Impact of Temporal Subsampling on Accuracy and Performance in Practical Video Classification

F. Scheidegger*†, L. Cavigelli*, M. Schaffner*, A. C. I. Malossi†, C. Bekas†, L. Benini*‡
*ETH Zürich, 8092 Zürich, Switzerland
†IBM Research - Zürich, 8803 Rüschlikon, Switzerland
‡Università di Bologna, Italy

***Abstract*—In this paper we evaluate three state-of-the-art neural-network-based approaches for large-scale video classification, where the computational efficiency of the inference step is of particular importance due to the ever increasing amount of data throughput for video streams. Our evaluation focuses on finding good efficiency vs. accuracy tradeoffs by evaluating different network configurations and parameterizations. In particular, we investigate the use of different temporal subsampling strategies, and show that they can be used to effectively trade computational workload against classification accuracy. Using a subset of the YouTube-8M dataset, we demonstrate that workload reductions in the order of $10\times$, $50\times$ and $100\times$ can be achieved with accuracy reductions of only 1.3%, 6.2% and 10.8%, respectively. Our results show that temporal subsampling is a simple and generic approach that behaves consistently over the considered classification pipelines and which does not require retraining of the underlying networks.**

I. Introduction

Over the last years, two key enablers have driven the success of machine learning, and in particular neural network (NN) based approaches. First, the availability of large scale datasets with known ground truth [1–9] enables supervised learning for complex tasks such as face recognition [1], [2], action recognition [3], [4] or video classification [8–10]. Second, the availability of increased computational performance in today's computing systems typically achieved with graphics processing units (GPUs) enables to train large scale models.

State-of-the-art NNs for image classification typically have 10-200 million parameters and require 10-25 billion arithmetic operations to perform inference for a single image [11]. Such deep networks achieve high classification accuracies, but also require a long training time [12]. While the race to improve accuracy on challenges such as the ILSVRC [13] drives the community to develop ever more complex models, this trend is likely to continue with the increasing availability of video-based datasets. In order for these NNs to remain economically viable, it is important to keep the computational effort in mind to reduce the costs involved when building practical systems for large-scale inference, such as energy and infrastructure expenditures [14]. In this paper, we consider the task of video classification on large-scale datasets using single-frame and multi-frame NN approaches, where the computational burden is exacerbated by the fact that complete video sequences have to be considered. We accelerate the inference step and evaluate the impact of the temporal sampling on classification accuracy. To this end, we make the following two contributions. First, we compare three video classification approaches on a subset of the YouTube-8M dataset [9] and report the achieved accuracy and the associated computational workload. We then evaluate temporal subsampling as a simple and efficient method to extend video classification approaches to produce Pareto optimal fronts of accuracy v. workload tradeoffs. We show the generality of the approach by considering two frame subsampling strategies and applying them to different configurations of the three classification approaches. Temporal subsampling does not require any changes nor retraining of the underlying classification system.

This work was funded by the the European Union's H2020 research and innovation programme under grant agreement No 732631, project OPRECOMP.

II. Related Work

Traditional video classification approaches [15–17] based on global video descriptors have demonstrated success on a variety of datasets. First, interest points are localized and visual features are extracted, then compressed to a constant length global descriptor. Second, a standard classifier (e.g., SVM) predicts the final class.

Recent methods improve by adopting a data driven approach where also the features are learned. In [18], [19] the 3D extension of convolutional NNs (CNNs) is explored. However, this approach does not scale well to long videos and has thus only been applied to short video clips with a length in the order of seconds. Ng et al. [10] show that different feature pooling architectures employing long short-term memory (LSTM) [20] are better suited to video classification.

However, end-to-end training approaches become infeasible for very large datasets as the YouTube-8M. It contains 50 years of video footage, and even sequential processing of the individual frames becomes a demanding task. Finding good NN configurations is non-trivial due manual tuning and many learning and validation cycles, which may take weeks or even months. E.g., assuming that a single GPU allows processing at a rate of $\sim$4.3 fps[1], one pass through the 5.8 M training videos of YouTube-8M with an average of 230 frames/video requires around 9.8 years. Renting a cloud infrastructure with multiple GPUs allows to cut the total amount of time but the required costs are in the order of 75 000 $[2]. The estimated

[1]Based on the time required for one forward/backward pass for GoogLeNet using Nervana Systems' neon library on a Nvidia GTX Titan X (Maxwell) GPU https://github.com/soumith/convnet-benchmarks.

ISBN 978-0-9928626-7-1 © EURASIP 2017

cost of complete end-to-end training with the full dataset and a typical amount of 100-1000 learning epochs is therefore around 7.5 M $-75 M $, which is infeasible for most institutions. To this end, the baseline methods published together with the YouTube-8M dataset split the classification problem into two steps. First, frames are mapped to feature vectors with lower dimension using a state-of-the-art image classification network (Inception-v3). Second, classification is performed on the feature vectors. This approach has the advantage that pure data driven learning is still possible and computational costs remain tractable.

Fast inference methods, such as Low Rank Expansions [21] reach speedups $< \mathbf{5}\times$ at accuracy drops $< \mathbf{1}\%$. XNOR-Nets [22] gain $\mathbf{58}\times$ in speed at $\mathbf{12.5}\%$ accuracy reduction on ImageNet. Even though, we solve the more complex problem of video classification, our approach is on par with Low Rank Expansions and outperforms the XNOR-Nets trade-off.

III. Practical Video Classification Systems

Our approach targets a large-scale machine learning system based on the two-step approach introduced together with the YouTube-8M dataset. For practical scalability reasons, we avoid end-to-end learning approaches since the raw videos of that dataset amount to a data volume of ~ 1 Petabyte.

In this two-step approach, frame feature vectors are obtained by first extracting features using a CNN trained on an image classification task (without the final classification layer), before applying principal component analysis (PCA) to reduce the dimensionality to 1024 per frame and feature vector. These feature vectors are then used for video classification in a second step. Such a decomposition of the classification problem allows a practical intermediate representation of the video and the handling of the training of two separate subproblems. The vectors have been precomputed using the Inception-v3 CNN and are available as part of the YouTube dataset, which is convenient from a practical viewpoint as it enables to perform evaluations without having to evaluate the complete CNN.

In our evaluation, we consider three classification pipelines:

a) Feature vector aggregation [9], that produces a *global video descriptor* (**GVD**) which is then classified using a fully-connected neural network (FCNN).
b) *Frame-based classification* (**FBC**) [9] with an FCNN, followed by an aggregation of the classification results.
c) A *long short-term memory* (**LSTM**) architecture [10] processes the complete vector sequence of the video.

The three classification system variants are illustrated in Figure 1, and only differ in the way the 1024 dimensional feature vectors are processed, as explained in the following.

A. Global Video Descriptor (GVD)

As noted by the authors of YouTube-8M [9], GVD exhibits three advantages: fixed-length vectors allow to apply standard learning, the amount of data involved in learning is reduced by the average length of the video ($230\times$), and the approach is generic. As a baseline, we compute the video average feature vector μ as mean over the feature sequence. Since the provided feature vectors are already normalized, no further normalization is required and the resulting GVD μ is fed directly into a FCNN for final classification. The chosen network configurations for this evaluation are listed in Table I.

[2]Based on a 650 $/month rent for a high-end GPU https://aws.amazon.com/ec2/instance-types/p2/.

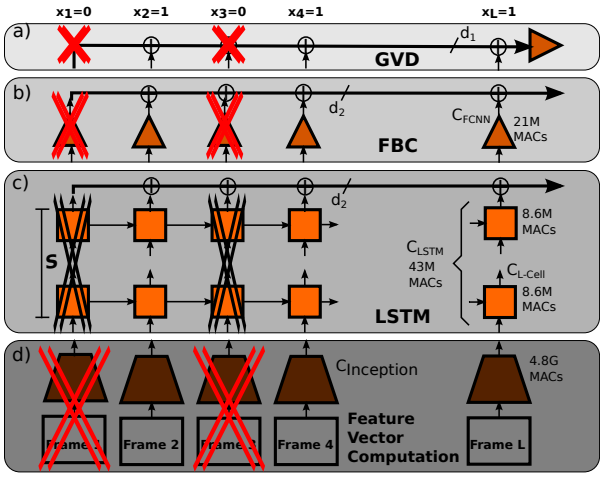

Fig. 1. Overview of the 3 video classification approaches. First, an inference with the Inception-v3 network (d) produces frame level feature vectors. Second, GVD (a), FBC (b) and LSTM (c) are applied to classify the video. Frames i are skipped if the corresponding decision vector entries $\mathbf{x}_i$ are 0. This is illustrated with red crosses for a regular temporal subsampling of $2\times$.

The FCNN has 1024 input dimensions, followed by two dense layers that end in H_1 and H_2 neurons, each using ReLU activations which are defined element-wise as $x \mapsto \max(0, x)$. The third, final layer outputs a softmax-normalized prediction for each class obtained by $\mathbf{x} \mapsto \frac{e^{\mathbf{x}}}{\|e^{\mathbf{x}}\|_1}$.

B. Frame Based Classification (FBC)

This approach maps each per-frame feature vector to a video class prediction, and all predictions of a video sequence are then aggregated to form the final classification. Learning of the FCNN is achieved by using the global label of a particular video sequence as local ground-truth label for all frames in that video. The final classification is obtained either by average or max-voting aggregation of the per-frame results. We observed that averaging consistently outperforms a max-voting aggregation and thus use the averaging scheme henceforth.

C. Long Short-Term Memory (LSTM)

LSTM cells [20] are used to implement a recurrent neural network (RNN) [23], that enables learning of high-level concepts from the temporal information of the input sequence. Long-update chains in RNN architectures may cause vanishing or exploding gradient problems [24]. LSTMs are able to mitigate that problem by having an internal state storing long temporal information, whereas the rest of the structure (*input*, *output* and *forget* gates) is modeled to capture local (short-term) effects. Table I lists the hyper-parameter choice used in our evaluations. H denotes the dimensionality of the involved hidden state and S is the amount of stacked LSTM-cell chains

ISBN 978-0-9928626-7-1 © EURASIP 2017

TABLE I
HYPER-PARAMETER CONFIGURATIONS AND CAUSED WORKLOADS IN TERMS OF MILLION MAC OPERATIONS PER BUILDING BLOCK. FCNN CONFIGURATIONS ARE USED IN BOTH, THE GVD AND FBC PIPELINES.

FCNN	k=0	1	2	3	4	5	6	7
H_1	4096	4096	1024	512	256	128	64	32
H_2	1024	4096	1024	512	256	128	64	32
C_{FCNN}	8.4	21	2.1	0.8	0.33	0.15	0.07	0.03
LSTM	k=0	1	2	3	4	5	6	7
H	32	64	128	512	1024	32	64	128
S	1	1	1	1	1	2	2	2
C_{LSTM}	0.02	0.05	0.16	2.21	8.62	0.03	0.09	0.32
LSTM	k=8	9	10	11	12	13	14	
H	512	1024	32	64	128	512	1024	
S	2	2	5	5	5	5	5	
C_{LSTM}	4.42	17.2	0.08	0.23	0.79	11.0	43.0	

over the full sequence. The LSTM configuration $k = 13$ corresponds to the proposed configuration from [10] and $k = 9$ refers to the configuration used in [9].

D. Computational Complexity

The number of MAC operations reliably estimates the inference time for a wide class of CNNs [25]. Hence, we state the workload for one video inference as:

$$\begin{aligned} C_{\text{GVD}} &= L/r \cdot (C_{\text{Inception}} + d_1) + C_{\text{FCNN}_k}, \\ C_{\text{FBC}} &= L/r \cdot (C_{\text{Inception}} + C_{\text{FCNN}_k} + d_2), \\ C_{\text{LSTM}} &= L/r \cdot (C_{\text{Inception}} + C_{\text{LSTM}_k} + d_2), \end{aligned} \tag{1}$$

where L refers to the length of the video sequence, $d_1 = 1024$ is the dimension of the used feature vectors, $d_2 = 20$ the number of classes, $C_{\text{Inception}} = 4.8 \cdot 10^9$ the workload to compute the feature vectors, C_{FCNN_k} and C_{LSTM_k} the FCNN configurations according Table I, S the number of stacked layers and r the temporal subsampling factor that will be introduced in more detail in the next section. Due to the involved orders of magnitude, we have that $C_{\text{Inception}} \gg C_{\text{FCNN}_k}$, $C_{\text{Inception}} \gg C_{\text{LSTM}_k}$ and $C_{\text{Inception}} \gg d_{1,2}$. Therefore, we observe that $C_{\text{GVD}} \approx C_{\text{FBC}} \approx C_{\text{LSTM}} \approx L/r \cdot C_{\text{Inception}}$.

IV. TEMPORAL SUBSAMPLING

More than 99% of the inference workload is caused by the first frame-level feature extraction step, irrespective of the actual classifier chosen. It is therefore crucial to introduce optimizations leading to fewer computations in the first step.

There are two common approaches to achieve this. First, a more efficient network architecture than Inception-v3 could be sought by exploring different network topologies, parametrizations and quantizations such as in [26]. Second, a more efficient CNN evaluation engine could be built by means of specialized hardware [27]. While the first approach requires several long training and validation cycles, the second approach requires custom hardware design which is a time consuming task. In this paper, we explore a third approach, where temporal subsampling is used to significantly reduce the amount of CNN evaluations in the first step. Note that this is orthogonal to the other two approaches and does not involve any expensive training iterations of the large CNN in the first step. In this paper, we use the full length of the videos to train the three architectures described in Section III, and introduce temporal subsampling during inference to save computational complexity. Reducing the workload of the inference step is especially critical in large-scale datacenter applications where video footage is being uploaded at an ever increasing rate[3]. We define a decision function $d(.)$ that outputs a Boolean vector $\mathbf{x} \in \{0,1\}^L$ determining whether a given frame at index i in the video sequence is processed ($x_i = 1$) or skipped ($x_i = 0$). A skipped frame allows to save a full CNN evaluation in the first step, and the computational workload decreases linearly with the amount of skipped frames.

A. Subsampling Strategies

Regular subsampling $d_{\text{reg}}(L, r)$ reduces the total amount of frames L by a fixed factor r in an uniform manner, and has the advantage that it can be applied at almost no cost. *Prior-based subsampling* leverages the observation that the sampling positions of the selected frames have a significant impact on the overall classification outcome.

For illustration, we consider the conditional probability $p(f \text{ sufficient}|f_r)$ given the relative frame position $f_r \in [0,1]$ that a single frame f is sufficient to correctly classify the full video sequence. We empirically estimate $\phi(f_r) = \text{Interpolate}(\text{Histogram}(f \text{ is sufficient}))$ as interpolated aggregation of normalized histograms where correct classification results from Section III-B are binned according f_r. Figure 2 shows that frames taken from the middle of the video sequences are more likely to lead to correct classification outcomes. This is probably due to the fact that most videos contain lead-in and lead-out portions. The prior-based subsampling strategy $d_{\text{prior}}(L, r, \phi)$ introduces this a-priory knowledge by randomly drawing $\lfloor L/r \rfloor$ frame positions from a distribution that is obtained by scaling $\phi(f_r)$ to match the sequence length L.

V. EVALUATION AND RESULTS

Section V-A gives more details on the employed dataset, Section V-B and Section V-C explain the training procedure and Section V-D finally states the comparison of achieved accuracy versus workload tradeoffs.

A. Subset of YouTube-8M

To shorten the development cycle and make the evaluation practicable, we constructed a subset of the YouTube-8M dataset, a multi-labeled dataset with an average 1.8 of entity labels per video. We filter out all multi-labeled videos in order to create a single label problem. Table II states our choice, we assumed as practical resource constraint a data volume limit that fits on a single GPU (Nvidia GeForce GTX Titan X, 12GB RAM). Our subset allows hyper-parameter tuning and multiple repetitions of the full training to state the variance caused due to the random NN initialization in reasonable time. Note that this runtime reduction approach is suitable in our case, since

[3]For example, several hundreds video hours are uploaded to YouTube every minute https://fortunelords.com/youtube-statistics/.

ISBN 978-0-9928626-7-1 © EURASIP 2017

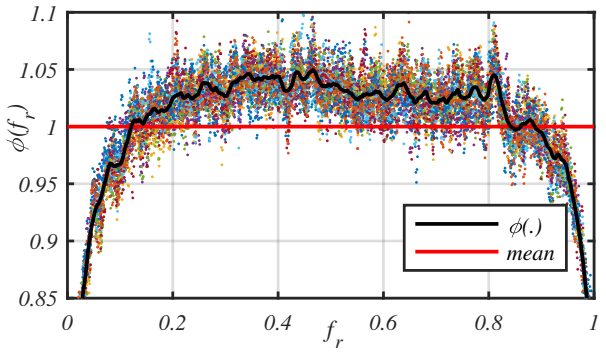

Fig. 2. Estimated probability density $\phi(f_r)$ that a single frame alone is sufficient to correctly classify the complete video sequence. *Prior-based subsampling* improves accuracy by using $\phi(f_r)$ to bias sampling points to frames that are beneficial for classification.

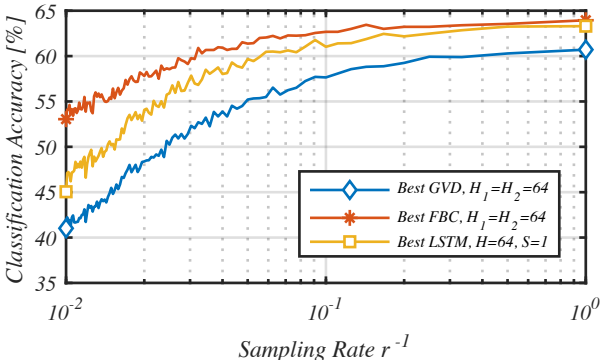

Fig. 3. Comparison of accuracy achieved with best configurations of GVD, FBC and LSTM approach as function of subsampling.

we are not interested in maximizing the absolute classification accuracy, but in the relative accuracy degradation behaviour due to temporal subsampling. The feature vectors provided in the YouTube dataset have been sampled at a rate of 1Hz (one vector per second of video) and the subsampling factor r used in this paper is relative to that rate.

B. FCNN Training

We used the Adam optimizer [28] to train the FCNNs by minimizing the average of the cross entropy with a learning rate of 10^{-4}. Note that both the GVD and FBC pipelines employ the same FCNN configurations listed in Table I. A batch size of 200 input samples is employed, and the initialization of the weights is achieved using random normal distributed values with a standard deviation of 0.1. The bias values are initialized with a constant offset of 0.1 to break the symmetry. Before the start of each epoch all input samples are shuffled with a uniform distribution over all possible permutations of the input samples. Training is run for 100 epochs.

C. LSTM Training

Similar to [9], we unrolled the LSTM for 60 iterations and trained on sequences of 60 consecutive frames at a random offset within the video for 380 epochs. The loss function weighs all individual frame losses with increasing values starting from $1/N$ for the first frame, up to 1 for the last frame in order to enhance learning and classification performance as in [9], [10]. Predictions are computed by considering the full length video. During training we used a dropout factor of 0.5.

D. Results & Discussion

TABLE II
DATASET USED IN THIS PAPER

	Original	Training Set	Validation Set
Number of Classes	4 800	20	20
Videos per Class	2229†	500	100
Total Videos	8 264 650	10 000	2000
Average Video Length	229.6	227.8	228.2
Number of frames	$1.9 \cdot 10^9$	2 277 717	456 427
Feature data volume	$\sim$ 1 PB	$\sim$ 9 GB	$\sim$ 2 GB

† Average value, videos/entity $\in [120, 539\,926]$ [9]

In all experiments, the subsampling factor has been swept over the range $1\times$ to $100\times$. The accuracy is measured as Top-k rate with $k = 1$, i.e., the number of correct classified videos relative to all classified videos.

Figure 3 compares the accuracy behaviour of the three approaches depending on *regular subsampling*. We explain the stronger decay in case of the LSTM approach by the fact that LSTM was explicitly trained to learn temporal information which is partially destroyed by subsampling. Still, the best configurations remain ordered as a function of subsampling.

Figure 4 shows the obtained classification accuracy versus computational workload for the GVD (a), FBC (b) and LSTM (c) approaches using *regular subsampling*. Even though different NNs and different classification systems are considered, the characteristic caused by subsampling is consistent which demonstrates the generality of the approach. In the range where $r \leq 5$ the accuracy degradation is negligible. For larger r factors, subsampling gradually decreases the classification accuracy. Interestingly, FBC seems to be more robust against subsampling than the other two approaches. The best LSTM and GVD configurations lose around 8% of classification accuracy from $r = 50\times$ to $r = 100\times$, whereas FBC only loses 5%. Figure 4 d), e) and f) show the additional accuracy improvements (in terms of percentage differences) achieved when employing *prior-based subsampling* instead of *regular subsampling*. Positive values are in favour of *prior-based subsampling*. Noise levels appear more pronounced due the zoomed-in view. Leveraging prior knowledge in aggressively subsampled regions allows to improve the accuracy around 2% for the GVD and FBC approaches, and around 4% for the LSTM at negligible extra cost.

VI. CONCLUSION

We considered three video classification approaches that employ a two-step classification procedure and presented accuracy and computational complexity results for various NN configurations. Our evaluations show that, while the LSTM-based pipeline outperforms a simple GVD approach, similar classification accuracies as achieved by the LSTM approach can be reached by aggregating frame-based classifier (FBC) results. The FBC method has the advantage that it is considerably easier to train than an LSTM approach.

ISBN 978-0-9928626-7-1 © EURASIP 2017

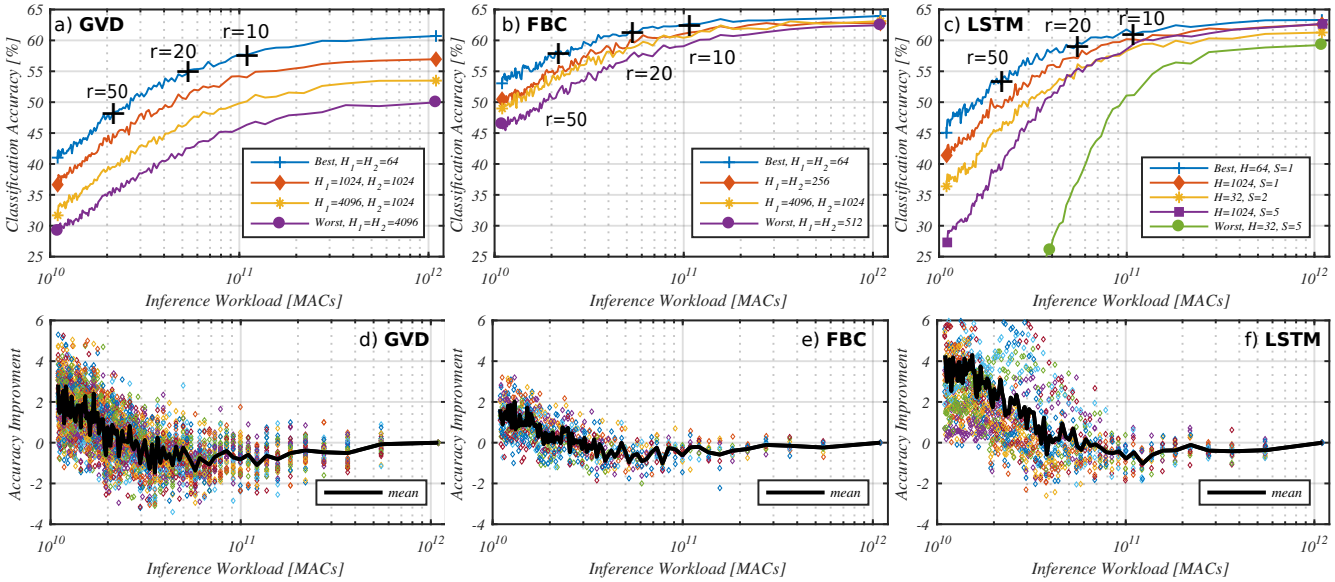

Fig. 4. Comparison of accuracy versus workload tradeoffs achieved with *regular subsampling* for different configurations for the a) GVD b) FBC and c) LSTM approach. For each approach, a selection of representative curves is shown, comprising the best, the worst, and two in-between configurations from Table I. Temporal subsampling allows to effectively trade computational workload against classification accuracy, thereby generating a wide range of Pareto optimal operating points. Improvements due to *prior-based subsamplig* over *regular subsampling* are quantified in d-f) for the three approaches.

Further, we note that most of the computational burden stems from the first step involving the evaluation of a large CNN, and investigate temporal subsampling as a simple yet effective way to trade computational complexity against classification accuracy. We introduce two different subsampling strategies and show that significant workload reductions in the order of $10\times$ can be achieved with negligible impact on classification accuracy. Larger workload reductions up to $100\times$ are possible, leading to accuracy degradations in the order of 10% for the best NN configurations.

Temporal subsampling is a simple and generic approach that does not require retraining of large CNNs, and which behaves consistently over the considered classification pipelines. It is straightforward to implement, and therefore well suited for static or dynamic load balancing and cost optimization in large-scale, industrial video classification systems.

References

[1] L. Wolf, T. Hassner, and I. Maoz, "Face Recognition in Unconstrained Videos with Matched Background Similarity," in *IEEE CVPR*, 2011.

[2] G. Zhao, X. Huang *et al.*, "Facial Expression Recognition From Near-Infrared Videos," *Image and Vision Computing*, vol. 29, no. 9, 2011.

[3] L. Xia, C.-C. Chen, and J. Aggarwal, "View Invariant Human Action Recognition using Histograms of 3D Joints," in *IEEE CVPRW*, 2012.

[4] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding," in *IEEE CVPR*, June 2015.

[5] K. G. Derpanis, M. Lecce, K. Daniilidis, and R. P. Wildes, "Dynamic Scene Understanding: The Role of Orientation Features in Space and Time in Scene Classification," in *IEEE CVPR*, 2012, pp. 1306–1313.

[6] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild," *CoRR*, vol. abs/1212.0402, 2012.

[7] O. Kliper-Gross, T. Hassner, and L. Wolf, "The Action Similarity Labeling Challenge," *IEEE TPAMI*, vol. 34, no. 3, pp. 615–621, 2012.

[8] A. Karpathy, G. Toderici *et al.*, "Large-scale Video Classification with Convolutional Neural Networks," in *IEEE CVPR*, 2014.

[9] S. Abu-El-Haija, N. Kothari *et al.*, "Youtube-8M: A Large-Scale Video Classification Benchmark," *arXiv:1609.08675*, 2016.

[10] J. Yue-Hei Ng, M. Hausknecht *et al.*, "Beyond Short Snippets: Deep Networks for Video Classification," in *IEEE VCPR*, June 2015.

[11] C. Szegedy, V. Vanhoucke *et al.*, "Rethinking the Inception Architecture for Computer Vision," *CoRR*, vol. abs/1512.00567, 2015.

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

[13] J. Deng, W. Dong *et al.*, "Imagenet: A large-scale hierarchical image database," in *IEEE CVPR*, 2009, pp. 248–255.

[14] A. Shehabi, S. J. Smith *et al.*, "United states data center energy usage report," 06/2016 2016.

[15] J. Liu, J. Luo, and M. Shah, "Recognizing Realistic Actions from Videos "In The Wild"," in *IEEE CVPR*, 2009, pp. 1996–2003.

[16] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *ECCV*. Springer, 2010, pp. 392–405.

[17] H. Wang, M. M. Ullah *et al.*, "Evaluation of Local Spatio-Temporal Features for Action Recognition," in *BMVC*. BMVA Press, 2009.

[18] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," *IEEE TPAMI*, vol. 35, no. 1, 2013.

[19] M. Baccouche, F. Mamalet *et al.*, "Sequential Deep Learning for Human Action Recognition," in *HBU*. Springer, 2011, pp. 29–39.

[20] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.

[22] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, *XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks*. Cham: Springer International Publishing, 2016, pp. 525–542.

[23] T. Mikolov, M. Karafiát *et al.*, "Recurrent neural network based language model." in *Interspeech*, vol. 2, 2010, p. 3.

[24] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE TNNLS*, 1994.

[25] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[26] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016.

[27] K. Ovtcharov, O. Ruwase *et al.*, "Accelerating deep convolutional neural networks using specialized hardware," *Microsoft Research Whitepaper*, vol. 2, no. 11, 2015.

[28] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, 2014.

ISBN 978-0-9928626-7-1 © EURASIP 2017