the binary cross entropy between the ground truth labels and prediction from model m (mixture prediction).

4 Experiments

This section provides the implementation details and presents results on the Youtube-8M dataset[2].

4.1 Youtube-8M Dataset

Youtube-8M dataset (2018) consists of about 6.1M videos from Youtube.com, each of which has at least 1000 views with video time ranging from 120 to 300 seconds and is labeled with one or multiple tags from a vocabulary of 3862 visual entities. These videos further split into 3 partitions: train(70%), validate(20%) and test(10%). Along with the video ids and labels, visual and audio features are provided for every second of the videos, which are referred as frame-level features. The visual features consists of hidden representations immediately prior to the classification layer in Inception[33], which is pre-trained on Imagenet[34]. The audio features are extracted from a audio classification CNN[35]. PCA and whitening are then applied to reduce the dimension of visual and audio feature to 1024 and 128 respectively.

In the 2nd Youtube-8M video understanding challenge, submissions are evaluated using Global Average Pricision(GAP) at 20. For each video, the predictions are sorted by confidence and the GAP score is calculated as:

$$GAP = \sum_{i=1}^{20} p(i) * r(i) \tag{17}$$

in which $p(i)$ is the precision and $r(i)$ is the recall given the top i predictions.

4.2 Implementation Details

Our implementation is based on the TensorFlow[36] starter code[1]. All of the models are trained using the Adam optimizer[37] with an initial learning rate of 0.0002 on two Nvidia 1080 TI GPUs. The batch size is set to be 160 (80 on each GPU). We apply a l_2(1e-5) regularizer to the parameters of the video-level classifier and use a dropout ratio of 0.5 aiming to avoid overfitting. No data augmentation is used in training NeXtVLAD models and the padding frames are masked out during the aggregation process via:

$$v_{ijk}^{g} = mask(i)\alpha_g(\dot{x}_i)\alpha_{gk}(\dot{x}_i)(\tilde{x}_{ij}^{g} - c_{kj}) \tag{18}$$

where

$$mask(i) = \begin{cases} 1 & \text{if} \quad i <= M \\ 0 & \text{else} \end{cases} \tag{19}$$

In all the local experiments, models are trained for 5 epochs (about 120k steps) using only the training partition and the learning rate is exponentially decreased by a factor of 0.8 every 2M samples. Then the model is evaluated using only about $\frac{1}{10}$ of the evaluation partition, which is consistently about 0.002 smaller than the score at public leaderboard[2] for the same models. As for the final submission model, it is trained for 15 epochs (about 460k steps) using both training and validation partitions and the learning rate is exponentially decreased by a factor of 0.9 every 2.5M samples. More details can be found at https://github.com/linrongc/youtube-8m.

4.3 Model Evaluation

Table 1. Performance (on local validation partition) comparison for single aggregation models. The parameters inside parenthesis represents (group number G, dropout ratio, cluster number K, hidden size H)

Model	Parameter	GAP
NetVLAD (-, 0.5drop, 128K, 2048H)	297M	0.8474
NetVLAD_random (-, 0.5drop, 256k, 1024H)	274M	0.8507
NetVLAD_small (-, 0.5drop, 128K, 2048H)	88M	0.8582
NeXtVLAD (32G, 0.2drop, 128K, 2048H)	55M	0.8681
NeXtVLAD (16G, 0.2drop, 128K, 2048H)	58M	0.8685
NeXtVLAD (16G, 0.5drop, 128K, 2048H)	58M	0.8697
NeXtVLAD (8G, 0.5drop, 128K, 2048H)	89M	0.8723

We evaluate the performance and parameter efficiency of individual aggregation models in Table 1. For fair comparison, we apply a reverse whitening

[1] https://github.com/google/youtube-8m
[2] https://www.kaggle.com/c/youtube8m-2018/leaderboard

layer for video features, a dropout layer after concatenation of video and audio features and a logistic model as the video-level classifier in all the presented models. Except for NetVLAD_random which sampled 300 random frames for each video, all the other models didn't use any data augmentation techniques. NetVLAD_small use a linear fully-connected layer to reduce the dimension of inputs to $\frac{1}{4}$ of the original size for visual and audio features, so that the number of parameters are much comparable to other NeXtVLAD models.

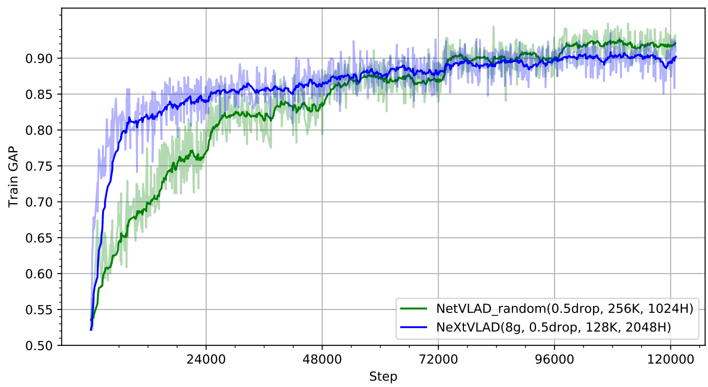

Fig. 6. Training GAP on Youtube-8M dataset. The ticks of x axis are near the end of each epoch.

From Table 1, we can observe that our proposed NeXtVLAD neural networks are both more effective and parameter efficient than the original NetVLAD model by a significantly large margin. With only about 30% of the size of NetVLAD_random model[5], NeXtVLAD increase the GAP score by about 0.02, which is a significant improvement considering the large size of Youtub-8M dataset. Furthermore, as shown in Figure 6, the NeXtVLAD model is converging faster, which reaches a training GAP score of about 0.85 in just 1 epoch.

Surprisely, the NetVLAD model performs even worse than the NetVLAD_small model, which indicates NetVLAD models tend to overfit the training dataset. Another interesting observation in Figure 6 is that the most of GAP score gains happens around the beginning of a new epoch for NetVLAD model. The observation implies that the NetVLAD model are more prone to remember the data instead of find useful feature patterns for generalization.

To meet of the requirements of the competition, we use an ensemble of 3 NeXtVLAD models with parameters (0.5drop, 112K, 2048H), whose size is about 944M bytes. As shown in Table 2, training longer can always lead to better performance of NeXtVLAD models. Our best submission is trained about 15 epochs, which takes about 3 days on 2 1080 TI GPUs. If we only retain one

Table 2. The GAP scores of submissions during the competition. All the other parameters used are (0.5drop, 112K, 2048H). The final submissions are tagged with *

Model	Parameter	Private GAP	Public GAP
single NeXtVLAD(460k steps)	79M	0.87846	0.87910
3 NeXtVLAD (3T, 250k steps)	237M	0.88583	0.88657
3 NeXtVLAD (3T, 346k steps)	237M	0.88681	0.88749
3 NeXtVLAD* (3T, 460k steps)	237M	0.88722	0.88794
3 NeXtVLAD* (3T, 647k steps)	237M	0.88721	0.88792

branch from the mixture model, a single NeXtVLAD model with only 79M parameters will achieve a GAP score of 0.87846, which could be ranked 15/394 in the final leaderboard.

Due to time and resource limit, we set the parameters $T = 3$, which is the temperature in on-the-fly knowledge distillation, as suggested in [32]. We ran an AB test experiments after the competition, as shown in 3, somehow indicates $T = 3$ is not optimal. Further tuning of the parameter could result in a better GAP score.

Table 3. The results (on local validation set) of an AB test experiment for T tuning.

Model	Parameter	local GAP
3 NeXtVLAD (0T, 120k steps)	237M	0.8798
3 NeXtVLAD (3T, 120k steps)	237M	0.8788

5 Conclusion

In this paper, we tackle the problem of large-scale video classification under budget constraints with the novel NeXtVLAD model. The experimental results on Youtube-8M dataset shows the proposed networks are both more effective and parameter efficient than the previous state-of-the-art model, NetVLAD, which is the winner of the first Youtube-8M video understanding challenge.

6 Acknowledgement

The authors would like to thank kaggle and the Google team for hosting the Youtube-8M video understanding challenge and providing the Youtube-8M Tensorflow Starter Code.

References

1. Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN architecture for weakly supervised place recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. (2016)

2. Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, A.P., Toderici, G., Varadarajan, B., Vijayanarasimhan, S.: Youtube-8m: A large-scale video classification benchmark. In: arXiv:1609.08675. (2016)
3. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8) (November 1997) 1735–1780
4. Cho, K., van Merrienboer, B., Glehre, a., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In Moschitti, A., Pang, B., Daelemans, W., eds.: EMNLP, ACL (2014) 1724–1734
5. Miech, A., Laptev, I., Sivic, J.: Learnable pooling with context gating for video classification. CoRR (2017)
6. Xie, S., Girshick, R.B., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. CoRR (2016)
7. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: IEEE International Conference on Computer Vision. Volume 2. (2003) 1470–1477
8. Perronnin, F., Dance, C.R.: Fisher kernels on visual vocabularies for image categorization. In: CVPR, IEEE Computer Society (2007)
9. Jegou, H., Douze, M., Schmid, C., Prez, P.: Aggregating local descriptors into a compact image representation. In: CVPR, IEEE Computer Society (2010) 3304–3311
10. Laptev, I., Marszaek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: IN: CVPR. (2008)
11. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local svm approach. In: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03. ICPR '04, IEEE Computer Society (2004) 32–36
12. Girdhar, R., Ramanan, D., Gupta, A., Sivic, J., Russell, B.: ActionVLAD: Learning spatio-temporal aggregation for action classification. In: CVPR. (2017)
13. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR. (2014)
14. Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J.: Activitynet: A large-scale video benchmark for human activity understanding. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2015)
15. Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A.: Sequential deep learning for human action recognition. In: Proceedings of the Second International Conference on Human Behavior Unterstanding. HBU'11, Springer-Verlag (2011) 29–39
16. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1. NIPS'14, MIT Press (2014) 568–576
17. Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. (January 2013) 221–231
18. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). ICCV '15, IEEE Computer Society (2015) 4489–4497
19. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. CoRR (2016)
20. Wu, Z., Jiang, Y., Wang, X., Ye, H., Xue, X., Wang, J.: Fusing multi-stream deep networks for video classification. CoRR (2015)

21. Ng, J.Y.H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: Computer Vision and Pattern Recognition. (2015)
22. Ballas, N., Yao, L., Pal, C., Courville, A.C.: Delving deeper into convolutional networks for learning video representations. CoRR (2015)
23. Fernando, B., Gavves, E., Oramas, J.M., Ghodrati, A., Tuytelaars, T.: Modeling video evolution for action recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2015)
24. Wang, X., Farhadi, A., Gupta, A.: Actions ˜ transformations. In: CVPR. (2016)
25. Bilen, H., Fernando, B., Gavves, E., Vedaldi, A.: Action recognition with dynamic image networks. CoRR **abs/1612.00738** (2016)
26. Wang, L., Li, W., Li, W., Gool, L.V.: Appearance-and-relation networks for video classification. Technical report, arXiv (2017)
27. Arandjelovic, R., Zisserman, A.: All about vlad. In: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, IEEE Computer Society (2013) 1578–1585
28. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR. (2018)
29. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop. (2015)
30. Zhang, Y., Xiang, T., Hospedales, T.M., Lu, H.: Deep mutual learning. CoRR (2017)
31. Li, Z., Hoiem, D.: Learning without forgetting. CoRR (2016)
32. Xu Lan, Xiatian Zhu, S.G.: Knowledge distillation by on-the-fly native ensemble. In: arXiv:1806.04606. (2018)
33. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15, JMLR.org (2015) 448–456
34. Deng, J., Dong, W., Socher, R., jia Li, L., Li, K., Fei-fei, L.: Imagenet: A large-scale hierarchical image database. In: In CVPR. (2009)
35. Hershey, S., Chaudhuri, S., Ellis, D.P.W., Gemmeke, J.F., Jansen, A., Moore, R.C., Plakal, M., Platt, D., Saurous, R.A., Seybold, B., Slaney, M., Weiss, R.J., Wilson, K.W.: CNN architectures for large-scale audio classification. CoRR (2016)
36. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X.: Tensorflow: A system for large-scale machine learning. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. OSDI'16, USENIX Association (2016) 265–283
37. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR (2014)