Training Compact Deep Learning Models for Video Classification Using Circulant Matrices

Alexandre Araujo[1,2](✉), Benjamin Negrevergne[1](✉), Yann Chevaleyre[1](✉), and Jamal Atif[1](✉)

[1] PSL, Université Paris-Dauphine, LAMSADE, CNRS, UMR 7243, Paris, France
alexandre.araujo@dauphine.eu, benjamin.negrevergne@dauphine.fr, {yann.chevaleyre,jamal.atif}@lamsade.dauphine.fr
[2] Wavestone, Paris, France

Abstract. In real world scenarios, model accuracy is hardly the only factor to consider. Large models consume more memory and are computationally more intensive, which make them difficult to train and to deploy, especially on mobile devices. In this paper, we build on recent results at the crossroads of Linear Algebra and Deep Learning which demonstrate how imposing a structure on large weight matrices can be used to reduce the size of the model. Building on these results, we propose very compact models for video classification based on state-of-the-art network architectures such as *Deep Bag-of-Frames*, *NetVLAD* and *NetFisherVectors*. We then conduct thorough experiments using the large *YouTube-8M* video classification dataset. As we will show, the circulant DBoF embedding achieves an excellent trade-off between size and accuracy.

Keywords: Deep learning · Computer vision · Structured matrices Circulant matrices

1 Introduction

The top-3 most accurate approaches proposed during the first *YouTube-8M*[1] video classification challenge were all ensembles models. The ensembles typically combined models based on a variety of deep learning architectures such as *NetVLAD*, *Deep Bag-of-Frames* (DBoF), *NetFisherVectors* (NetFV) and *Long-Short Term Memory* (LSTM), leading a large aggregation of models (25 distinct models have been used by the first contestant [24], 74 by the second [33] and 57 by the third [20]). Ensembles are accurate, but they are not ideal: their size make them difficult to maintain and deploy, especially on mobile devices.

A common approach to compress large models into smaller ones is to use *model distillation* [13]. Model distillation is a two steps training procedure: first, a large model (or an ensemble model) is trained to be as accurate as possible. Then, a second compact model is trained to approximate the first one,

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

L. Leal-Taixé and S. Roth (Eds.): ECCV 2018 Workshops, LNCS 11132, pp. 271–286, 2019.
https://doi.org/10.1007/978-3-030-11018-5_25

while satisfying the given size constraints. The success of model distillation and other model compression techniques begs an important question: is it possible to devise models that are compact by nature while exhibiting the same generalization properties as large ones?

In linear algebra, it is common to exploit structural properties of matrices to reduce the memory footprint of an algorithm. Cheng et al. [6] have used this principle in the context of deep neural networks to design compact network architectures by imposing a structure on weight matrices of fully connected layers. They were able to replace large, unstructured weight matrices with structured *circulant matrices* without significantly impacting the accuracy. And because a n-by-n circulant matrix is fully determined by a vector of dimension n, they were able to train a neural network using only a fraction of the memory required to train the original network.

Inspired by this result, we designed several compact neural network architectures for video classification based on standard video architectures such as NetVLAD, DBoF, NetFV and we evaluated them on the large *YouTube-8M* dataset. However, instead of adopting the structure used by [6] (initially proposed by [32]), we decomposed weight matrices into products of diagonal and circulant matrices (as in [29]). In contrast with [32] which has been proved to approximate distance preserving projections, this structure can approximate *any* transformation (at the cost of a larger number of weights). As we will show, this approach exhibit good results on the video classification task at hand.

In this paper, we bring the following contributions:

- We define a compact architecture for video classification based on circulant matrices. As a side contribution, we also propose a new pooling technique which improves the Deep Bag-of-Frames embedding.
- We conduct thorough experimentations to identify the layers that are less impacted by the use of circulant matrices and we fine-tune our architectures to achieve the best trade-off between size and accuracy.
- We combine several architectures into a single model to achieve new model trained-end-to-end that can benefit from architectural diversity (as in ensembles).
- We train all our models on the Youtube-8M dataset with the 1 GB model size constraint imposed by the *2nd YouTube-8M Video Understanding Challenge*[2], and compares the different models in terms of size vs. accuracy ratio. Our experiments demonstrate that the best trade-off between size and accuracy is obtained using circulant DBoF embedding layer.

2 Related Works

Classification of unlabeled videos streams is one of the challenging tasks for machine learning algorithms. Research in this field has been stimulated by the recent release of several large annotated video datasets such as *Sports-1M* [19], *FCVID* [17] or the *YouTube-8M* [2] dataset.

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

The naive approach to achieve video classification is to perform frame-by-frame image recognition, and to average the results before the classification step. However, it has been shown in [2,24] that better results can be obtained by building features across different frames and several deep learning architectures have been designed to learn embeddings for sets of frames (and not single frames). For example Deep Bag-of-Frames (DBoF) [2], NetVLAD [3] or architectures based on Fisher Vectors [27].

The DBoF embedding layer, proposed in [2] processes videos in two steps. First, a learned transformation projects all the frames together into a high dimensional space. Then, a max (or average) pooling operation aggregates all the embedded frames into a single discriminative vector representation of the video. The NetVLAD [3] embedding layer is built on *VLAD* [16], a technique that aggregates a large number of local frame descriptors into a compact representation using a codebook of visual words. In NetVlad, the codebook is directly learned end-to-end during training. Finally, NetFisherVector (NetFV) is inspired by [27] and uses first and second-order statistics as video descriptors also gathered in a codebook. The technique can benefit from deep learning by using a deep neural network to learn the codebook [24].

All the architectures mentioned above can be used to build video features in the sense of features that span across several frames, but they are not designed to exploit the sequential nature of videos and capture motion. In order to learn truly spatio-temporal features and account for motion in videos, several researchers have looked into recurrent neural networks (e.g. LSTM [20,36]) and 3D convolutions [19] (in space and time). However, these approaches do not outperform non-sequential models, and the single best model proposed in [24] (winner of the first *YouTube-8M* competition) is based on NetVLAD [3].

The *2nd YouTube-8M Video Understanding Challenge* includes a constraint on the model size and many competitors have been looking into building efficient memory models with high accuracy. There are two kinds of techniques to reduce the memory required for training and/or inference in neural networks. The first kind aims at *compressing* an existing neural network into a smaller one, (thus it only impacts the size of the model at inference time). The second one aims at *constructing models that are compact* by design.

To compress an existing network several researchers have investigated techniques to prune parameters that are redundant (e.g. [9,12,21]). Redundant parameters can be omitted from the model without significantly changing the accuracy. It is also possible to use sparsity regularizers during training, to be able to compress the model after the training using efficient sparse matrix representations (e.g. [7,9,22]). Building on the observation that weight matrices are often redundant, another line of research has proposed to use matrix factorization [10,15,35] in order to decompose large weight matrices into factors of smaller matrices before inference.

An important idea in model compression, proposed by Buciluă et al. ([4]), is based on the observation that the model used for training is not required to be the same as the one used for inference. First, a large complex model is trained

using all the available data and resources to be as accurate as possible, then a smaller and more compact model is trained to approximate the first model. The technique which was later specialized for deep learning models by [13] (a.k.a. model distillation) is often used to compress large ensemble models into compact single deep learning models.

Instead of compressing the model after the training step, one can try to design models that are compact by nature (without compromising the generalization properties of the network). The benefit of this approach is that it reduces memory usage required during both training and inference. As a consequence, users can train models that are virtually larger using less time and less computing resources. They also save the trouble of training two models instead of one as it is done with distillation. These techniques generally work by constraining the weight representation, either at the level of individual weights (e.g. using floating variable with limited precision [11], quantization [8,23,28]) or at the level of the whole matrix, (e.g. using weight hashing techniques [5]) which can achieve better compression ratio. However in practice, hashing techniques are difficult to use because of their irregular memory access patterns which make them inadequate for GPU-execution.

Another way of constraining the weight representation is to impose a structure on weight matrices (e.g. using circulant matrices [6,30], Vandermonde [30] or Fastfood transforms [34]). In this domain, [6] have proposed to replace two fully connected layers of AlexNet by circulant and diagonal matrices where the circulant matrix is learned by a gradient based optimization algorithm and the diagonal matrix entries are sampled at random in $\{-1, 1\}$. The size of the model is reduced by a factor of 10 without loss in accuracy[3]. Most of the time the resulting algorithms are easy to execute on GPU-devices.

3 Preliminaries on Circulant Matrices

In this paper, we use *circulant matrices* to build compact deep neural networks. A n-by-n circulant matrix C is a matrix where each row is a cyclic right shift of the previous one as illustrated below.

$$C = circ(c) = \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \cdots & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ c_2 & c_1 & c_0 & & c_3 \\ \vdots & & & \ddots & \vdots \\ c_{n-1} & c_{n-2} & c_{n-3} & & c_0 \end{bmatrix}$$

Because the circulant matrix $C \in \mathbb{R}^{n \times n}$ is fully determined by the vector $c \in \mathbb{R}^n$, the matrix C can be compactly represented in memory using only n real values instead of n^2.

[3] In network such as AlexNet, the last 3 fully connected layers use 58M out of the 62M total trainable parameters (>90% of the total number of parameters).

An additional benefit of circulant matrices, is that they are computationally efficient, especially on GPU devices. Multiplying a circulant matrix C by a vector x is equivalent to a circular convolution between c and x (denoted $c \star x$). Furthermore, the circular convolution can be computed in the Fourier domain as follows.

$$Cx = c \star x = \mathcal{F}^{-1}\left(\mathcal{F}(c) \times \mathcal{F}(x)\right)$$

where $\mathcal{F}$ is the Fourier transform. Because this operation can be simplified to a simple element wise vector multiplication, the matrix multiplication Cx can be computed in $O(n \log n)$ instead of $O(n^2)$.

Among the many applications of circulant matrices, matrix decomposition is one of the interest. In particular, Schmid et al. have shown in [26,29], that any complex matrix $A \in \mathbb{C}^{n \times n}$ can be decomposed into the product of diagonal and circulant matrices, as follows:

$$A = D^{(1)}C^{(1)}D^{(2)}C^{(2)} \dots D^{m}C^{m} = \prod_{i=1}^{m} D^{(i)}C^{(i)} \tag{1}$$

Later in [14], Huhtanen and Perämäki have demonstrated that choosing $m = n$ is sufficient to decompose any complex matrix $A \in \mathbb{C}^{n \times n}$. By [29], the result in Eq. 1 also holds for a real matrix $A \in \mathbb{R}^{n \times n}$, but the proof yields a much bigger value of m. However, the construction of [29] is far from optimal and it is likely that most real matrices can be decomposed into a reasonable number of factors. The authors of [25], made this conjecture, and they have leveraged the decomposition described in Eq. 1 in order to implement compact fully connected layers.

4 Compact Video Classification Architecture Using Circulant Matrices

Building on the decomposition presented in the previous section and the previous results obtained in [25], we now introduce a compact neural network architecture for video classification where dense matrices have been replaced by products of circulant and diagonal matrices.

4.1 Base Model

We demonstrate the benefit of circulant matrices using a base model which has been proposed by [24]. This architecture can be decomposed into three blocks of layers, as illustrated in Fig. 1. The first block of layers, composed of the Deep Bag-of-Frames embedding, is meant to process audio and video frames independently. The DBoF layer computes two embeddings: one for the audio and one for the video. In the next paragraph, we will only focus on describing the video embedding. (The audio embedding is computed in a very similar way.)

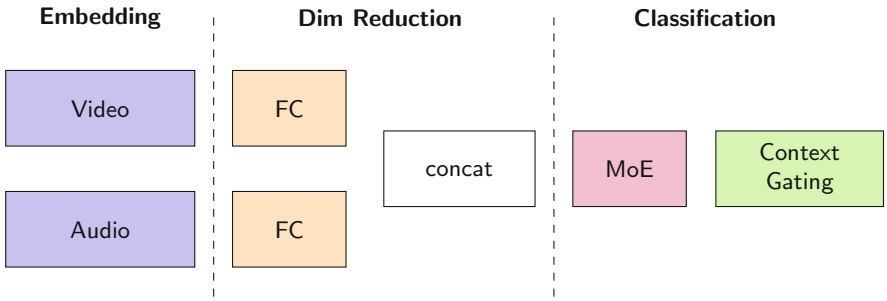

Fig. 1. This figure shows the architecture used for the experiences. The network samples at random video and audio frames from the input. The sample goes through an embedding layer and is reduced with a Fully Connected layer. The results are then concatenated and classified with a Mixture-of-Experts and Context Gating layer.

We represent a video V as a set of m frames $\{v_1, \ldots, v_m\}$ where each frame $v_i \in \mathbb{R}^k$ is a vector of visual features extracted from the frame image. In the context of the *YouTube-8M* competition, each v_i is a vector of 1024 visual features extracted using the last fully connected layer of an Inception network trained on ImageNet. The DBoF embedding layer then embed a video V into a vector v' drawn from a p dimensional vector space as follows.

$$v' = \max \{W v_i \mid v_i \in V\}$$

where W is a matrix in $\mathbb{R}^{p \times k}$ (learned) and max is the element-wise maximum operator. We typically choose $p > k$, (e.g. $p = 8192$). Note that because this formulation is based on set, it can process videos of different lengths (i.e., a different value of m).

A second block of layers reduces the dimensionality each embedding layer (audio and video), and merges the result into a single vector with using a simple concatenation operation. We chose to reduce the dimensionality of each embedding layer separately *before* the concatenation operation to avoid the concatenation of two high dimensional vectors.

Finally, the classification block uses a combination of Mixtures-of-Experts (MoE) and Context Gating to calculate the final probabilities. The Mixtures-of-Experts layer introduced in [18] and proposed for video classification in [2] are used to predict each label independently. It consists of a gating and experts networks which are concurrently learned. The gating network learns which experts to use for each label and the experts layers learn how to classify each label. The context gating operation was introduced in [24] and captures dependencies among features and re-weight probabilities based on the correlation of the labels. For example, it can capture the correlation of the labels *ski* and *snow* and re-adjust the probabilities accordingly.

Table 1 shows the shapes of the layers as well as the shapes of the weight matrices.

Table 1. This table shows the architecture of our base model with a DBoF Embedding and 150 frames sampled from the input. For more clarity, weights from batch normalization layers have been ignored. The −1 in the activation shapes corresponds to the batch size. The size of the MoE layers corresponds to the number of mixtures used.

Layer	Layer size	Activation shape	Weight matrix shape	#Weights
Video DBoF	8192	(−1, 150, 1024)	(1024, 8192)	8 388 608
Audio DBoF	4096	(−1, 150, 128)	(128, 4096)	524 288
Video FC	512	(−1, 8192)	(8192, 512)	4 194 304
Audio FC	512	(−1, 4096)	(4096, 512)	2 097 152
Concat	-	(−1, 1024)	-	-
MoE gating	3	(−1, 1024)	(1024, 19310)	19 773 440
MoE experts	2	(−1, 1024)	(1024, 15448)	15 818 752
Context gating	-	(−1, 3862)	(3862, 3862)	14 915 044

4.2 Robust Deep Bag-of-Frames Pooling Method

We propose a technique to extract more performance from the base model with DBoF embedding. The maximum pooling is sensitive to outliers and noise whereas the average pooling is more robust. We propose a method which consists in taking several samples of frames, applying the upsampling followed by the maximum pooling to these samples, and then averaging over all samples. More formally, assume a video contains m frames $v_1, \ldots, v_m \in \mathbb{R}^{1024}$. We first draw n random samples $S_1 \ldots S_n$ of size k from the set $\{v_1, \ldots, v_m\}$. The output of the robust-DBoF layer is:

$$\frac{1}{n} \sum_{i=1}^{n} \max \{v \times W : v \in S_i\}$$

Depending on n and k, this pooling method is a tradeoff between the max pooling and the average pooling. Thus, it is more robust to noise, as will be shown in the experiments section.

4.3 Compact Representation of the Base Model

In order to train this model in a compact form we build upon the work of [6] and use a more general framework presented by Eq. 1. The fully connected layers are then represented as follows:

$$h(x) = \phi \left(\left[\prod_{i=1}^{m} D^{(i)} C^{(i)} \right] x + b \right)$$

where the parameters of each matrix $D^{(i)}$ and $C^{(i)}$ are trained using a gradient based optimization algorithm, and m defines the number of factors. Increasing the value of m increases the number of trainable parameters and therefore the modeling capabilities of the layer. In our experiments, we chose the number of factors m empirically to achieve the best trade-off between model size and accuracy.

To measure the impact of the size of the model and its accuracy, we represent layers in their compact form independently. Given that circulant and diagonal matrices are square, we use concatenation and slicing to achieve the desired dimension. As such, with $m = 1$, the weight matrix (1024, 8192) of the video embedding is represented by a concatenation of 8 DC matrices and the weight matrix of size (8192, 512) is represented by a single DC matrix with shape (8192, 8192) and the resulting output is sliced at the 512 dimension. We denote layers in their classic form as *"Dense"* and layers represented with circulant and diagonal factors as *"Compact"*.

4.4 Leveraging Architectural Diversity

In order to benefit from architectural diversity, we also devise a single model architecture that combines different types of embedding layers. As we can see in Fig. 2, video and audio frames are processed by several embedding layers before being reduced by a series of compact fully connected layers. The output of the compact fc layers are then averaged, concatenated and fed into the final classification block. Figure 7 shows the result of different models given the number of parameters. The use of circulant matrices make us able to fit this model in gpu memory. For example, the diversity model with a NetVLAD embedding (cluster size of 256) and NetFV embedding (cluster size of 128) has 160 millions parameters (600 Mo) in the compact version and 728M (2.7 Go) in the dense version.

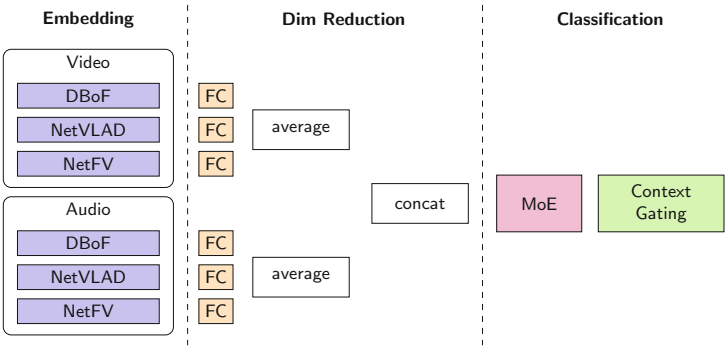

Fig. 2. This figure shows an evolution of the first architecture from Fig. 1 with several embeddings. This architecture is made to leverage the diversity of an Ensemble in a single model.

5 Experiments

In this section, we first evaluate the pooling technique proposed in Sect. 4.2. Then, we conduct experiments to evaluate the accuracy of our compact models. In particular, we investigate which layer benefits the most from a circulant representation and show that the decomposition presented in Sect. 3 performs better than the approach from [6] for the video classification problem. Finally, we compare all our models on a two dimensional size vs. accuracy scale in order to evaluate the trade-off between size and accuracy of each one of our models.

5.1 Experimental Setup

Dataset. All the experiments of this paper have been done in the context of the *2nd YouTube-8M Video Understanding Challenge* with the *YouTube-8M* dataset. We trained our models with the training set and 70% of the validation set which correspond to a total of 4 822 555 examples. We used the data augmentation technique proposed by [31] to virtually double the number of inputs. The method consists in splitting the videos into two equal parts. This approach is motivated by the observation that a human could easily label the video by watching either the beginning or the ending of the video.

All the code used in this experimental section is available online.[4]

Hyper-parameters. All our experiments are developed with TensorFlow Framework [1]. We trained our models with the CrossEntropy loss and used Adam optimizer with a 0.0002 learning rate and a 0.8 exponential decay every 4 million examples. All fully connected layers are composed of 512 units. DBoF, NetVLAD and NetFV are respectively 8192, 64 and 64 of cluster size for video frames and 4096, 32, 32 for audio frames. We used 4 mixtures for the MoE Layer. We used all 150 frames available and robust max pooling introduced in 4.2 for the DBoF embedding. In order to stabilize and accelerate the training, we used batch normalization before each non linear activation and gradient clipping.

Evaluation Metric. We used the GAP (Global Average Precision), as used in the *2nd YouTube-8M Video Understanding Challenge*, to compare our experiments. The GAP metric is defined as follows:

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

where N is the number of final predictions, $p(i)$ the precision, and $r(i)$ the recall. We limit our evaluation to 20 predictions for each video.

Hardware. All experiments have been realized on a cluster of 12 nodes. Each node has 160 POWER8 processor, 128 Go of RAM and 4 Nividia Titan P100.

[4] https://github.com/araujoalexandre/youtube8m-circulant.

5.2 Robust Deep Bag-of-Frames Pooling Method

We evaluate the performance of our Robust DBoF embedding. In accordance with the work from [2], we find that average pooling performs better than maximum pooling. Figure 3 shows that the proposed robust maximum pooling method outperforms both maximum and average pooling.

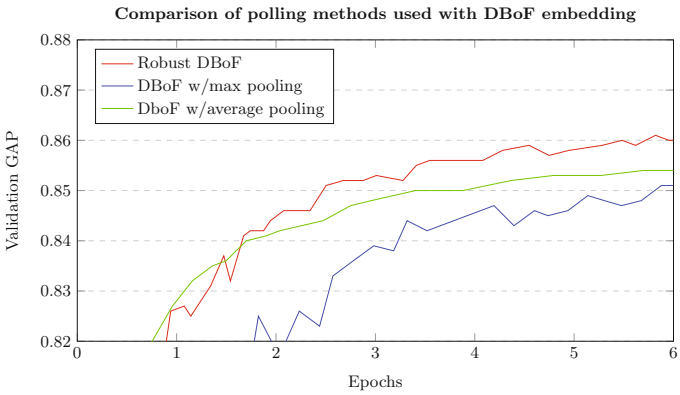

Fig. 3. This graphic shows the impact of *robust DBoF* (i.e. red line) with $n = 10$ and $k = 15$ on the Deep Bag-of-Frames embedding compared to max and average pooling. (Color figure online)

5.3 Impact of Circulant Matrices on Different Layers

This series of experiments aims at understanding the effect on compactness over different layers. Table 2 shows the result in terms of number of weights, size of the model (MB) and GAP. We also compute the compression ratio with respect to the dense model. The compact fully connected layer achieves a compression rate of 9.5 while having a very similar performance, whereas the compact DBoF and MoE achieve a higher compression rate at the expense of accuracy. Figure 4 shows that the model with a compact FC converges faster than the dense model. The model with a compact DBoF shows a big variance over the validation GAP which can be associated with a difficulty to train. The model with a compact MoE is more stable but at the expense of its performance. Another series of experiments investigates the effect of adding factors of the compact matrix DC (i.e. the parameters m specified in Sect. 4.3). Table 3 shows that there is no gain in accuracy even if the number of weights increases. It also shows that adding factors has an important effect on the speed of training. On the basis of this result, i.e. given the performance and compression ratio, we can consider that representing the fully connected layer of the base model in a compact fashion can be a good trade-off.

Table 2. This table shows the effect of the compactness of different layers. In these experiments, for speeding-up the training phase, we did not use the audio features and exploited only the video information.

Baseline model	#Weights	Size (MB)	Compress. Rate (%)	GAP@20	Diff.
Dense model	45 359 764	173	-	**0.846**	-
Compact DBoF	36 987 540	141	18.4	0.838	−0.008
Compact FC	41 181 844	157	9.2	0.845	**−0.001**
Compact MoE	12 668 504	48	72.0	0.805	−0.041

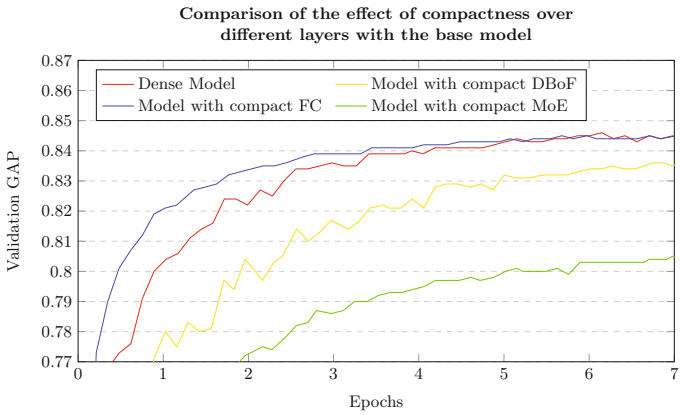

Fig. 4. Validation GAP according to the number of epochs for different compact models.

Table 3. This table shows the evolution of the number of parameters and the accuracy according to the number of factors. Despite the addition of degrees of freedom for the weight matrix of the fully connected layer, the model does not improve in performance. The column *#Examples/sec* shows the evolution of images per sec processed during the training of the model with a compact FC according to the number of factors.

#Factors	#Examples/sec	#Parameters in FC layer	Compress. Rate of FC layer (%)	GAP@20
1	1 052	12 288	99.8	0.861
3	858	73 728	98.8	0.861
6	568	147 456	97.6	0.859
Dense FC	1 007	6 291 456	-	0.861

5.4 Comparison with Related Works

Circulant matrices have been used in neural networks in [6]. They proposed to replace fully connected layers by a circulant and diagonal matrices where the circulant matrix is learned by a gradient based optimization algorithm and the diagonal matrix is random with values in $\{-1, 1\}$. We compare our more general

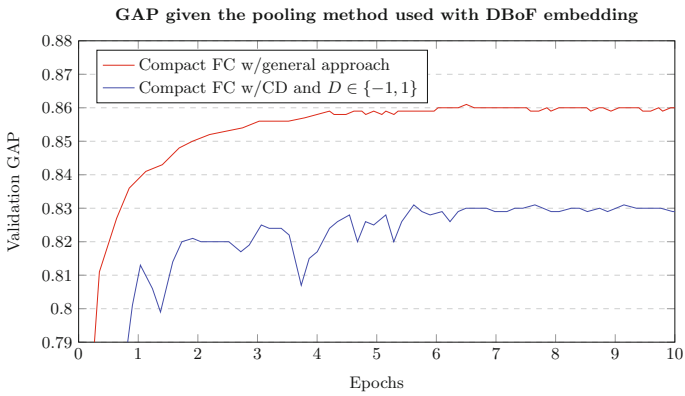

Fig. 5. This figure shows the GAP difference between the CD approach proposed in [6] and the more generalized DC approach from Sect. 4.3. Instead of having $D \in \{-1, +1\}$ fixed, the generalized approach allows D to be learned.

framework with their approach. Figure 5 shows the validation GAP according to the number of epochs of the base model with a compact fully connected layer implemented with both approach.

5.5 Compact Baseline Model with Different Embeddings

To compare the performance and the compression ratio we can expect, we consider different settings where the compact fully connected layer is used together with different embeddings. Figure 6 and Table 4 show the performance of the base model with DBoF, NetVLAD and NetFV embeddings with a *Dense* and *Compact* fully connected layer. Notice that we can get a bigger compression rate with NetVLAD and NetFV due to the fact that the output of the embedding is in a higher dimensional space which implies a larger weight matrix for the fully connected layer. Although the compression rate is higher, it is at the expense of the accuracy.

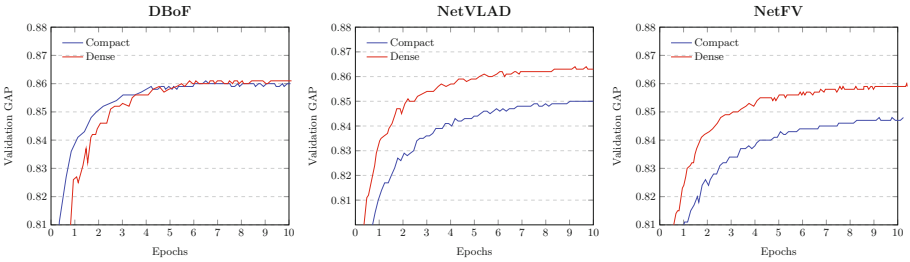

Fig. 6. The figures above show the validation GAP of compact and *Dense* fully connected layer with different embeddings according to the number of epochs.

Table 4. This table shows the impact of the compression of the fully connected layer of the model architecture shown in Fig. 1 with Audio and Video features vector and different types of embeddings. The variable compression rate is due to the different width of the output of the embedding.

Method	#Parameters	Size (MB)	Compress. Rate (%)	GAP@20
DBoF				
FC dense	65 795 732	251	-	0.861
FC circulant	59 528 852	227	9.56	0.861
NetVLAD				
FC dense	86 333 460	330	-	0.864
FC circulant	50 821 140	194	41.1	0.851
NetFisher				
FC dense	122 054 676	466	-	0.860
FC circulant	51 030 036	195	58.1	0.848

5.6 Model Size vs. Accuracy

To conclude our experimental evaluation, we compare all our models in terms of size and accuracy. The results are presented in Fig. 7.

As we can see in this figure, the most compact models are obtained with the circulant NetVLAD and NetFV. We can also see that the complex architectures described in Sect. 4.4 (DBoF + NetVLAD) achieve top performance but at the cost of a large number of weights. Finally, the best trade-off between size and accuracy is obtained using the DBoF embedding layer and achieves a GAP of 0.861 for only 60 millions weights.

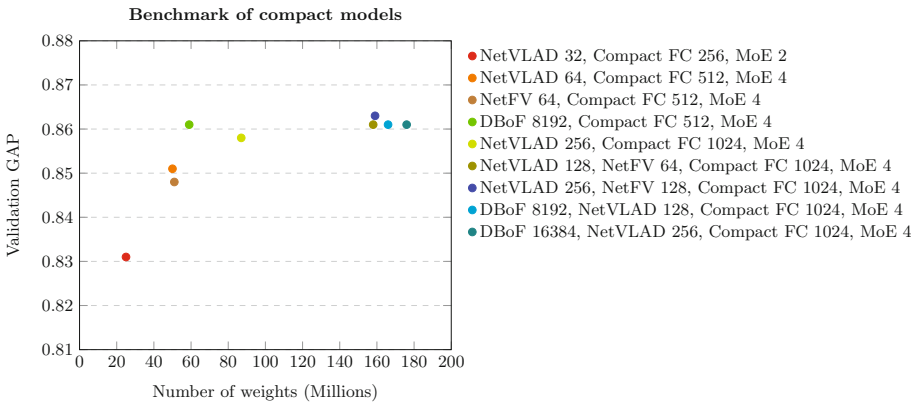

Fig. 7. Comparison between different models with compact fully connected layers.

6 Conclusion

In this paper, we demonstrated that circulant matrices can be a great tool to design compact neural network architectures for video classification tasks. We proposed a more general framework which improves the state of the art and conducted a series of experiments aiming at understanding the effect of compactness on different layers. Our experiments demonstrate that the best trade-off between size and accuracy is obtained using circulant DBoF embedding layers. We investigated a model with multiple embeddings to leverage the performance of an Ensemble but found it ineffective. The good performance of Ensemble models, i.e. why aggregating different distinct models performs better that incorporating all the diversity in a single architecture is still an open problem. Our future work will be devoted to address this challenging question and to pursue our effort to devise compact models achieving the same accuracy as larger one, and to study their theoretical properties.

Acknowledgement. This work was granted access to the OpenPOWER prototype from GENCI-IDRIS under the Preparatory Access AP010610510 made by GENCI. We would like to thank the staff of IDRIS who was really available for the duration of this work, Abdelmalek Lamine and Tahar Nguira, interns at Wavestone for their work on circulant matrices. Finally, we would also like to thank Wavestone to support this research.

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org/. Software available from tensorflow.org
2. Abu-El-Haija, S., et al.: YouTube-8M: a large-scale video classification benchmark. arXiv:1609.08675 (2016). https://arxiv.org/pdf/1609.08675v1.pdf
3. 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)
4. Buciluǎ, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 535–541. ACM (2006)
5. Chen, W., Wilson, J.T., Tyree, S., Weinberger, K.Q., Chen, Y.: Compressing neural networks with the hashing trick. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning, ICML 2015, vol. 37, pp. 2285–2294. JMLR.org (2015). http://dl.acm.org/citation.cfm?id=3045118.3045361
6. Cheng, Y., Yu, F.X., Feris, R.S., Kumar, S., Choudhary, A., Chang, S.F.: An exploration of parameter redundancy in deep networks with circulant projections. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 2857–2865, December 2015
7. Collins, M.D., Kohli, P.: Memory bounded deep convolutional networks. CoRR abs/1412.1442 (2014)

8. Courbariaux, M., Bengio, Y., David, J.P.: BinaryConnect: training deep neural networks with binary weights during propagations. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, NIPS 2015, vol. 2, pp. 3123–3131. MIT Press, Cambridge (2015). http://dl.acm.org/citation.cfm?id=2969442.2969588
9. Dai, B., Zhu, C., Guo, B., Wipf, D.: Compressing neural networks using the variational information bottleneck. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 80, pp. 1143–1152. PMLR, Stockholmsmässan, Stockholm, Sweden, 10–15 July 2018. http://proceedings.mlr.press/v80/dai18d.html
10. Denil, M., Shakibi, B., Dinh, L., Ranzato, M.A., de Freitas, N.: Predicting parameters in deep learning. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26, pp. 2148–2156. Curran Associates, Inc. (2013). http://papers.nips.cc/paper/5025-predicting-parameters-in-deep-learning.pdf
11. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning, ICML 2015, vol. 37, pp. 1737–1746. JMLR.org (2015). http://dl.acm.org/citation.cfm?id=3045118.3045303
12. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. In: International Conference on Learning Representations (ICLR) (2016)
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop (2015). http://arxiv.org/abs/1503.02531
14. Huhtanen, M., Perämäki, A.: Factoring matrices into the product of circulant and diagonal matrices. J. Fourier Anal. Appl. **21**(5), 1018–1033 (2015). https://doi.org/10.1007/s00041-015-9395-0
15. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. CoRR abs/1405.3866 (2014)
16. Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: CVPR 2010 - 23rd IEEE Conference on Computer Vision and Pattern Recognition, pp. 3304–3311. IEEE Computer Society, San Francisco, June 2010. https://doi.org/10.1109/CVPR.2010.5540039. https://hal.inria.fr/inria-00548637
17. Jiang, Y.G., Wu, Z., Wang, J., Xue, X., Chang, S.F.: Exploiting feature and class relationships in video categorization with regularized deep neural networks. IEEE Trans. Patt. Anal. Mach. Intell. **40**(2), 352–364 (2018). https://doi.org/10.1109/TPAMI.2017.2670560
18. Jordan, M.I., Jacobs, R.A.: Hierarchical mixtures of experts and the EM algorithm. In: Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan), vol. 2, pp. 1339–1344, October 1993. https://doi.org/10.1109/IJCNN.1993.716791
19. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1725–1732 (2014)
20. Li, F., et al.: Temporal modeling approaches for large-scale YouTube-8M video understanding. CoRR abs/1707.04555 (2017)

21. Lin, J., Rao, Y., Lu, J., Zhou, J.: Runtime neural pruning. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30, pp. 2181–2191. Curran Associates, Inc. (2017). http://papers.nips.cc/paper/6813-runtime-neural-pruning.pdf
22. Liu, B., Wang, M., Foroosh, H., Tappen, M., Penksy, M.: Sparse convolutional neural networks. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 806–814, June 2015. https://doi.org/10.1109/CVPR.2015.7298681
23. Mellempudi, N., Kundu, A., Mudigere, D., Das, D., Kaul, B., Dubey, P.: Ternary neural networks with fine-grained quantization. CoRR abs/1705.01462 (2017)
24. Miech, A., Laptev, I., Sivic, J.: Learnable pooling with context gating for video classification. CoRR abs/1706.06905 (2017)
25. Moczulski, M., Denil, M., Appleyard, J., de Freitas, N.: ACDC: a structured efficient linear layer. arXiv preprint arXiv:1511.05946 (2015)
26. Müller-Quade, J., Aagedal, H., Beth, T., Schmid, M.: Algorithmic design of diffractive optical systems for information processing. Phys. D Nonlinear Phenom. **120**(1–2), 196–205 (1998)
27. Perronnin, F., Dance, C.: Fisher kernels on visual vocabularies for image categorization. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8, June 2007. https://doi.org/10.1109/CVPR.2007.383266
28. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-net: ImageNet classification using binary convolutional neural networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016, Part IV. LNCS, vol. 9908, pp. 525–542. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_32
29. Schmid, M., Steinwandt, R., Müller-Quade, J., Rötteler, M., Beth, T.: Decomposing a matrix into circulant and diagonal factors. Linear Algebra Appl. **306**(1–3), 131–143 (2000)
30. Sindhwani, V., Sainath, T., Kumar, S.: Structured transforms for small-footprint deep learning. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 3088–3096. Curran Associates, Inc. (2015). http://papers.nips.cc/paper/5869-structured-transforms-for-small-footprint-deep-learning.pdf
31. Skalic, M., Pekalski, M., Pan, X.E.: Deep learning methods for efficient large scale video labeling. arXiv preprint arXiv:1706.04572 (2017)
32. Vybíral, J.: A variant of the johnson-lindenstrauss lemma for circulant matrices. J. Funct. Anal. **260**(4), 1096–1105 (2011). https://doi.org/10.1016/j.jfa.2010.11.014. http://www.sciencedirect.com/science/article/pii/S0022123610004507
33. Wang, H., Zhang, T., Wu, J.: The monkeytyping solution to the YouTube-8M video understanding challenge. CoRR abs/1706.05150 (2017)
34. Yang, Z., et al.: Deep fried convnets. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1476–1483, December 2015. https://doi.org/10.1109/ICCV.2015.173
35. Yu, X., Liu, T., Wang, X., Tao, D.: On compressing deep models by low rank and sparse decomposition. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 67–76, July 2017. https://doi.org/10.1109/CVPR.2017.15
36. Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: deep networks for video classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4694–4702 (2015)