Sympathy for the Details: Dense Trajectories and Hybrid Classification Architectures for Action Recognition

César Roberto de Souza[1,2], Adrien Gaidon[1], Eleonora Vig[3], Antonio Manuel López[2]

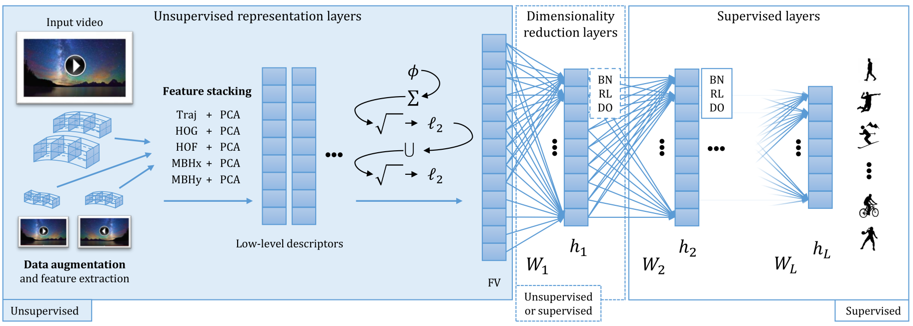
[1]Computer Vision Group, Xerox Research Center Europe, Meylan, France
[2]Centre de Visió per Computador, Universitat Autònoma de Barcelona, Bellaterra, Spain
[3]German Aerospace Center, Wessling, Germany
{cesar.desouza, adrien.gaidon}@xrce.xerox.com,
eleonora.vig@dlr.de, antonio@cvc.uab.es

Abstract. Action recognition in videos is a challenging task due to the complexity of the spatio-temporal patterns to model and the difficulty to acquire and learn on large quantities of video data. Deep learning, although a breakthrough for image classification and showing promise for videos, has still not clearly superseded action recognition methods using hand-crafted features, even when training on massive datasets. In this paper, we introduce hybrid video classification architectures based on carefully designed unsupervised representations of hand-crafted spatio-temporal features classified by supervised deep networks. As we show in our experiments on five popular benchmarks for action recognition, our hybrid model combines the best of both worlds: it is data efficient (trained on 150 to 10000 short clips) and yet improves significantly on the state of the art, including recent deep models trained on millions of manually labelled images and videos.

1 Introduction

Classifying human actions in real-world videos is an open research problem with many applications in multimedia, surveillance, and robotics [1]. Its complexity arises from the variability of imaging conditions, motion, appearance, context, and interactions with persons, objects, or the environment over different spatio-temporal extents. Current state-of-the-art algorithms for action recognition are based on statistical models learned from manually labeled videos. They belong to two main categories: models relying on features *hand-crafted* for action recognition (*e.g.,* [2,3,4,5,6,7,8,9,10]), or more recent end-to-end *deep architectures* (*e.g.,* [11,12,13,14,15,16,17,18,19,20,21,22]). These approaches have complementary strengths and weaknesses. Models based on hand-crafted features are data efficient, as they can easily incorporate structured prior knowledge (*e.g.,* the importance of motion boundaries along dense trajectories [2]), but their lack of flexibility may impede their robustness or modeling capacity. Deep models make fewer assumptions and are learned end-to-end from data (*e.g.,* using 3D-ConvNets [23]),

Accepted for publication in the 14th European Conference on Computer Vision (ECCV), Amsterdam, 2016.



Hand-crafted features are extracted along optical flow trajectories from original and generated videos. Those features are then normalized using RootSIFT [29], PCA-transformed, and augmented with their (x, y, t) coordinates, forming our low-level descriptors. The descriptors for each feature channel are then encoded (ϕ) as Fisher Vectors, separately aggregated (Σ) into a video-level representation, square-rooted, and ℓ_2-normalized. These representations are then concatenated ($\cup$) and renormalized. A dimensionality reduction layer is learned supervisedly or unsupervisedly. Supervised layers are followed by Batch-Normalization (BN) [30], ReLU (RL) non-linearities [31], and Dropout (DO) [32] during training. The last layer uses sigmoids (multi-label datasets) or softmax (multi-class datasets) non-linearities to produce action-label estimates.

Fig. 1: Our hybrid unsupervised and supervised deep multi-layer architecture.

but they rely on hand-crafted architectures and the acquisition of large manually labeled video datasets (*e.g.,* Sports-1M [12]), a costly and error-prone process that poses optimization, engineering, and infrastructure challenges.

Although deep learning for videos has recently made significant improvements (*e.g.,* [13,23,14]), models using hand-crafted features are the state of the art on many standard action recognition benchmarks (*e.g.,* [7,10,9]). These models are generally based on *improved Dense Trajectories* (iDT) [3,4] with Fisher Vector (FV) encoding [24,25]. Recent deep models for action recognition therefore combine their predictions with complementary ones from iDT-FV for better performance [23,26].

In this paper, we study an alternative strategy to *combine the best of both worlds via a single hybrid classification architecture* consisting in chaining sequentially the iDT hand-crafted features, the unsupervised FV representation, unsupervised or supervised dimensionality reduction, and a supervised deep network (*cf.* Figure 1). This family of models was shown by Perronnin and Larlus [27] to perform on par with the deep convolutional network of Krizhevsky *et al.* [28] for large scale image classification. We adapt this type of architecture differently for action recognition in videos with particular care for data efficiency.

Our **first contribution** consists in a careful design of the first *unsupervised* part of our hybrid architecture, which even with a simple SVM classifier is already on par with the state of the art. We experimentally observe that showing *sympathy for the details* (*e.g.,* spatio-temporal structure, normalization) and doing *data augmentation by feature stacking* (instead of duplicating training samples) are critical for performance, and that optimal design decisions generalize across datasets.

Our **second contribution** consists in a *data efficient hybrid architecture* combining unsupervised representation layers with a deep network of multiple fully connected layers. We show that *supervised mid-to-end learning of a dimensionality reduction layer together with non-linear classification layers* yields an excellent compromise between recognition accuracy, model complexity, and transferability of the model across datasets thanks to reduced risks of overfitting and modern optimization techniques.

The paper is organized as follows. Section 2 reviews the related works in action recognition. Section 3 presents the details of the first unsupervised part (based on iDT-FV) of our hybrid model, while Section 4 does so for the rest of the architecture and our learning algorithm. In Section 5 we report experimental conclusions from parametric studies and comparisons to the state of the art on five widely used action recognition datasets of different sizes. In particular, we show that our hybrid architecture improves significantly upon the current state of the art, including recent combinations of iDT-FV predictions with deep models trained on millions of images and videos.

2 Related Work

Existing action recognition approaches (*cf.* [1] for a recent survey) can be organized into four broad categories based on whether they involve *hand-crafted vs. deep-based* video features, and a *shallow vs. deep* classifier, as summarized in Table 1.

Table 1: Categorization of related recent action recognition methods

	SHALLOW CLASSIFIER	DEEP CLASSIFIER
HAND-CRAFTED FEATURES	[2,3,4,5,6,7,8,9,10]	[33], our method
DEEP-BASED FEATURES	[34,26,23,35]	[11,12,13,14,15,16,17,18,19,20,21,22]

Hand-crafted features, shallow classifier. A significant part of the progress on action recognition is driven by the development of local hand-crafted spatio-temporal features encoded as bag-of-words representations classified by "shallow" classifiers such as SVMs [2,3,4,5,6,7,8,9,10]. Most successful approaches use *improved Dense Trajectories (iDT)* [3] to aggregate local appearance and motion descriptors into a video-level representation through the Fisher Vector (FV) encoding [24,25]. Local descriptors such as HOG [36], HOF [37], and MBH [2] are extracted along dense point trajectories obtained from optical flow fields. There are several recent improvements to iDT, for instance, using motion compensation [5,38,39,6] and stacking of FVs to obtain a multi-layer encoding similar to mid-level representations [40]. To include global spatio-temporal location information, Wang *et al.* [5] compute FVs on a spatio-temporal pyramid (STP) [41] and use Spatial Fisher Vectors (SFV) [42]. Fernando *et al.* [10] model the global temporal evolution over the entire video using ranking machines learned on time-varying average FVs. Another recent improvement is the Multi-skIp Feature Stacking (MIFS) technique [7], which stacks features extracted at multiple frame-skips for better invariance to speed variations. An extensive study of the different steps of this general iDT pipeline and various feature fusion methods is provided in [8].

End-to-end learning: deep-based features, deep classifier. The seminal supervised deep learning approach of Krizhevsky *et al.* [28] has enabled impressive performance improvements on large scale image classification benchmarks, such as ImageNet [43], using Convolutional Neural Networks (CNN) [44]. Consequently, several approaches explored deep architectures for action recognition. While earlier works resorted to unsupervised learning of 3D spatio-temporal features [45], supervised end-to-end learning has recently gained popularity [11,12,13,14,15,16,17,18,19,20,21,22]. Karpathy *et al.* [12] studied several architectures and fusion schemes to extend 2D CNNs to the time domain. Although trained on the very large Sports-1M dataset, their 3D networks performed only marginally better than single-frame models. To overcome the difficulty of learning spatio-temporal features jointly, the Two-Stream architecture [13] is composed of two CNNs trained independently, one for appearance modeling on RGB input, and another for temporal modeling on stacked optical flow. Sun *et al.* [14] factorize 3D CNNs into learning 2D spatial kernels, followed by 1D temporal ones. Alternatively, other recent works use recurrent neural networks (RNN) in conjunction with CNNs to encode the temporal evolution of actions [16,17,19]. Overall, due to the difficulty of training 3D-CNNs and the need for vast amounts of training videos (*e.g.,* Sports-1M [12]), end-to-end methods report only marginal improvements over traditional baselines, and our experiments show that the iDT-FV often outperforms these approaches.

Deep-based features, shallow classifier. Several works [34,26,23,35] explore the encoding of general-purpose deep-learned features in combination with "shallow" classifiers, transferring ideas from the iDT-FV algorithm. Zha *et al.* [34] combine CNN features trained on ImageNet [43] with iDT features through a Kernel SVM. The TDD approach [26] extracts per-frame convolutional feature maps from two-stream CNN [13] and pools these over spatio-temporal cubes along extracted trajectories. Similar to [12], C3D [23] learns general-purpose features using a 3D-CNN, but the final action classifier is a linear SVM. Like end-to-end deep models, these methods rely on large datasets to learn generic useful features, which in practice perform on par or worse than iDT.

Hybrid architectures: hand-crafted features, deep classifier. There is little work on using unsupervised encodings of hand-crafted local features in combination with a deep classifier. In early work, Baccouche *et al.* [33] learn temporal dynamics of traditional per-frame SIFT-BOW features using a RNN. The method, coupled with camera motion features, improves on BoW-SVM for a small set of soccer videos.

Our work lies in this category, as it combines the strengths of iDT-FV encodings and supervised deep multi-layer non-linear classifiers. Our method is inspired by the recently proposed hybrid image classification architecture of Perronnin and Larlus [27], who stack several unsupervised FV-based and supervised layers. Their hybrid architecture shows significant improvements over the standard FV pipeline, closing the gap on [28], which suggests there is still much to learn about FV-based methods.

Our work investigates this type of hybrid architectures, with several noticeable differences: (i) FV is on par with the current state of the art for action recognition, (ii) iDT features contain many different appearance and motion descriptors, which also results in more diverse and higher-dimensional FV, (iii) most action recognition training sets are small due to the cost of labeling and processing videos, so overfitting and data efficiency are major concerns. In this context, we adopt different techniques from modern hand-crafted and deep models, and perform a wide architecture and parameter study showing conclusions regarding many design choices specific to action recognition.

3 Fisher Vectors in Action: From Baseline to State of the Art

We first recall the iDT approach of Wang & Schmid [3], then describe the improvements that can be stacked together to transform this strong baseline into a state-of-the-art method for action recognition. In particular, we propose a data augmentation by feature stacking method motivated by MIFS [7] and data augmentation for deep models.

3.1 Improved Dense Trajectories

Local spatio-temporal features. The iDT approach used in many state-of-the-art action recognition algorithms (*e.g.,* [3,4,5,7,40,8,10]) consists in first extracting dense trajectory video features [2] that efficiently capture appearance, motion, and spatio-temporal statistics. Trajectory shape (Traj) [2], HOG [36], HOF [37], and MBH [2] descriptors are extracted along trajectories obtained by median filtering dense optical flow. We extract dense trajectories from videos in the same way as in [3], applying RootSIFT normalization [29] (ℓ_1 normalization followed by square-rooting) to all descriptors. The number of dimensions in the Traj, HOG, HOF, MBHx, and MBHy descriptors are respectively 30, 96, 108, 96, and 96.

Unsupervised representation learning. Before classification, we combine the multiple trajectory descriptors in a single video-level representation by accumulating their Fisher Vector encodings (FV) [24,25], which was shown to be particularly effective for action recognition [5,46]. This high-dimensional representation is based on the gradient of a generative model, a Gaussian Mixture Model (GMM), learned in an *unsupervised* manner on a large set of trajectory descriptors in our case. Given a GMM with K Gaussians, each parameterized by its mixture weight w_k, mean vector μ_k, and standard deviation vector σ_k (assuming a diagonal covariance), the FV encoding of a trajectory descriptor $x \in \mathbb{R}^D$ is $\Phi(x) = [\phi_1(x), \ldots, \phi_K(x)] \in \mathbb{R}^{2KD}$, where:

$$\phi_k(x) = \left[\frac{\gamma(k)}{\sqrt{w_k}} \left(\frac{x - \mu_k}{\sigma_k} \right), \frac{\gamma(k)}{\sqrt{2w_k}} \left(\frac{(x - \mu_k)^2}{\sigma_k^2} - 1 \right) \right] \tag{1}$$

and $\gamma(k)$ denotes the soft assignment of descriptor x to Gaussian k. We use $K = 256$ Gaussians as a good compromise between accuracy and efficiency [3,4,5]. We randomly sample 256,000 trajectories from the pool of training videos, irrespectively of their labels, to learn one GMM per descriptor channel using 10 iterations of EM. Before learning the GMMs, we apply PCA to the descriptors, reducing their dimensionality by a factor of two. The reduced dimensions for the are, respectively, 15, 48, 54, 48, 48. After learning the GMMs, we extract FV encodings for all descriptors in each descriptor channel and combine these encodings into a per-channel, video-level representation using sum-pooling, *i.e.* by adding FVs together before normalization. In addition, we apply further post-processing and normalization steps, as discussed in the next subsection.

Supervised classification. When using a linear classification model, we use a linear SVM. As it is standard practice and in order to ensure comparability with previous works [3,7,47,26], we fix $C = 100$ unless stated otherwise and use *one-vs-rest* for multi-class and multi-label classification. This forms a strong baseline for action recognition, as shown by previous works [5,26] and confirmed in our experiments. We will now show how to make this baseline competitive with recent state-of-the-art methods.

3.2 Bag of Tricks for Bag-of-Words

Incorporating global spatio-temporal structure. Incorporating the spatio-temporal position of local features can improve the FV representation. We do not use spatio-temporal pyramids (STP) [41], as they significantly increase both the dimensionality of the representation and its variance [48]. Instead, we simply concatenate the PCA-transformed descriptors with their respective $(x, y, t) \in \mathbb{R}^3$ coordinates, as in [48,7]. We refer to this method as Spatio-Temporal Augmentation (STA). This approach is linked to the Spatial Fisher Vector (SFV) [42], a compact model related to soft-assign pyramids, in which the descriptor generative model is extended to explicitly accommodate the (x, y, t) coordinates of the local descriptors. When the SFV is created using Gaussian spatial models (*cf.* eq. 18 in [42]), the model becomes equivalent to a GMM created from augmented descriptors (assuming diagonal covariance matrices). Using STA, the dimensions for the descriptors before GMM estimation become 18, 51, 57, 51, 51. With 256 Gaussians, the dimension of the FVs generated for each descriptor channel are therefore 9,216, 26,112, 29,184, 26,112, and 26,112.

Normalization. We apply signed-square-rooting followed by ℓ_2 normalization, then concatenate all descriptor-specific FVs and reapply this same normalization, following [7]. The double normalization re-applies square rooting, and is thus equivalent to using a smaller power normalization [25], which improves action recognition performance [49].

Multi-Skip Feature Stacking (MIFS). MIFS [7] improves the robustness of FV to videos of different lengths by increasing the pool of features with frame-skipped versions of the same video. Standard iDT features are extracted from those frame-skipped versions and stacked together before descriptor encoding, decreasing the expectation and variance of the condition number [7,50,51] of the extracted feature matrices. We will now see that the mechanics of this technique can be expanded to other transformations.

3.3 Data Augmentation by Feature Stacking (DAFS)

Data augmentation is an important part of deep learning [26,52,53], but it is rarely used with hand-crafted features and shallow classifiers, particularly for action recognition where duplicating training examples can vastly increase the computational cost. Common data augmentation techniques for images include the use of random horizontal flipping [52,26], random cropping [52], and even automatically determined transformations [54]. For video classification, [9,10] duplicate the training set by mirroring.

Instead, we propose to generalize MIFS to arbitrary transformations, an approach we call *Data Augmentation by Feature Stacking* (DAFS). First, we extract features from multiple transformations of an input video (frame-skipping, mirroring, etc.) that do not change its semantic category. Second, we obtain a large feature matrix by stacking the obtained spatio-temporal features prior to encoding. Third, we encode the feature matrix, pool the resulted encodings, and apply the aforementioned normalization steps along this pipeline to obtain a *single augmented video-level representation*.

This approach yields a representation that simplifies the learning problem, as it can improve the condition number of the feature matrix further than just MIFS thanks to leveraging data augmentation techniques traditionally used for deep learning. In contrast to data augmentation for deep approaches, however, we build a single more robust and useful representation instead of duplicating training examples. Note also that DAFS is particularly suited to FV-based representation of videos as pooling FV from a much larger set of features decreases one of the sources of variance for FV [55].

After concatenation, the final representation for each video is 116,736-dimensional.

4 Hybrid Classification Architecture for Action Recognition

4.1 System Architecture

Our hybrid action recognition model combining FV with neural networks (cf. Fig. 1) starts with the previously described steps of our iDT-DAFS-FV pipeline, which can be seen as a set of *unsupervised layers*. The next part of our architecture consists of a set of L fully connected *supervised layers*, each comprising a dot-product followed by a non-linearity. Let h_0 denote the FV output from the last unsupervised layer in our hybrid architecture, h_{j-1} the input of layer $j \in \{1, ..., L\}$, $h_j = g(W_j h_{j-1})$ its output, with W_j the corresponding parameter matrix to be learned. We omit the biases from our equations for better clarity. For intermediate hidden layers we use the Rectified Linear Unit (ReLU) non-linearity [31] for g. For the final output layer we use different non-linearity functions depending on the task. For multi-class classification over c classes, we use the softmax function $g(z_i) = \exp(z_i)/\sum_{k=1}^{c} exp(z_k)$. For multi-label tasks we consider the sigmoid function $g(z_i) = 1/(1 + exp(-z_i))$.

Connecting the last unsupervised layer to the first supervised layer can result in a much higher number of weights in this section than in all other layers of the architecture. Since this might be an issue for small datasets due to the higher risk of overfitting, we study the impact of different ways to learn the weights of this *dimensionality reduction layer*: either with unsupervised learning (*e.g.,* using PCA as in [27]), or by learning a low-dimensional projection end-to-end with the next layers of the architecture.

4.2 Learning

Unsupervised layers. Our unsupervised layers are learned as described in Section 3.1. Namely, we learn one GMM of $K = 256$ Gaussians per descriptor channel using EM on a set of 256,000 trajectories randomly sampled from the pool of training videos.
Supervised layers. We use the standard cross-entropy between the network output $\hat{y}$ and the ground-truth label vectors y as loss function. For multi-class classification problems, we minimize the categorical cross-entropy cost function over all n samples:

$$C_{cat}(y, \hat{y}) = -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik} log(\hat{y}_{ik}), \tag{2}$$

whereas for multi-label problems we minimize the binary cross-entropy:

$$C_{bin}(y, \hat{y}) = -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik} log(\hat{y}_{ik}) - (1 - y_{ik}) log(1 - \hat{y}_{ik}). \tag{3}$$

Optimization. For parameter optimization we use the recently introduced Adam algorithm [56]. Since Adam automatically computes individual adaptive learning rates for the different parameters of our model, this alleviates the need for fine-tuning of the learning rate with a costly grid-search or similar methods.

Adam uses estimates of the first and second-order moments of the gradients in the update rule:

$$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{m_t}{(1-\beta_1^t)\sqrt{\frac{v_t}{1-\beta_2^t}} + \epsilon} \quad \text{where} \quad \begin{aligned} g_t &\leftarrow \nabla_\theta f(\theta_{t-1}) \\ m_t &\leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \\ v_t &\leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot {g_t}^2 \end{aligned} \tag{4}$$

and where $f(\theta)$ is the function with parameters θ to be optimized, t is the index of the current iteration, $m_0 = 0$, $v_0 = 0$, and β_1^t and β_2^t denotes β_1 and β_2 to the power of t, respectively. We use the default values for its parameters $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ proposed in [56] and implemented in Keras [57].

Batch normalization and regularization. During learning, we use batch normalization (BN) [30] and dropout (DO) [32]. Each BN layer is placed immediately before the ReLU non-linearity and parametrized by two vectors γ and β learned alongside each fully-connected layer. Given a training set $X = \{x_1, x_2, ..., x_n\}$ of n training samples, the transformation learned by BN for each input vector $x \in X$ is given by:

$$BN(x; \gamma, \beta) = \gamma \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \quad \text{where} \quad \begin{aligned} \mu_B &\leftarrow \tfrac{1}{n} \sum_{i=1}^{n} x_i \\ \sigma_B^2 &\leftarrow \tfrac{1}{n} \sum_{i=1}^{n} (x_i - \mu_B)^2 \end{aligned} \tag{5}$$

Together with DO, the operation performed by hidden layer j can now be expressed as $h_j = r \odot g(BN(W_j h_{j-1}; \gamma_j, \beta_j))$, where r is a vector of Bernoulli-distributed variables with probability p and $\odot$ denotes the element-wise product. We use the same DO rate p for all layers. The last output layer is not affected by this modification.

Dimensionality reduction layer. When unsupervised, we fix the weights of the dimensionality reduction layer from the projection matrices learned by PCA dimensionality reduction followed by whitening and ℓ_2 normalization [27]. When it is supervised, it is treated as the first fully-connected layer, to which we apply BN and DO as with the rest of the supervised layers. To explain our initialization strategy for the unsupervised case, let us denote the set of n mean-centered d-dimensional FVs for each sample in our dataset as a matrix $X \in \mathbb{R}^{d \times n}$. Recall that the goal of PCA projection is to find a $r \times d$ transformation matrix P, where $r \leqslant d$ on the form $Z = PX$ such that the rows of Z are uncorrelated, and therefore its $d \times d$ scatter matrix $S = ZZ^t$ is diagonal. In its primal form, this can be accomplished by the diagonalization of the $d \times d$ covariance matrix XX^t. However, when $n \ll d$ it can become computationally inefficient to compute XX^t explicitly. For this reason, we diagonalize the $n \times n$ Gram matrix $X^t X$ instead. By Eigendecomposition of $X^t X = V \Lambda V^t$ we can take $P = V^t X^t \Lambda^{-1/2}$, which also diagonalizes the scatter matrix S but is more efficient to compute [58,59].

To accommodate whitening, we set the weights of our first reduction layer to $W_1 = V^t X^t \Lambda^{-1} \sqrt{n}$ and keep them fixed during training.

Bagging. Since our first unsupervised layers can be fixed, we can train ensemble models and average their predictions very efficiently for bagging purposes [60,61,27] by caching the output of the unsupervised layers and reusing it in the subsequent models.

5 Experiments

We first describe the datasets used in our experiments, then provide a detailed analysis of the iDT-FV pipeline and our proposed improvements. Based on our observations, we then perform an ablative analysis of our proposed hybrid architecture. Finally, we study the transferability of our hybrid models, and compare to the state of the art.

5.1 Datasets

We use five publicly available and commonly used datasets for action recognition. We briefly describe these datasets and their evaluation protocols.

The **Hollywood2** [62] dataset contains 1,707 videos extracted from 69 Hollywood movies, distributed over 12 overlapping action classes. As one video can have multiple class labels, results are reported using the mean average precision (mAP).

The **HMDB-51** [63] dataset contains 6,849 videos distributed over 51 distinct action categories. Each class contains at least 101 videos and presents a high intra-class variability. The evaluation protocol is the average accuracy over three fixed splits [63].

The **UCF-101** [64] dataset contains 13,320 video clips distributed over 101 distinct classes. This is the same dataset used in the THUMOS'13 challenge [65]. The performance is again measured as the average accuracy on three fixed splits.

The **Olympics** [66] dataset contains 783 videos of athletes performing 16 different sport actions, with 50 sequences per class. Some actions include interactions with objects, such as *Throwing*, *Bowling*, and *Weightlifting*. Following [7,3], we report mAP over the train/test split released with the dataset.

The **High-Five** (a.k.a. TVHI) [67] dataset contains 300 videos from 23 different TV shows distributed over four different human interactions and a negative (no-interaction) class. As in [67,68,5,6], we report mAP for the positive classes (mAP+) using the train/test split provided by the dataset authors.

5.2 Detailed Study of Dense Trajectory Baselines for Action Recognition

Table 2 reports our results comparing the iDT baseline (Section 3.1), its improvements discussed in Section 3.2, and our proposed data augmentation strategy (Section 3.3).
Reproducibility. We first note that there are multiple differences in the iDT pipelines used across the literature. While [3] applies RootSIFT only on HOG, HOF, and MBH, in [7] this normalization is also applied to the Traj descriptor. While [3] includes Traj in their pipeline, [5] omits it. Additionally, person bounding boxes are used to ignore human motions when doing camera motion compensation in [5], but are not publicly available for all datasets. Therefore, we reimplemented the main baselines and compare our results to the officially published ones. As shown in Table 2, we successfully reproduce the original iDT results from [3] and [47], as well as the MIFS results of [7].
Improvements of iDT. Table 2 shows that double-normalization (DN) alone improves performance over iDT on most datasets without the help of STA. We show that STA gives comparable results to SFV+STP, as hypothesized in section 3.2. Given that STA and DN are both beneficial for performance, we combine them with our own method.

Table 2: Analysis of iDT baselines and several improvements

	UCF-101 %mAcc (s.d.)	HMDB-51 %mAcc (s.d.)	Hollywood2 %mAP	High-Five %mAP+ (s.d.)	Olympics %mAP
iDT [3]	84.8 [47]*†	57.2	64.3	-	91.1
Our reproduction	85.0 (1.32)*†	57.0 (0.78)	64.2	67.7 (1.90)	88.6
iDT+SFV+STP [5]	85.7*†	60.1*	66.8*	68.1*†	90.4*
Our reproduction	85.4 (1.27)*†	59.3 (0.80)*	67.1*	67.8 (3.78)*†	88.3*
iDT+STA+DN [7]	87.3	62.1	67.0	-	89.8
Our reproduction	87.3 (0.96)†	61.7 (0.90)	66.8	70.4 (1.63)	90.7
iDT+STA+MIFS+DN [7]	89.1	65.1	68.0	-	91.4
Our reproduction	89.2 (1.03)†	65.4 (0.46)	67.1	70.3 (1.84)	91.1
iDT+DN	86.3 (0.95)†	59.1 (0.45)	65.7	67.5 (2.27)	89.5
iDT+STA	86.0 (1.14)†	60.3 (1.32)	66.8	70.4 (1.96)	88.2
iDT+STA+DAFS+DN	**90.6 (0.91)**†	**67.8 (0.22)**	**69.1**	**71.0 (2.46)**	**92.8**

iDT: Improved Dense Trajectories; SFV: Spatial Fisher Vector; STP: Spatio-Temporal Pyramids; STA: Spatio-Temporal Augmentation; MIFS: Multi-skIp Feature Stacking; DN: Double-Normalization; DAFS: Data Augmentation Feature Stacking; *without Trajectory descriptor; †without Human Detector.

Data Augmentation by Feature Stacking (DAFS). Although more sophisticated transformations can be used, we found that combining a limited number of simple transformations already allows to significantly improve the iDT-based methods in conjunction with the aforementioned improvements, as shown in the "iDT+STA+DAFS+DN" line of Table 2. In practice, we generate on-the-fly 7 different versions for each video, considering the possible combinations of frame-skipping up to level 3 and horizontal flipping.
Fine tuned and non-linear SVMs. Attempting to improve our best results, we also performed experiments both fine-tuning C and also using a Gaussian kernel while fine-tuning γ. However, we found that those two sets of experiments did not lead to significant improvements. As DAFS already brings results competitive with the current state of the art, we set those results with fixed C as our current shallow baseline (FV-SVM). We will now incorporate those techniques in the first unsupervised layers of our hybrid models.

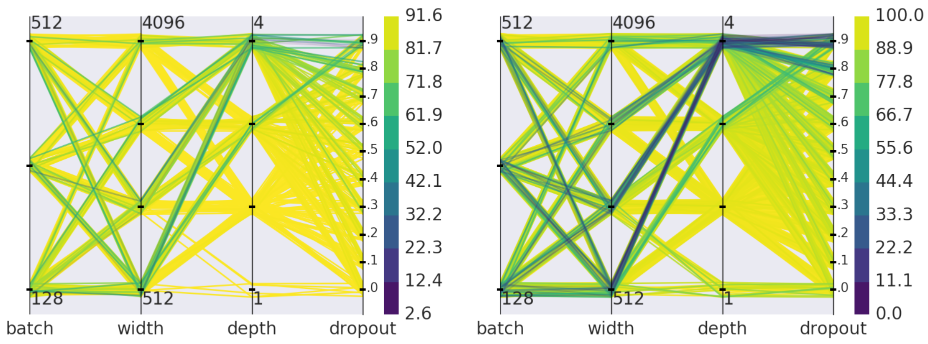
5.3 Analysis of Hybrid Models

In this section, we start from hybrid architectures with unsupervised dimensionality reduction learned by PCA. For UCF-101 (the largest dataset) we initialize W_1 with $r = 4096$ dimensions, whereas for all other datasets we use the number of dimensions responsible for 99% of the variance (yielding less dimensions than training samples).

We study the interactions between four parameters that can influence the performance of our hybrid models: the output dimension of the intermediate fully connected layers (*width*), the number of layers (*depth*), the dropout rate, and the mini-batch size of Adam (*batch*). We systematically evaluate all possible combinations and rank the architectures by the average relative improvement *w.r.t.* the best FV-SVM model. Training all 480 combinations for one split of UCF-101 can be accomplished in less than two days with a single Tesla K80 GPU. We report the top results in Table 3 and visualize all results using the parallel coordinates plot in Figure 1. Our observations are as follows.

Table 3: Top-5 best performing hybrid architectures with consistent improvements

Depth	Width	Batch	UCF-101 %mAcc	HMDB-51 %mAcc	Hollywood2 %mAP	High-Five %mAP+	Olympics %mAP	Relative Improv.
2	**4096**	**128**	91.6	68.1	72.6	**73.1**	**95.3**	**2.46%**
2	4096	256	91.6	67.8	72.5	72.9	95.3	2.27%
2	2048	128	91.5	68.0	72.7	72.7	94.8	2.21%
2	2048	256	91.4	67.9	72.7	72.5	95.0	2.18%
2	512	128	91.0	67.4	**73.0**	72.4	95.3	2.05%
1	-	-	**91.9**	**68.5**	70.4	71.9	93.5	1.28%
Best FV-SVM (*cf.* Tab. 2)			90.6	67.8	69.1	71.0	92.8	0.00%



(a) UCF-101 (b) All datasets (normalized)

Each line represents one combination of parameters and color indicates performance of our hybrid architectures with unsupervised dimensionality reduction. Depth 2 correlates with high-performing architectures, whereas a small width and a large depth is suboptimal.

Fig. 2: Parallel coordinates plots showing the impact of multiple parameters.

Unsupervised dimensionality reduction. Performing dimensionality reduction using the weight matrix from PCA is beneficial for all datasets, and using this layer alone, achieves 1.28% average improvement (Table 3, depth 1) upon our best SVM baseline.

Width. We consider networks with fully connected layers of size 512, 1024, 2048, and 4096. We find that a large width (4096) gives the best results in 4 of 5 datasets.

Depth. We consider hybrid architectures with depth between 1 and 4. Most well-performing models have depth 2 as shown in Figure 1, but one layer is enough for the big datasets.

Dropout rate. We consider dropout rates from 0 to 0.9. We find dropout to be dependent of both architecture and dataset. A high dropout rate significantly impairs classification results when combined with a small width and a large depth.

Mini-batch size. We consider mini-batch sizes of 128, 256, and 512. We find lower batch sizes to bring best results, with 128 being the more consistent across all datasets. We observed that large batch sizes were detrimental to networks with a small width.

Best configuration with unsupervised dimensionality reduction. We find the following parameters to work the best: small batch sizes, a large width, moderate depth, and dataset-dependent dropout rates. The most consistent improvements across datasets are with a network with batch-size 128, width 4096, and depth 2.

Table 4: Supervised dimensionality reduction hybrid architecture evaluation

Depth	Width	Batch	UCF-101 %mAcc (s.d.)	HMDB-51 %mAcc (s.d.)	Hollywood2 %mAP	High-Five %mAP+ (s.d.)	Olympics %mAP
1	1024	128	92.3 (0.77)	**69.4 (0.16)**	72.5	71.8 (1.37)	95.2
1	512	128	**92.3 (0.70)**	69.2 (0.09)	72.2	72.2 (1.14)	95.2
2	1024	128	91.9 (0.78)	68.8 (0.46)	71.8	72.0 (1.03)	94.8
2	512	128	92.1 (0.68)	69.1 (0.36)	70.8	71.9 (2.22)	94.2
Best unsup. (*cf.* Tab. 3)			91.9	68.5	**73.0**	**73.1**	**95.3**

Supervised dimensionality reduction. Our previous findings indicate that the dimensionality reduction layer can have a large influence on the overall classification results. Therefore, we investigate whether a *supervised* dimensionality reduction layer trained *mid-to-end* with the rest of the architecture could improve results further. Due to memory limitations imposed by the higher number of weights to be learned between our 116K-dimensional input FV representation and the intermediate fully-connected layers, we decrease the maximum network width to 1024. In spite of this limitation, our results in Table 4 show that much smaller hybrid architectures with supervised dimensionality reduction improve (on the larger UCF-101 and HMDB-51 datasets) or maintain (on the other smaller datasets) recognition performance.
Comparison to hybrid models for image recognition. Our experimental conclusions and optimal model differ from [27], both on unsupervised and supervised learning details (*e.g.,* dropout rate, batch size, learning algorithm), and in the usefulness of a supervised dimensionality reduction layer trained mid-to-end (not explored in [27]).

5.4 Transferability of Hybrid Models

In this section, we study whether the first layers of our architecture can be transferred across datasets. As a reference point, we use the first split of UCF-101 to create a base model and transfer elements from it to other datasets. We chose UCF-101 for the following reasons: it is the largest dataset, has the largest diversity in number of actions, and contains multiple categories of actions, including human-object interaction, human-human interaction, body-motion interaction, and practicing sports.
Unsupervised representation layers. We start by replacing the dataset-specific GMMs with the GMMs from the base model. Our results in the second row of Table 5 show that the transferred GMMs give similar performance to the ones using dataset-specific GMMs. This, therefore, greatly simplifies the task of learning a new model for a new dataset. We keep the transferred GMMs fixed in the next experiments.
Unsupervised dimensionality reduction layer. Instead of configuring the unsupervised dimensionality reduction layer with weights from the PCA learned on its own dataset, we configure it with the weights learned in UCF-101. Our results are in the third row of Table 5. This time we observe a different behavior: for Hollywood2 and HMDB-51, the best models were found without transfer, whereas for Olympics it did not have any measurable impact. However, transferring PCA weights brings significant improvement in High-Five. One of the reasons for this improvement is the evidently smaller training set size of High-Five (150 samples) in contrast to other datasets. The fact that the

Table 5: Transferability experiments involving unsupervised dimensionality reduction

Representation Layers	Reduction Layer	Supervised Layers	HMDB-51 %mAcc (s.d.)	Hollywood2 %mAP	High-Five %mAP+ (s.d.)	Olympics %mAP
own	own	own	68.0 (0.65)	**72.6**	73.1 (1.01)	95.3
UCF	own	own	**68.0 (0.40)**	72.4	73.7 (1.76)	94.2
UCF	UCF	own	66.5 (0.88)	70.0	**76.3 (0.96)**	94.0
UCF	UCF	UCF	66.8 (0.36)	69.7	71.8 (0.12)	**96.0**

Table 6: Transferability experiments involving supervised dimensionality reduction

Representation Layers	Supervised Layers	HMDB-51 %mAcc (s.d.)	Hollywood2 %mAP	High-Five %mAP+ (s.d.)	Olympics %mAP
own	own	69.2 (0.09)	72.2	72.2 (1.14)	95.2
UCF	own	69.4 (0.16)	**72.5**	71.8 (1.37)	95.2
UCF	UCF	**69.6 (0.36)**	72.2	**73.2 (1.89)**	**96.3**

improvement becomes less visible as the number of samples in each dataset increases (before eventually degrading performance) indicates there is a threshold below which transferring starts to be beneficial (around a few hundred training videos).

Supervised layers after unsupervised reduction. We also study the transferability of further layers in our architecture, after the unsupervised dimensionality reduction transfer. We take the base model learned in the first split of UCF-101, remove its last classification layer, re-insert a classification layer with the same number of classes as the target dataset, and fine-tune this new model in the target dataset, using an order of magnitude lower learning rate. The results can be seen in the last row of Table 5. The same behavior is observed for HMDB-51 and Hollywood2. However, we notice a decrease in performance for High-Five and a performance increase for Olympics. We attribute this to the presence of many sports-related classes in UCF-101.

Mid-to-end reduction and supervised layers. Finally, we study whether the architecture with supervised dimensionality reduction layer transfers across datasets, as we did for the unsupervised layers. We again replace the last classification layer from the corresponding model learned on the first split of UCF-101, and fine-tune the whole architecture on the target dataset. Our results in the second and third rows of Table 6 show that transferring this architecture brings improvements for Olympics and HMDB-51, but performs worse than transferring unsupervised layers only on High-Five.

5.5 Comparison to the State of the Art

In this section, we compare our best models found previously to the state of the art.

Best models. For UCF-101, the most effective model leverages its large training set using supervised dimensionality reduction (*cf.* Table 4). For HMDB-51 and Olympics, the best models result from transferring the supervised dimensionality reduction models from the related UCF-101 dataset (*cf.* Table 6). Due to its specificity, the best architecture for Hollywood2 is based on unsupervised dimensionality reduction learned on its own data (*cf.* Table 3), although there are similarly-performing end-to-end transferred models (*cf.* Table 6). For High-Five, the best model is obtained by transferring the unsupervised dimensionality reduction models from UCF-101 (*cf.* Table 5).

Table 7: Comparison against the state of the art in action recognition

	Method	UCF-101 %mAcc (s.d.)	HMDB-51 %mAcc (s.d.)	Hollywood2 %mAP	High-Five %mAP+ (s.d.)	Olympics %mAP
HANDCRAFTED	iDT+FV[3]	84.8 [47]	57.2	64.3	-	91.1
	SDT-ATEP[6]	-	41.3	54.4	62.4	85.5
	iDT+FM[8]	87.9	61.1	-	-	-
	RCS[9]	-	-	73.6	71.1	-
	iDT+SFV+STP[5]	86.0	60.1	66.8	69.4	90.4
	iDT+MIFS[7]	89.1	65.1	68.0	-	91.4
	VideoDarwin[10]	-	61.6	69.6	-	-
	VideoDarwin+HF+iDT[10]	-	63.7	**73.7**	-	-
DEEP-BASED	2S-CNN[13]^{IN}	88.0	59.4	-	-	-
	2S-CNN+Pool[17]^{IN}	88.2	-	-	-	-
	2S-CNN+LSTM[17]^{IN}	88.6	-	-	-	-
	Objects+Motion(R*)[69]^{IN}	88.5	61.4	66.4	-	-
	Comp-LSTM[18]^{ID}	84.3	44.0	-	-	-
	C3D+SVM[23]^{S1M,ID}	85.2	-	-	-	-
	FSTCN[14]^{IN}	88.1	59.1	-	-	-
HYBRID	iDT+StackFV[40]	-	66.8	-	-	-
	TDD[26]^{IN}	90.3	63.2	-	-	-
	TDD+iDT[26]^{IN}	91.5	65.9	-	-	-
	CNN-hid6[34]^{S1M}	79.3	-	-	-	-
	CNN-hid6+iDT[34]^{S1M}	89.6	-	-	-	-
	C3D+iDT+SVM[23]^{S1M,ID}	90.4	-	-	-	-
	Best from state-of-the-art	91.5 [26]	66.8 [40]	**73.7** [10]	71.1 [9]	91.4 [7]
	Our best FV+SVM	90.6 (0.91)	67.8 (0.22)	69.1	71.0 (2.46)	92.8
	Our best hybrid	**92.5 (0.73)**	**70.4 (0.97)**	72.6	**76.7 (0.39)**	**96.7**

Methods are organized by category (cf. Table 1) and sorted in chronological order in each block. Our hybrid models improve upon the state of the art, and our handcrafted-shallow FV-SVM improves upon competing end-to-end architectures relying on external data sources (IN: uses ImageNet, S1M: uses Sports-1M, ID: uses private internal data).

Bagging. As it is standard practice [27], we take the best models and perform bagging with 8 models initialized with distinct random initializations. This improves results by around one point on average, and our final results are in Table 7.

Discussion. In contrast to [27], our models outperform the state of the art, including methods trained on massive labeled datasets like ImageNet or Sports-1M, confirming both the excellent performance and the data efficiency of our approach. Table 8 illustrates some failure cases of our methods. Confusion matrices and precision-recall curves for all datasets are available in the supplementary material for fine-grained analysis.

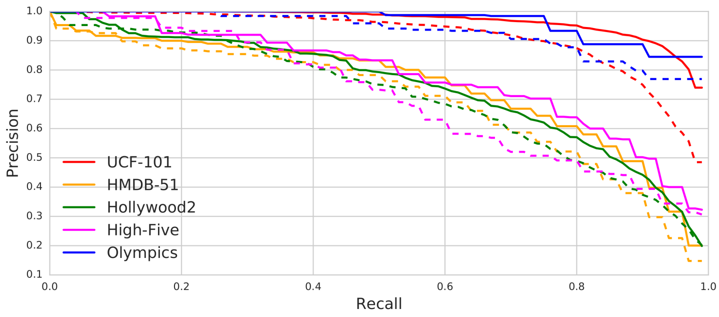
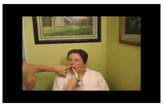
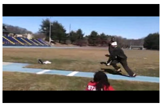
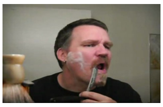
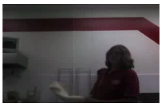
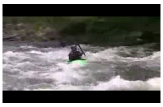
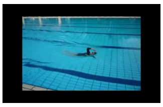
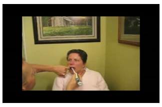
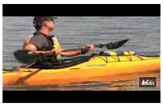
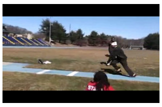
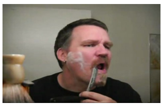
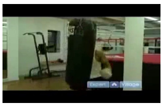
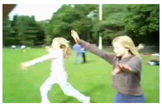
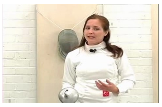
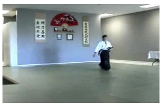
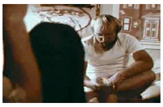
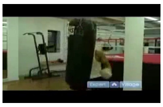
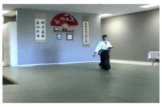
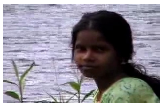
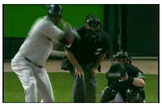
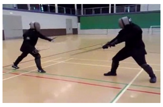
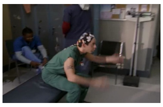
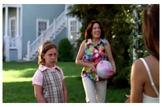
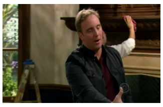
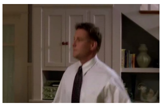
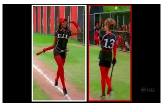
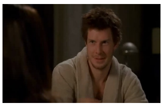
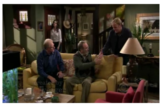
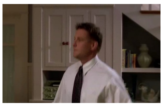
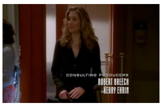
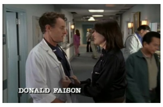
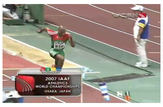
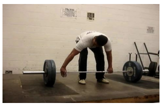
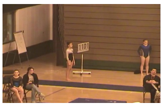
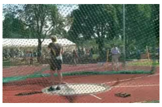
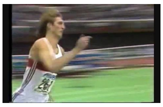
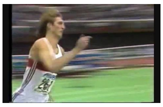
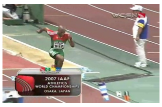
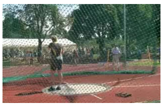
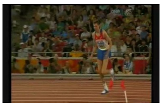
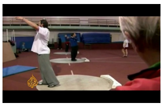


Fig. 3: Average Precision-Recall curves. Dashed: best FV-SVM, Full: best hybrid.

Table 8: Top-5 most confused classes for our best FV-SVM and Hybrid models

	Top-5 Confusions	#1	#2	#3	#4	#5
UCF-101	Hybrid	 P: ShavingBeard GT: BrushingTeeth	 P: FrisbeeCatch GT: LongJump	 P: ApplyLipstick GT: ShavingBeard	 P: Nunchucks GT: PizzaTossing	 P: Rafting GT: Kayaking
	FV-SVM	 P: BreastStroke GT: FrontCrawl	 P: ShavingBeard GT: BrushingTeeth	 P: Rafting GT: Kayaking	 P: FrisbeeCatch GT: LongJump	 P: ApplyLipstick GT: ShavingBeard
HMDB-51	Hybrid	 P: sword GT: punch	 P: flic_flac GT: cartwheel	 P: draw_sword GT: sword_exercise	 P: sword_exercise GT: draw_sword	 P: drink GT: eat
	FV-SVM	 P: sword GT: punch	 P: sword_exercise GT: draw_sword	 P: chew GT: smile	 P: throw GT: swing_baseball	 P: fencing GT: sword
High-Five	Hybrid	 P: negative GT: highFive	 P: negative GT: handShake	 P: hug GT: handShake	 P: hug GT: kiss	 P: handShake GT: highFive
	FV-SVM	 P: negative GT: highFive	 P: negative GT: handShake	 P: hug GT: kiss	 P: hug GT: handShake	 P: negative GT: kiss
Olympics	Hybrid	 P: long_jump GT: triple_jump	 P: clean_and_jerk GT: snatch	 P: high_jump GT: vault	 P: discus_throw GT: hammer_throw	 P: vault GT: high_jump
	FV-SVM	 P: vault GT: high_jump	 P: long_jump GT: triple_jump	 P: discus_throw GT: hammer_throw	 P: pole_vault GT: high_jump	 P: bowling GT: shot_put

6 Conclusion

We investigate hybrid architectures for action recognition, effectively combining hand-crafted spatio-temporal features, unsupervised representation learning based on the FV encoding, and deep neural networks. In addition to paying attention to important details like normalization and spatio-temporal structure, we integrate data augmentation at the feature level, end-to-end supervised dimensionality reduction, and modern optimization and regularization techniques. We perform an extensive experimental analysis on a variety of datasets, showing that our hybrid architecture yields data efficient, transferable models of small size that yet outperform much more complex deep architectures trained end-to-end on millions of images and videos. We believe our results open interesting new perspectives to design even more advanced hybrid models, *e.g.,* using recurrent neural networks, targeting better accuracy, data efficiency, and transferability.
Acknowledgements. Antonio M. Lopez is supported by the Spanish MICINN project TRA2014-57088-C2-1-R, and by the Secretaria d'Universitats i Recerca del Departament d'Economia i Coneixement de la Generalitat de Catalunya (2014-SGR-1506).

References

1. Vrigkas, M., Nikou, C., Kakadiaris, I.: A review of human activity recognition methods. Frontiers in Robotics and AI **2** (2015) 1–28
2. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Action recognition by dense trajectories. In: CVPR. (2011)
3. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: ICCV. (2013)
4. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Dense trajectories and motion boundary descriptors for action recognition. IJCV **103** (2013) 60–79
5. Wang, H., Oneata, D., Verbeek, J., Schmid, C.: A robust and efficient video representation for action recognition. IJCV (July 2015) 1–20
6. Gaidon, A., Harchaoui, Z., Schmid, C.: Activity representation with motion hierarchies. IJCV **107** (2014) 219–238
7. Lan, Z., Lin, M., Li, X., Hauptmann, A.G., Raj, B.: Beyond gaussian pyramid : Multi-skip feature stacking for action recognition. In: CVPR. (2015)
8. Peng, X., Wang, L., Wang, X., Qiao, Y.: Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. arXiv1405.4506 (May 2014)
9. Hoai, M., Zisserman, A.: Improving human action recognition using score distribution and ranking. In: ACCV. (2014)
10. Fernando, B., Gavves, E., Oramas, M., Ghodrati, A., Tuytelaars, T., Leuven, K.U.: Modeling video evolution for action recognition. In: CVPR. (2015)
11. Ji, S., Yang, M., Yu, K., Xu, W.: 3D convolutional neural networks for human action recognition. T-PAMI **35** (2013) 221–31
12. Karpathy, A., Leung, T.: Large-scale video classification with convolutional neural networks. In: CVPR. (2014)
13. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: NIPS. (2014)
14. Sun, L., Jia, K., Yeung, D.Y., Shi, B.E.: Human action recognition using factorized spatio-temporal convolutional networks. In: ICCV. (2015)
15. Wu, Z., Jiang, Y.g., Wang, X., Ye, H., Xue, X., Wang, J.: Fusing multi-stream deep networks for video classification. arXiv:1509.06086 (September 2015)

16. Donahue, J., Hendricks, L.A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: CVPR. (2015)
17. Ng, J.Y.H., Hausknecht, M.J., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: CVPR. (2015)
18. Srivastava, N., Mansimov, E., Salakhutdinov, R.: Unsupervised learning of video representations using LSTMs. arXiv:1502.04681 (March 2015)
19. Ballas, N., Yao, L., Pal, C., Courville, A.C.: Delving deeper into convolutional networks for learning video representations. In: ICLR. (2013)
20. Gan, C., Wang, N., Yang, Y., Yeung, D.Y., Hauptmann, A.G.: DevNet: A Deep Event Network for multimedia event detection and evidence recounting. In: CVPR. (2015)
21. Wang, X., Farhadi, A., Gupta, A.: Actions $\sim$ Transformations. In: CVPR. (2015)
22. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional Two-Stream Network Fusion for Video Action Recognition. In: CVPR. (2016)
23. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: CVPR. (2014)
24. Perronnin, F., Dance, C.: Fisher kernels on visual vocabularies for image categorization. In: CVPR. (2007)
25. Perronnin, F., Sánchez, J., Mensink, T.: Improving the Fisher kernel for large-scale image classification. In: ECCV. (2010)
26. Wang, L., Qiao, Y., Tang, X.: Action recognition with trajectory-pooled deep-convolutional descriptors. In: CVPR. (2015)
27. Perronnin, F., Larlus, D.: Fisher vectors meet neural networks: A hybrid classification architecture. In: CVPR. (2015)
28. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012)
29. Arandjelovi, R., Zisserman, A.: Three things everyone should know to improve object retrieval. In: CVPR. (2012)
30. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. (2015)
31. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML. (2010)
32. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal on Machine Learning Research **15** (2014) 1929–1958
33. Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A.: Action classification in soccer videos with long short-term memory recurrent neural networks. In: Proceedings of the International Conference on Artificial Neural Networks. (2010)
34. Zha, S., Luisier, F., Andrews, W., Srivastava, N., Salakhutdinov, R.: Exploiting image-trained CNN architectures for unconstrained video classification. In: BMVC. (2015)
35. Wu, Z., Wang, X., Jiang, Y., Ye, H., Xue, X.: Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In: ACM MM. (2015)
36. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005)
37. Dalal, N., Triggs, B., Schmid, C.: Human detection using oriented histograms of flow and appearance. In: ECCV. (2006)
38. Gaidon, A., Harchaoui, Z., Schmid, C.: Recognizing activities with cluster-trees of tracklets. In: BMVC. (2012)
39. Jain, M., Jegou, H., Bouthemy, P.: Better exploiting motion for better action recognition. In: CVPR. (2013)

40. Peng, X., Zou, C., Qiao, Y., Peng, Q.: Action recognition with stacked fisher vectors. In: ECCV. (2014)
41. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR. (2008)
42. Krapac, J., Verbeek, J., Jurie, F.: Modeling spatial layout with Fisher vectors for image categorization. In: ICCV. (2011)
43. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV **115**(3) (2015) 211–252
44. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Handwritten digit recognition with a back-propagation network. In: NIPS. (1989)
45. Le, Q.V., Zou, W.Y., Yeung, S.Y., Ng, A.Y.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: CVPR. (2011)
46. Chatfield, K., Lempitsky, V., Vedaldi, A., Zisserman, A.: The devil is in the details : an evaluation of recent feature encoding methods. In: BMVC. (2011)
47. Wang, H., Schmid, C.: Lear-inria submission for the thumos workshop. In: ICCV workshop on action recognition with a large number of classes. (2013)
48. Sanchez, J., Perronnin, F., De Campos, T.: Modeling the Spatial Layout of Images Beyond Spatial Pyramids. Pattern Recognition Letters **33**(16) (2012) 2216–2223
49. Narayan, S., Ramakrishnan, K.R.: Hyper-Fisher Vectors for Action Recognition. arXiv1509.08439 (2015)
50. Poggio, T., Smale, S.: The mathematics of learning: dealing with data. Notices of the American Mathematical Society **50**(May 2003) (2003) 537–544
51. Bousquet, O., Elisseeff, A.: Stability and Generalization. Journal on Machine Learning Research **2** (2002) 499–526
52. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the Devil in the Details: Delving Deep into Convolutional Nets. In: BMVC. (2014)
53. Konda, K., Bouthillier, X., Memisevic, R., Vincent, P.: Dropout as data augmentation. In: ICLR. (2015)
54. Paulin, M., Harchaoui, Z., Perronnin, F., Paulin, M.: Transformation pursuit for image classification. In: CVPR. (2014)
55. Boureau, Y.L., Ponce, J., LeCun, Y.: A theoretical analysis of feature pooling in visual recognition. In: ICML. (2010)
56. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv1412.6980 (December 2014)
57. Chollet, F.: Keras: Deep learning library for Theano and TensorFlow (2015)
58. Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., Schmid, C.: Aggregating local image descriptors into compact codes. T-PAMI **34** (2012) 1704–1716
59. Bishop, C.M.: Pattern Recognition and Machine Learning. (2006)
60. Maclin, R., Opitz, D.: An empirical evaluation of bagging and boosting. In: AAAI. (1997)
61. Zhou, Z.H.H., Wu, J., Tang, W.: Ensembling neural networks: Many could be better than all. Artificial Intelligence **137** (2002) 239–263
62. Marszalek, M., Laptev, I., Schmid, C.: Actions in context. In: CVPR. (2009)
63. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: a large video database for human motion recognition. In: ICCV. (2011)
64. Soomro, K., Zamir, A.R., Shah, M.: UCF101: A dataset of 101 human actions classes from videos in the wild. arXiv:1212.0402 (December 2012)
65. Jiang, Y.G., Liu, J., Roshan Zamir, a., Laptev, I., Piccardi, M., Shah, M., Sukthankar, R.: THUMOS Challenge: Action Recognition with a Large Number of Classes (2013)
66. Niebles, J.C., Chen, C.W., Fei-Fei, L.: Modeling temporal structure of decomposable motion segments for activity classification. In: ECCV. (2010)

67. Patron-Perez, A., Marszalek, M., Zisserman, A., Reid, I.: High Five: Recognising human interactions in TV shows. In: BMVC. (2010)
68. Patron-Perez, A., Marszalek, M., Reid, I., Zisserman, A.: Structured learning of human interaction in TV shows. T-PAMI **34** (2012) 2441–2453
69. Jain, M., Van Gemert, J., Snoek, C.G.M.: What do 15,000 object categories tell us about classifying and localizing actions? In: CVPR. (2015)
70. Few, S.: Multivariate analysis using parallel coordinates. Perceptual Edge (2006) 1–9
71. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500) (2000) 2323–2326
72. Gosselin, P.H., Murray, N., Jégou, H., Perronnin, F.: Revisiting the Fisher Vector for fine-grained classification. Pattern Recognition Letters **49** (2014) 92–98

Sympathy for the Details: Dense Trajectories and Hybrid Classification Architectures for Action Recognition
Supplementary material

César Roberto de Souza[1,2], Adrien Gaidon[1], Eleonora Vig[3], Antonio Manuel López[2]

[1]Computer Vision Group, Xerox Research Center Europe, Meylan, France
[2]Centre de Visió per Computador, Universitat Autònoma de Barcelona, Bellaterra, Spain
[3]German Aerospace Center, Wessling, Germany
{cesar.desouza, adrien.gaidon}@xrce.xerox.com,
eleonora.vig@dlr.de, antonio@cvc.uab.es

1 Introduction

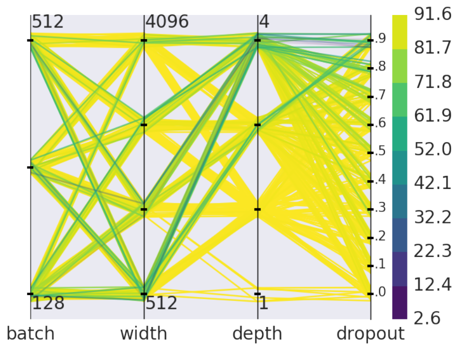
This material provides additional details regarding our submission to ECCV16. It presents graphs and plots that helped guide our experiments and consolidate our findings, but that were tangential to the main objectives of the submission. In particular, we provide the complete parallel coordinates plots for all datasets considered (whereas the submission shows only UCF-101 and overall combined results) and the confusion matrices and precision-recall plots detailing the failure and improvement cases between our hybrid models and our Fisher Vector with Support Vector Machines (FV-SVM) baseline.

2 Hybrid Unsupervised and Supervised Architectures

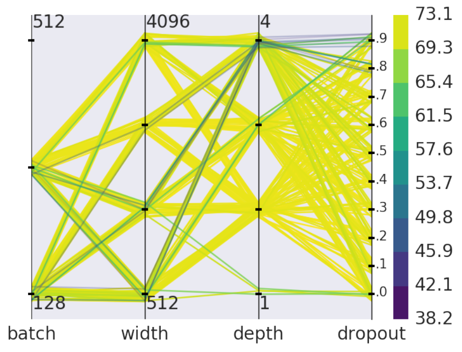
We present per-dataset parallel coordinates plots showing all the explored hybrid architectures with *unsupervised* dimensionality reduction, expanding Figure 2 of the submitted paper. The complete parallel coordinates plots for all datasets can be seen in Figure 1 and are explained below.

Explored architectures. Our parametric study, whose overall conclusion is described in Section 5.3 of the main submission, evaluates *480* architectures *for each dataset*, corresponding to three batch sizes $(128, 256, 512)$, four widths $(512, 1024, 2048, 4096)$, four depths $(1, 2, 3, 4)$ and ten dropout rates $(0.0, 0.1, ..., 0.9)$. We recall that by *depth* we mean number of network layers, and by *width* we mean dimension of the intermediate fully connected layers (*cf.* main text).

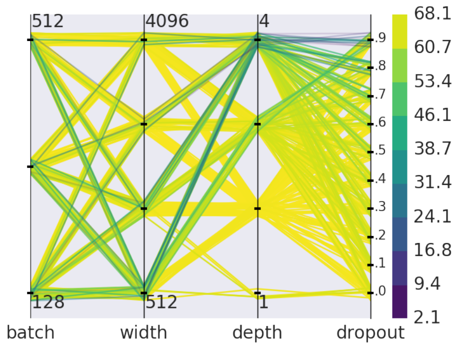
Explanation of Parallel Coordinates (PC). PC plots are a visualization technique for displaying high-dimensional data in 2D. They can highlight meaningful multivariate patterns, especially when used interactively [70]. In a PC plot, each data dimension is associated with a vertical line crossing the x-axis. Plotting a single multi-dimensional data sample involves two steps: (i) placing each dimension on its related vertical line, then (ii) connecting these points with a line of the same color, thus creating a path that crosses all the y-axes. Each dimension is normalized to the unit interval before plotting.



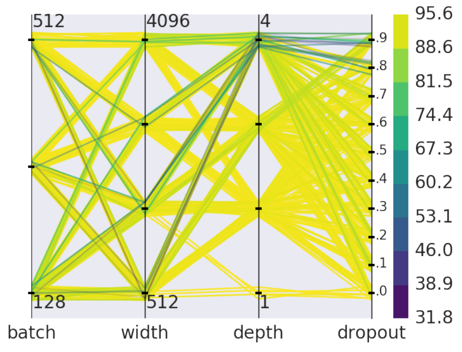
(a) UCF-101



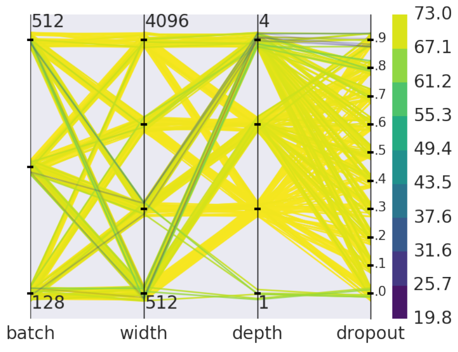
(b) High-Five



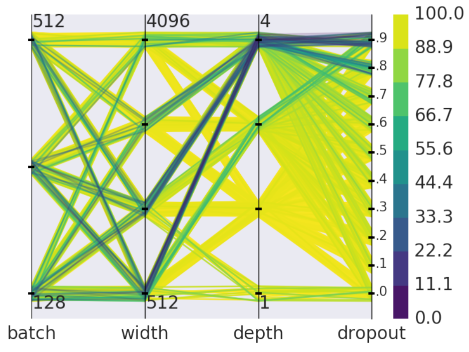
(c) HMDB-51



(d) Olympic Sports



(e) Hollywood2



(f) Combined (normalized)

Fig. 1: Parallel Coordinates plots showing the impact of multiple parameters on our hybrid architectures with unsupervised dimensionality reduction for each dataset. Each line represents one combination of parameters and colour indicates performance. Depth 2 correlates with high-performing architectures, whereas a small width and a large depth is suboptimal. The combined plot (f) has been generated by normalizing the performance in each dataset to the [0-100] interval and stacking the architectures tuples together (*cf.* explanatory text in Section 2).

In our case, each data point is a quadruplet $(batch, width, depth, dropout)$ coupled with a recognition performance number. Each hyper-parameter value has a position along the y-axis corresponding to this hyper-parameter. We slightly shift that position randomly for better readability. The recognition performance of the path corresponding to a set of hyper-parameters is encoded in the color of the line (brighter is better) and its transparency (more solid is better).
Analysis. As mentioned in Section 5.3 of the main text, a clear pattern that appears in the PC plots is the elevated brightness of depth 2 for all datasets. Another visible pattern is the relationship between lower widths and higher depths, which is always marked by the presence of suboptimal architectures (dark lines from width 512 and 1024 to depth 4). Other patterns associated with suboptimal architectures appear between large batch sizes and a small width (dark lines from batch 512 to width 512). On the other hand, connections between large widths and depth 2 are always clear in the plots for all datasets. This pattern supports the best architecture found by our systematic ranking of the architectures discussed in Section 5.3 of our submission: batch size of 128, width of 4096, and depth 2. Regarding the dropout rate, the PC plots in Figure 1 suggest that too high rates might be detrimental to the model, and the rate should be tuned depending on the rest of the architecture and on the dataset.

3 Failure and Success Cases per Dataset

In this section we provide a detailed analysis for failure and success cases for all datasets considered, contrasting our best hybrid models against our strong FV-SVM baseline.

The datasets used in our experiments can be divided according to two criteria: the presence of overlapping classes (multi-label or multi-class) and performance metric (mean Average Precision, mAP; or mean accuracy, mAcc). In this section we present precision-recall curves for datasets whose performance is computed using mAP (Hollywood2, High-Five, Olympics) and confusion matrices for datasets which are multi-class (UCF-101, HMDB-51, Olympics, High-Five). We cluster the rows and columns of the confusion matrices by level of confusion, which we obtain by ordering the rows via a 1D Locally Linear Embedding (LLE) [71]. For the largest datasets, we show only the top-25 confusion classes as determined by this embedding. For the precision-recall curves, we show at every recall decile the quantized cumulated precision segmented per class.

3.1 UCF-101

UCF-101 is the largest dataset we investigate. Using split 1, we determine the top 25 classes that are responsible for most of the confusion of our baseline model (FV-SVM). Then, we show the evolution of those same classes in the confusion matrix of our best hybrid model (Hybrid) in Figure 2. Note that the differences are small overall, as both the FV-SVM and Hybrid models have excellent recognition performance (90.6% and 92.5% resp., *cf.* Table 8 in the main text).

The most confused classes are *BreastStroke*, *FrontCrawl*, *Rafting*, *Kayaking*, *ShavingBeard* and *Brushing Teeth*. Inspecting both confusion matrices, we can see how the Hybrid model solves most of the confusion between the *BreastStroke* and *FrontCrawl* swimming actions. It also solves a visible confusion between *FrontCrawl* and *Rowing*. Furthermore, it also improves many other classes, such as *ApplyLipstick*, *ApplyEyeMakeUp*, and *HeadMassage*, but seems to reinforce certain mistakes such as the confusion between *Nunchucks* and *SalsaSpin*. Our results therefore suggest that our higher-capacity hybrid models can improve on some fine-grained recognition tasks, but not all (*i.e. Rafting* and *Kayaking*), which confirms the previously known good performance of FV and linear classifiers for fine-grained tasks [72].

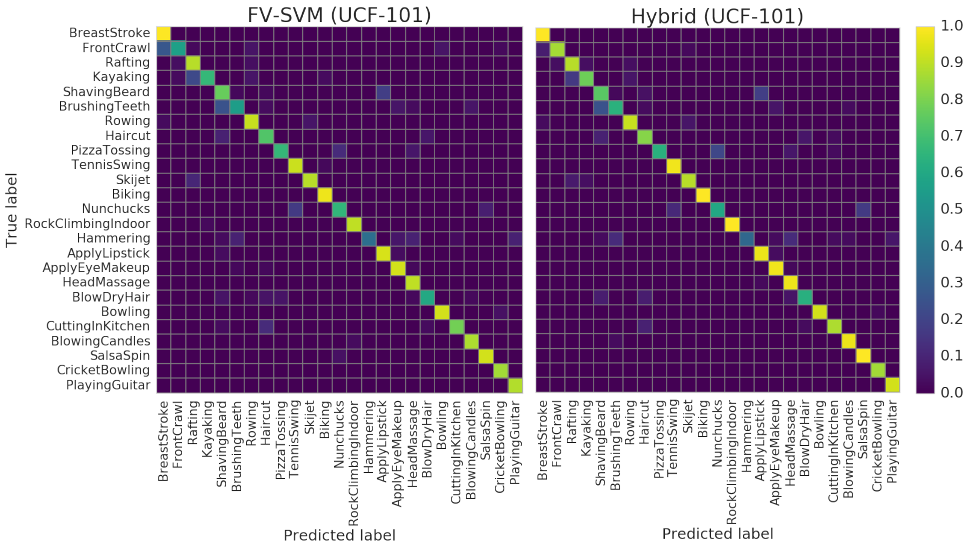


Fig. 2: Top-25 most confused classes for UCF-101.

3.2 HMDB-51

HMDB-51 is the second largest dataset we investigate. Using split 1, we generate the confusion matrices using the same method as before and show the evolution of the top 25 classes in Figure 3.

Most of the confusion in this dataset originates from the classes *punch*, *draw_sword*, *fencing*, *shoot_bow*, *sword_exercise* and *sword*. While our Hybrid model significantly improves the mean accuracy for this dataset (+2.6%, *cf.* Table 8 in the main text), the improvements are well distributed across all classes and are less visible when inspecting individual class pairs. Some improvements are between the classes *sword_exercise* and *shoot_bow*, *kick_ball* and *catch*, *wave* and *sword_exercise*.

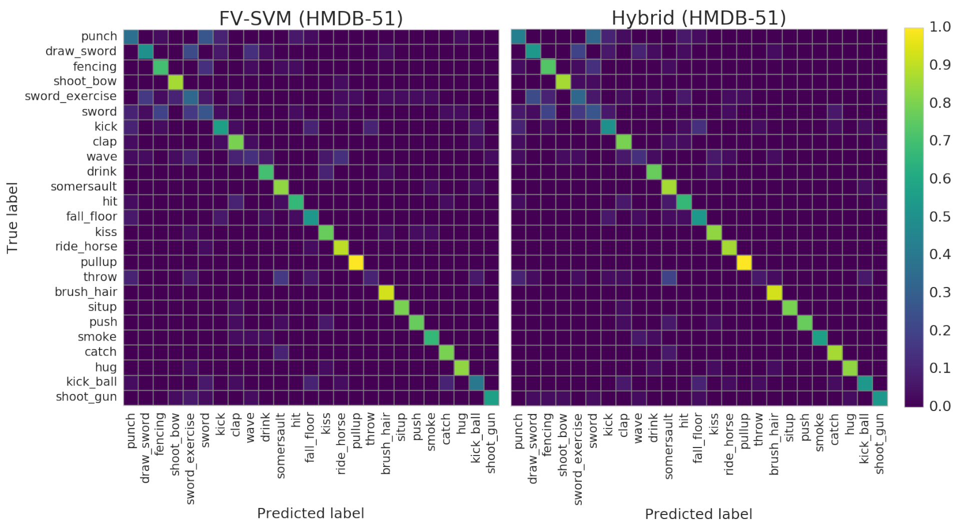


Fig. 3: Top-25 most confused classes for HMDB-51.

3.3 Hollywood2

Hollywood2 is the third largest dataset we investigate. As it is multi-label, we present a cumulative quantized precision-recall bar chart segmented per-class in Figure 4. This visualization can be understood as a quantized version of the precision-recall curve, but allows us to identify the separate influence of each class in the performance result.

We can see from the figure how the stacked bars corresponding to the Hybrid model (H) associated with higher recall rates present higher precision than the baseline (S). The difference in performance comes mostly from the *GetOutCar* and *HandShake* classes, whose precision improves at higher recall rates for the Hybrid model.

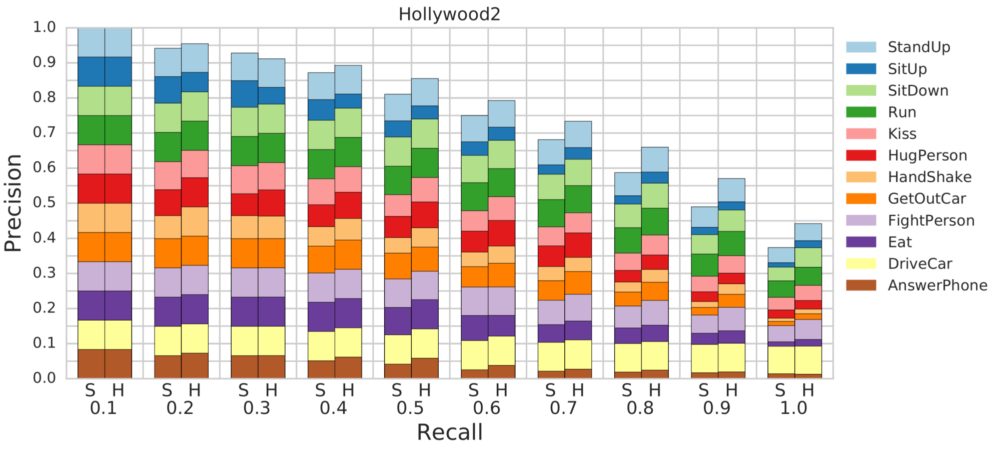


Fig. 4: Quantized precision and recall segmented per class for the Hollywood2 dataset, comparing class-specific precision between our baseline model (S) and the best hybrid model (H) at different recall rates.

3.4 Olympics

Olympics is multi-class and evaluated with mAP, so we present both confusion matrices (Figure 5) and per-class quantized precision-recall bar charts (Figure 6). Most of the confusion originates from the *long_jump*, *triple_jump*, *vault*, and *high_jump* classes. Our hybrid classifier is able to solve the strong confusion between *high_jump*, *vault*, and *pole_vault*, as well as *vault* and *platform_10m*. However, the Hybrid model has trouble distinguishing between *long_jump* and *triple_jump*, the smallest class in the dataset (17 videos for training, only 4 for testing). It has therefore low impact on mean accuracy: the overall performance in this dataset is indeed significantly improved (+3.9% *w.r.t.* FV-SVM and +5.3% *w.r.t.* the state of the art, *cf.* Table 8 in the main text).

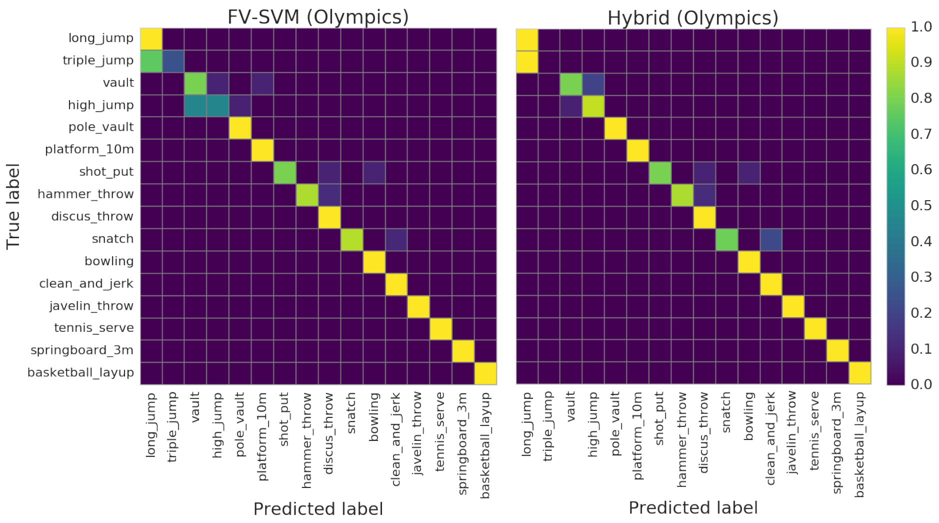


Fig. 5: Confusion matrices for Olympics.

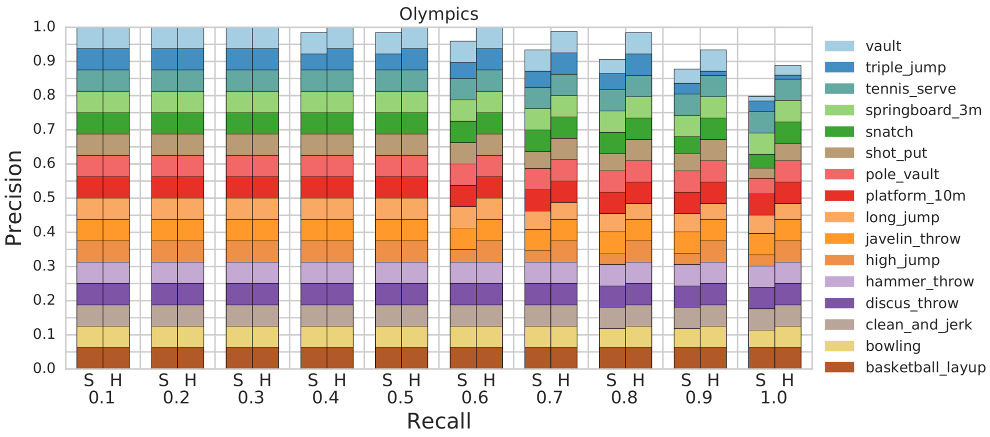


Fig. 6: Quantized precision-recall segmented per class for Olympics.

Figure 6 shows that both methods have excellent precision at low recall, with the performance between the two methods starting to differ after a recall of 40%. For higher recall rates, we can see how most of the differences stem indeed from the *triple_jump* class. The decrease in area for segments of this class are compensated by steady improvements in all other classes, as can be seen by comparing the class-specific segments between the baseline (S) and Hybrid (H) results for the 0.9 and 1.0 recall rates. This suggest that the majority of the improvements remaining are for this single class.

3.5 High-Five

High-Five is the smallest dataset (and among the most fine-grained) we experiment on. As it is multi-class and evaluated with mAP, we present confusion matrices (Figure 7) and per-class quantized precision-recall curves (Figure 8) on its first cross-validation split. Most of the confusion is between *kiss vs. hug*, and *handShake vs. hug*. The Hybrid model improves the disambiguation between *kiss* and *hug*, and *kiss* and *highFive*, but introduces some confusion between *highFive* and *handShake*, and *kiss* and *handShake*. Figure 8 shows that our Hybrid model is more precise at high recall rates, especially for *kiss* and *handShake*. Moderate improvements can be observed for all classes.

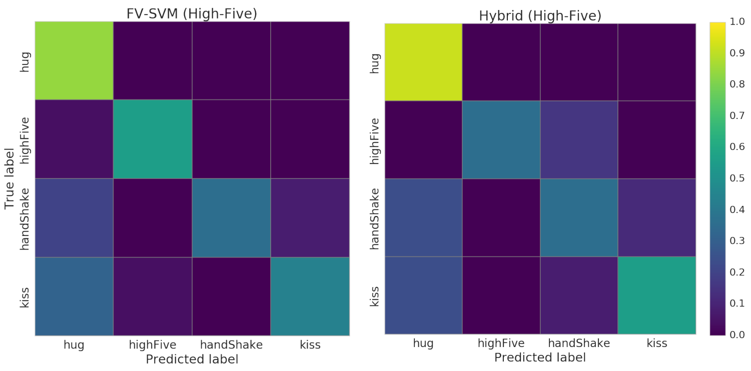


Fig. 7: Confusion matrices for the first cross-validation split of High-Five.

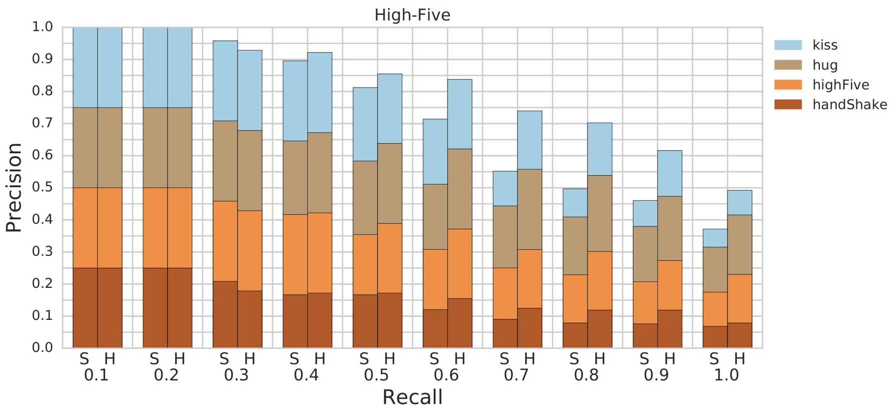


Fig. 8: Quantized precision-recall segmented per class for split 1 of High-Five.

3.6 Conclusion

Our detailed per-dataset analysis suggests that the improvements brought by our hybrid models are generally distributed across the classes of most datasets, without a single class being responsible for the main performance jump. We also show that the Hybrid model can differentiate between some fine-grained action groups, confirming that the unsupervised video-level FV representation contains fine-grained information about the original video, an information that may be more successfully exploited by our more complex (deeper) hybrid models.

The experimental evidence presented here therefore strengthens the overall good results reported in Table 7 of the main text, where we show that our method improves over the strong FV-SVM baseline between $+1.9\%$ for the large UCF-101 dataset and $+5.7\%$ on the small fine-grained High-Five dataset.