Augmentation of Two-stream CNN architectures with context and attention for action detection and recognition

Pedro Diogo Fernandes de Abreu

***Abstract*—Tasks such as action detection and recognition are a promising step in several areas such as retail, security, robotics and recommendation systems. Recently, challenging datasets have been introduced, which are representative of the task of multi-person spatiotemporal action detection and recognition task with multi-labels. We propose to augment the state-of-the-art two-stream CNN architectures for this task. These architectures are limited in that they try to detect the actions independent of the background and other humans in the same video. To this end, three novel contributions are presented: attention filtering, context streams and a combination of both. For attention filtering, with the goal of not only extracting information from a target but from the image background, we train two-stream CNN architectures with different kinds of filters applied on RGB and Optical Flow inputs. For context streams, with the goal of predicting the labels of a target using the labels of the surrounding neighbours, we use dataset labels to explicitly encode the relationship of classes performed by multiple humans as context features and train LSTM networks on these features. Finally, we combine these methods by fusing the context streams with the two-stream approaches trained with attention filtering. Results show the combination of the first two methods outperforms each of them and that all augmentations improve on a two-stream CNN baseline.**

***Index Terms*—Action Detection, Action Recognition, Multi label datasets, Attention filters, Spatiotemporal context label relationships, Two-stream Convolutional Neural Networks.**

I. Introduction

Recently video-based human action detection and recognition has been getting more attention since it allows for the detection of a larger range of actions and incorporation of more contextual information than inertial sensors [1]. As a consequence, video-based human action detection and recognition has many possible applications in retail, security [2], robotics and as a representative task for video understanding (i.e to understand human video dynamics). A large commercial payoff would come from accurate human action recognition and detection since, with the amount of online videos increasing everyday [3], a major drawback is the need to analyze video either for improving suggestions or to check for undesired content by hand.

A. Problem Description

We firstly introduce the basic concepts involving atomic, action and activities and also the type of task, namely, action recognition and action detection and multi-class opposed to multi-label tasks.

While most literature uses a somewhat fluid notion of many of these concepts it is important to separate the notions hierarchically [4] as activity, action and atomic action for humans. [5] define **atomic action** as an atomic movement that can be described at a limb level (leg forward), an **action** as a sequence of atomic actions, and an **activity** as a whole body movement (running) containing a number of subsequent actions. [6] also introduce verb-only labels which tend to be associated with atomic actions as a means to reduce the ambiguity in many action class labels. More recent datasets, Charades/Holywood In Homes [7], Moments in Time [8], and the dataset we use, AVA [9], tend to use atomic actions with that same intention of reducing ambiguity, while older datasets like UCF101 [10], HMDB51 [11] and Sports-1M [12] tend to focus on broader activities. Furthermore, [13] [14] show how actions can be decomposed into and grouped according to an atomic action structure that can be learned.

Similarly to the notion of actions as sequences, the idea of using meaning to categorize actions is also popular and [15] define action as the most elementary human movement that surrounds interaction with meaning. The meaning associated with this interaction is called the category of the action: this is important as it allows us to separate the actions into conceptual groups.

As for the tasks, from a computer vision standpoint the objective is to match an observation (video) with one or more labels of actions as a supervised learning task. Action recognition (classification is a synonym) [16] can be defined as the task of categorizing an action in a video clip to one or more of the pre-defined set of actions. Action detection [16] (localization is a synonym) is the task of correctly specifying via a boundary (e.g a bounding box) where an action is located throughout the video. This means that action detection can be both interpreted as spatial and temporal and multiple people can perform different actions in the same frame. Additionally, in theory, a single-person can have multiple labels, that is, a single person can perform one or more actions at the same time. This distinguishes between Single Label/Multi-class Datasets and Multi Label Datasets [17], and the tasks must adapt accordingly.

Action recognition tasks are challenging due to the long temporal context needed for models [16]. As shown in recent datasets [18], the task is made even harder as datasets tend to be large due to the large amount of long videos with a high FPS rate. This implies models are more complex and

computationally heavy which makes overfitting likely [19]. Since we have a very small fraction of the resources available to other implementations, one of our goals is to try to make our approach as light in terms of resources as possible without compromising results. Another noteworthy observation is that action recognition tasks are more difficult than actor detection [9] (i.e spatial detection of people performing actions). As such, for our approach we mainly focus on the action recognition component and assume the bounding boxes of the targets are given a priori.

Since we believe many recent datasets have been too simplistic and do not reflect the real complexity of this task [9] [20], our main goal in this work is to go beyond the common challenges of action recognition and tackle a multi-label multi-person action detection and recognition task in an imbalanced dataset where we can exploit a rich action structure as previously described.

B. Main Contributions

Our main contribution is a novel architecture that deals with an imbalanced, multi-label, action detection and recognition task with a small amount of computational resources that builds on previous similar approaches [21] [22] [23]. Our method extends these approaches by using additional mechanisms such as attention and inter-human class relationship context. Due to the large size of the dataset (AVA [9]), we partitioned it to obtain a smaller representative set with a similar distribution (miniAVA) which we provide for the research community. Additionally, since we will release our models online for free, we would like to highlight how there is a lack of uniform standards in the deep learning research community and how models are often available in non-compatible frameworks, which forced us to implement tools to translate between these frameworks. We also highlight that most of the available frameworks rarely have any utilities for multi-label or imbalanced problems and this fact often forced us to extend already existing frameworks to suit our needs.

II. Previous Work

In this section we present a brief overview of the main previous work that contributed to our approach and the current related state of the art.

Firstly, we can separate approaches in two types: before Convolutional Neural Networks (CNNs) and after CNNs. Before most pipelines were called *handcrafted* since they heavily focused on features that were often computed and encoded via interpretable, analytical methods of which iDT (improved Dense Trajectories) [24] is a good example.

After data-driven approaches became more powerful, the use of CNN architectures in action recognition quickly became the state of the art [25]. The more modern action recognition frameworks normally differ in how they employ temporal information and can be categorized as shown in Fig. 1. Since human actions can be interpreted as sequences over time it is relevant to mention an architecture that was designed to exploit temporal sequences: Recurrent Neural Networks (RNNs). More recently, a more efficient architecture was proposed by [26] named Long Short Term Memory (LSTMs), which was able to handle longer sequences by altering the main repeating cell in these networks to have memory elements. The main idea of most LSTM based action recognition approaches is to use LSTMs to model sequences [27] on trained features maps extracted with CNNs to capture long temporal information.

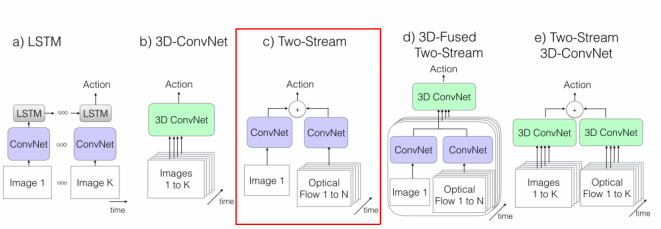


Fig. 1. Examples of typical action recognition approaches with CNNs as shown in [20]. The rightmost two approaches, while achieving the best results, tend to be computationally heavy due to performing 3D convolutions while the leftmost two tend to lack explicit motion features like OF which have been proven to be advantageous.

While we use LSTM networks, our main focus is on two stream approaches like Two-StreamFusion [28] or TSN [23]. The most novel addition introduced [21] by this type of network model is that the use of an additional CNN trained on explicit motion features such as optical flow to try and capture short temporal information. These networks are particularly attractive due to the possibility of achieving competitive results while using small amounts of computational resources since they tend to use only 2D CNNs and a frame level classification scheme. More recent and computationally heavy approaches like I3D [20] simply make these networks perform 3D Convolutions on stacked volumes of RGB frames and Optical Flow.

A key aspect of this architecture is optical flow computation. Optical Flow computes the displacement vectors between two frames to obtain a representation of motion, which are then stacked (over pair-wise video frames) and used as input. In our implementation we used the TV-L1 [29] algorithm since [20] showed it outperformed other methods including network-based methods offering a good balance between computational resources and accuracy.

One of the first architectures to employ two CNN streams for action recognition [21] addressed the failures of previous single-stream early approaches [12] by using explicit motion features such as Optical Flow. The input to the spatial stream is a single frame of the video, and the input to the temporal stream are successive optical frames centered around the input RGB frame. Each stream was pre-trained separately and the output scores of both streams are averaged as seen in Fig. 2 which is called *class score fusion*. Despite seeming a rudimentary approach, other recent state-of-the-art methods have achieved good results with this type of fusion [23] for a larger number of streams.

Recently, [28] go beyond the architecture of [21] and

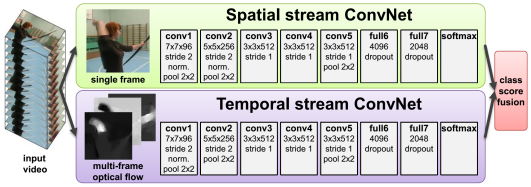


Fig. 2. Early 2-stream CNN [21] as an example of Class Score Fusion and two stream inputs.

explore alternative fusion strategies that allow for the learning of combined spatial and temporal features. An example of such method is *concatenation fusion*. While the definition used by [28] is for the fusion of convolutional layers the same ideas were used for the fusion of fully connected layers was used. Concatenation fusion does not define a layer with filters itself but leaves this to subsequent fully connected layers to learn suitable weights. While these results are tested on relatively small networks, the same authors [22] [30] tested their concepts also on deeper architectures like ResNet [31] variations with similar results.

III. Implementation

In Fig. 3 we show the main pipeline of our approach. Original videos are split in 3s second segments. We use the TV-L1 [29] for computing optical flow which we store in disk and we use groundtruth bounding boxes at training time. However, at testing time we can see that some error propagation from using another network for bounding box extraction as a pre-processing step can occur. We argue this impact is not the main priority when trying to improve performance as shown by the relevant ablation study in [9]. Some key aspects of the pipeline are the attention filtering to filter out background of frames given bounding boxes or using labels to generate context features which are then used to train context architectures. We note that even the two-stream architecture is tailored for this specific task and as such some aspects of it may not generalize to all action recognition datasets. Examples of this include our subsampling strategy and our custom output layers/loss function.

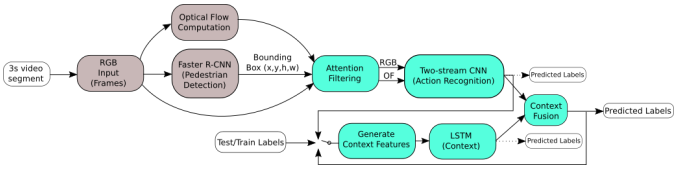


Fig. 3. Conceptual pipeline of our approach. In green are the pipeline components we are mostly focused on, the ones we do not focus on are grayed out. In white are the inputs and outputs of the pipeline. The switch shows how context features can be generated from 3 different sources.

In the following sections we first give a description of the dataset we use followed by all the improvements we proposed in the aforementioned pipeline.

A. AVA and miniAVA

Firstly we describe the main AVA [9] dataset. This dataset is a multi-label spatio-temporal action detection and recognition dataset which was created to be purposefully challenging to modern approaches. This dataset is heavily class imbalanced so as to emulate real data acquisition. All the actions are atomic actions (very elementary or "fine-grain" actions) which tend to interact in more complex patterns.

The annotations are done at a sampling frequency of 1 Hz, each sample corresponding to a keyframe. To provide temporal context, the labels are for short segments of 3 seconds centered on the keyframes. There are 80 atomic actions labels and the dataset is sourced from the 15th to the 30th minute time intervals of 192 movies at an average rate of 30 FPS.

Fig. 4. Examples of AVA [9] labeling. In yellow pose actions, in blue human-human actions and in red human-object actions.

The dataset has a unique multi-label structure as shown in Fig. 4. Each bounding box in a frame must have 1 pose label like *stand* or *walk* (from mutually exclusive C_P of them), 0-3 human-human labels (from non-mutually exclusive C_H of them) like *talk to* or *hit* and 0-3 human-object labels (from non-mutually exclusive C_O of them) like *hold object* or *watch (e.g TV)*. The total number of classes is $C = C_P + C_H + C_O$.

Due to the large dimension of the original AVA [9] dataset we had to make a partition of the original dataset on which to train and test our models. As we subsampled the dataset we had to keep two facts in mind. Firstly, we wanted to maintain temporal continuity in the samples (i.e have 3s segments from the same 15 minute snippets) which we achieved by sampling in the first segments from each split of the AVA dataset, which while not ideal, assured the continuity which would be lost in random sampling. Secondly, we wanted to maintain a distribution that still followed the original distribution for the training, validation and testing split as shown in Fig. 5 even with a smaller number of classes as shown in Table I. We took particular care to make sure that all classes had at least 20 samples in the test set. The number of samples chosen for each set are such that we could process them in a reasonable time, particularly for the training set.

TABLE I
CLASS CATEGORIES IN THE AVA [9] DATASET VS MINIAVA.

Class Category	miniAVA	AVA	Mutually Exclusive?
Pose	10	14	Yes
Human-Object	12	49	No
Human-Human	8	17	No

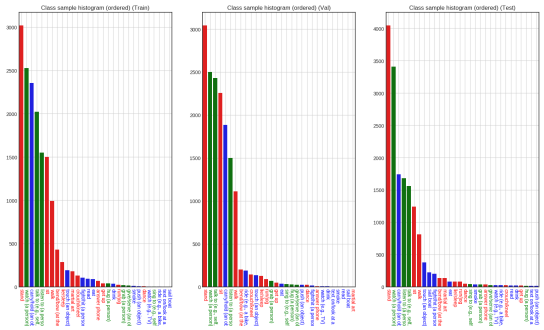


Fig. 5. The distribution of the training, validation and test set of our partition of the AVA [9] dataset. The colors of the labels reflect the type of action, we use alternative colors to the original dataset as in Fig. 4.

B. miniAVA Context

On another topic it is relevant to ascertain two important aspects: firstly, how much information from neighbours over a time window can actually be learned given that it is possible for segments to only have a single actor and secondly how rich is the inter-class, inter-actor temporal context in our partition of the larger AVA [9]. For the first aspect, in Fig. 6 we can

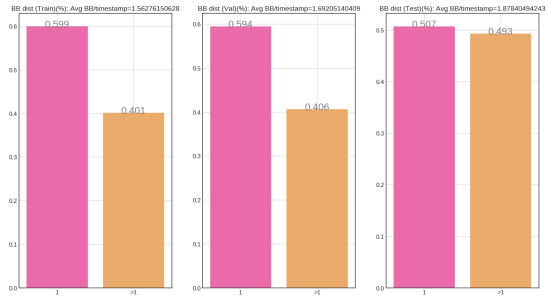


Fig. 6. Relative number of BB's in segments (in pink, percentage of single BB segments, in orange those with more than one). Notice that the average bounding box per video is lower on the training set than on the testing set and that the test set has a larger amount of groups.

see that in the training and validation sets most segments have only a single actor/BB. This means that the context networks are only learning relevant information from a small percentage of the dataset. The test dataset on the other hand seems to have a more balanced distribution in terms of BBs per frame which means that we fortunately have an ideal challenging situation on which to test our models. For the second aspect, in Fig. 7 we show a matrix that depicts the normalized co-occurrence of classes in our test set. Several interesting facts can be interpreted from this matrix that reveal structures a classifier could use (for example the large correlation between *martial-arts* and *fight/hit a person*)

C. Generalized Binary Loss Function

At a first glance, if we notice the label structure of the AVA [9] dataset as discussed in section III-A and if we followed the example of the provided AVA Localization Model [9] we would have an output layer of C (number of classes) independent sigmoid activation functions. Note however that this does not truly reflect the label structure of the AVA [9] dataset which is not entirely mutually exclusive but also not entirely multi-label.

As such, in our architecture we address this heterogeneity by having three separate output layers one of which (corresponding to the pose classes) has size C_P and has a softmax activation function and the other two each have many sigmoid activation functions for each of their element vectors and their sizes are respectively C_H and C_O.

This implies that the loss function cannot simply be categorical cross-entropy or binary cross-entropy [32]. Therefore the architecture must minimize a global loss which is the sum of the losses of each output layer and which can be expressed as follows:

$$\mathcal{L}_{GB} = \overbrace{-\log\left(\frac{e^{s_p}}{\sum_j^{C_P} e^{s_j}}\right)}^{pose\ classes\ loss} + \overbrace{\sum_j^{C_H}\left(-\sum_{i=1}^{2} t_{j,i}\log(\sigma(s_{j,i}))\right)}^{human-human\ classes\ loss} + \overbrace{\sum_j^{C_O}\left(-\sum_{i=1}^{2} t_{j,i}\log(\sigma(s_{j,i}))\right)}^{human-object\ classes\ loss} \quad (1)$$

where the s vectors are the predictions of each layer in equation 1 and are different for each component of the sum, s_p is the only positive label in the softmax layer, and σ is the sigmoid function. Note that this backpropagation is possible as the encoding of the target labels is done as a binary vector which can then be partitioned into smaller vectors for each output layer. Given all this and since we could not find any type of similar loss in the literature we decided to name this loss *generalized binary loss*.

D. Subsampling and Voting Scheme

A straightforward implementation, at a rate of 30 FPS for each 3s segment, would need a receptive field of 90 frames and a 3D architecture to process a single segment. This would imply a much more computationally heavy architecture and as such we propose a subsampling scheme as shown in Fig. 8.

While it is inevitable to lose some information this makes our implementation more computationally feasible and we consider that our sampling is spaced enough that the frames are representative of the whole segment. Since we have 5 frames and OF volumes representative of a segment, we use a voting scheme to arrive at a consensus for the label to assign to a segment.

As such, to obtain the predictions for a single segment we pass each frame of our 5 subsampled representative frames and its corresponding OF through the network and we store its predictions. For the mutually exclusive (pose classes) predictions we count the maximum valued prediction as a vote and for all non-mutually exclusive (human-human and human-object classes) predictions we count any prediction values above a certain threshold v as a valid vote. Then, for the mutually exclusive classes we take the most voted class

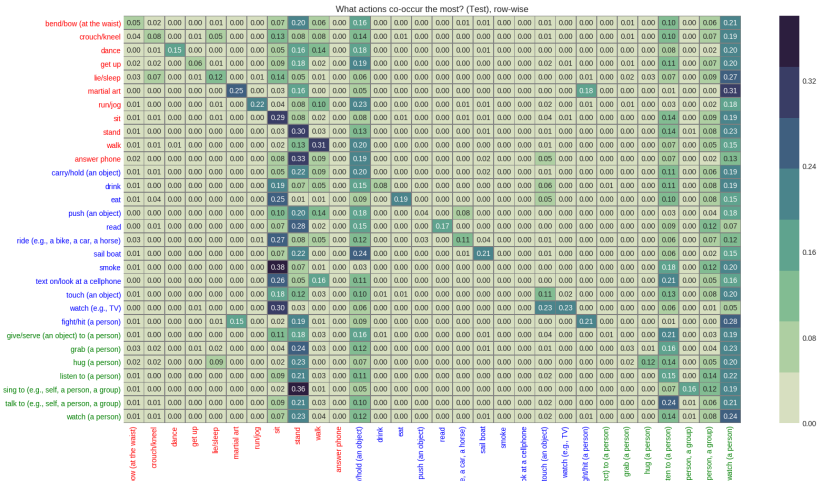


Fig. 7. Co-occurence matrix of the miniAVA Test Set reveals a rich contextual environment even for our small dataset. For all frames of a given action (as a row) we count the actions of other actors in the same frames (across the columns), and then we normalize across the rows. In red are the pose classes, in blue are the human-object classes, in green are the human-human classes.

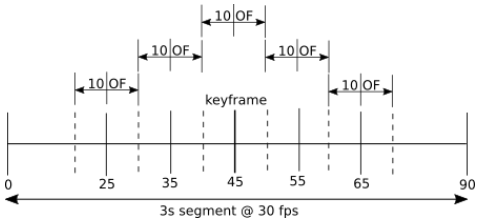


Fig. 8. Our subsampling scheme. We extract 5 representative RGB frames and 5 OF stacks around a keyframe.

as the predicted class for the segment and for each of the non mutually exclusive classes we take their top 3 most voted classes as the predicted classes for the segment. For the non-mutually exclusive classes the number of predicted classes will be between 0 (if no prediction is above the threshold) and 3.

E. Attention Filtering

The main idea for attention filters is that, given a region in an image (for example, a bounding box) we filter the image in such way that a classifier learns features related to that region rather than with the surrounding background.

Two naive approaches are possible: the first, which we refer to as Crop filter, would be to crop the area outside that region and the second, which we refer to as GBB (Gaussian Background Blur) would be to simply apply a Gaussian blur with a Gaussian kernel to everything outside that region.

Despite the goal of the attention filters, the Crop filter having no background context at all might hinder the classifier (i.e an abundance of blue might help a classifier guess the *swim* action). Additionally for both the GBB and the Crop filter, regions with large contrasts exist (i.e in the crop filter there is a sharp transition from the attention region to black and in the GBB there is a sharp transition from the relevant region to a blurred region). Since many network based approaches tend to learn edges as low-level features, we hypothesize these artificially introduced edges would compromise their effectiveness.

To solve this problem we use an artificial foveal vision filter from [33] inspired by human foveal vision [34] and the work of [35] which provides a smooth blur transition between the bounding box and the background. Firstly, a Gaussian pyramid is built with increasing levels of blur. The image g_{k+1} can be obtained from g_k via convolution with 2D isotopic Gaussian filter kernels with progressively higher $\sigma_k = 2^{k-1}\sigma_1$ standard deviations for each kth level. Next, a Laplacian pyramid [35] is computed from the difference between adjacent Gaussian levels. Finally, exponential weighted kernels are multiplied by each level of the Laplacian pyramid to emulate a smooth fovea:

$$k(u, v, f_{kx}, f_{ky}) = e^{-\left(\frac{(u-u_0)^2}{2f_{kx}^2} + \frac{(v-v_0)^2}{2f_{ky}^2}\right)}, \qquad 0 \leq k \leq K \tag{2}$$

where $f_{0x} = \frac{1}{2}w$, $f_{0y} = \frac{1}{2}h$, and $f_{kx} = 2^k f_{0x}$, $f_{ky} = 2^k f_{0y}$ is used to define the fovea intensity at the k-th level and the fovea is centered at $u_o = x + w/2$, $v_o = y + h/2$, given (x, y, h, w) where x, y are top-left corner coordinates and h, w are the height and width of the bounding box.

For the Optical Flow input frames we decided to only employ the crop filter because both GBB and fovea filtering involve blurring and the blurring of motion features would have a possibly misleading and undesirable interpretation of creating artificial motion vectors in incorrect places.

F. Two Stream Model

We propose a base architecture inspired by two stream approaches such as [28] [30]. We chose a ResNet50 architecture for each stream as not only has this architecture demonstrated to be successful [22] for these tasks, but we had pre-existing weights from [28] trained on UCF101 [10]. Each network stream, RGB and OF, was individually fine-tuned for our task.

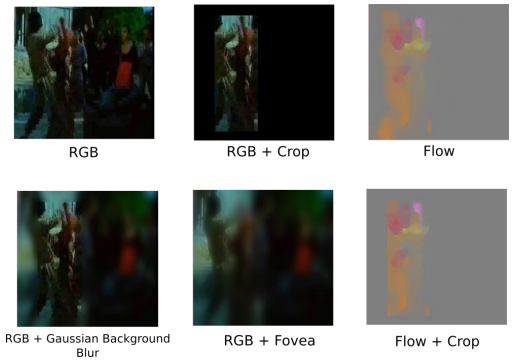


Fig. 9. Example of all the attention filters applied on an RGB frame and on an OF frame.

We fuse these networks using concatenation fusion of their last FC layers and then train a FC layer so that spatiotemporal features can be learned as shown in Fig. 10. Since retraining both networks together would be too computationally heavy we loaded their previously individually fine-tuned weights, and froze them (i.e not update them in backpropagation) so we only trained the desired FC spatiotemporal filters.

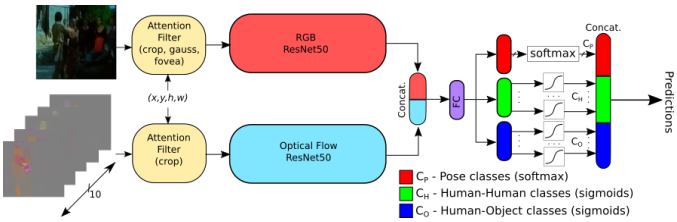


Fig. 10. Our proposed two-stream architecture. Remember that for the presented architecture the input is a single frame and an optical flow stack and the output are a set of floating point predictions. The predicted labels are merely illustrative.

G. Context Features and Architecture

Based on the idea of human to human interactions we compute features that encode relations of human actions over time. More specifically we define context as trying to learn the labels of one actor based on the labels of its neighbours across time. Given the labels for a given actor, we must encode the labels of their neighbours such that these can be used as input features.

We assume the number of timesteps used from the past and the number of timesteps used from the future are identical T (in seconds) and we chose the N closest neighbours to an actor using a pixel-wise Euclidean distance between their BB's center and we assume this order is kept throughout the entire context encoding. If our task has C classes then each timestep has NC features and the length of the context vector is $(T+1+T) \times N \times C$, as we can see in Fig. 11.

Our architecture has two LSTM layers as an input: one receives a sequence of the past timesteps and the present timestep (i.e it must have a $T+1$ timestep receptive field) and another receives a sequence of the future timesteps and

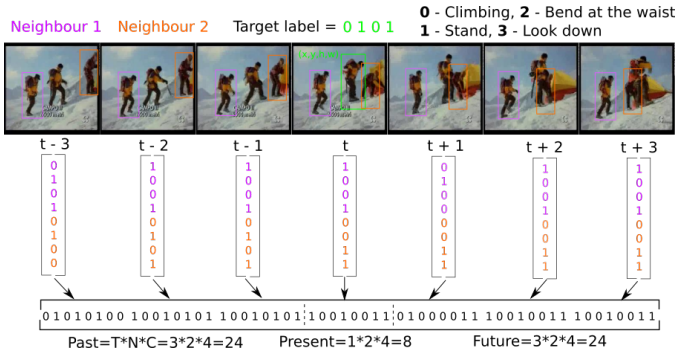
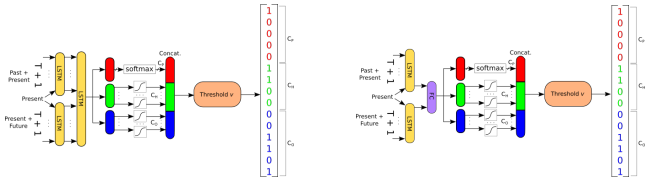


Fig. 11. A simple example of context generation with number of backward and forward timesteps $T = 3$, neighbours $N = 2$ and number of classes $C = 4$. At the end of the procedure we obtain the context vector for target (x, y, h, w) at time t. The same procedure must be done for the other targets at other times.

the present timestep (i.e it also must have a $T+1$ timestep receptive field). The present timestep is used in both inputs since using it only as input in one layer would introduce a bias towards either past or future timesteps. Two different approaches appear as how to merge the outputs of these two LSTM layers: using an additional LSTM layer or an FC layer as in Fig. 12:



(a) Model A **(b)** Model B

Fig. 12. Our two proposed context models: model A encodes the input sequence into yet another sequence, model B outputs two predictions which are then fused with a FC layer.

H. Context Fusion

We propose to explore two ways in which the fusion of our context models and the two-stream attention models can be made: concatenation fusion and class score fusion. Concatenation fusion is similar to the previous two stream fusion and is shown in Fig. 13, but the concatenation now also uses the last layer of the context models and for each of the 5 subsampled frames the context features are assumed to be the same (as shown in the further passes of Fig. 14).

The main idea behind class score fusion, shown in the further passes of Fig. 15, is that at testing time the predictions from both models (two stream and context) are averaged. Since context operates at a keyframe/segment level the prediction for every subsampled frame is the same, but the predictions for each frame from the two stream model differ. While this does not require training, there are no learned shared weights as in concatenation fusion. In a first approach we used groundtruth labels to generate context vectors for our context models to evaluate if context does indeed help. However, since this

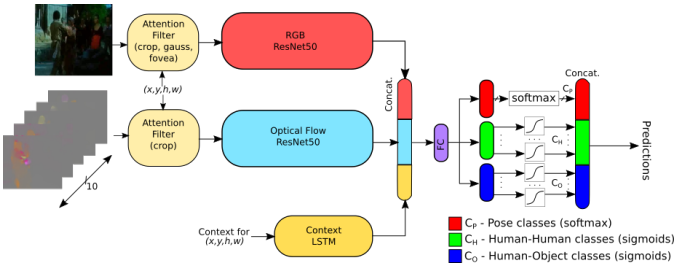


Fig. 13. Our proposed concatenation fusion model for context. Note that the Fig. shows only the input of a single frame and that each 3s segment is subsampled to 5 RGB frames.

assumption cannot be used in real cases, we propose a two-pass testing scheme in 15 where this is no longer the case.

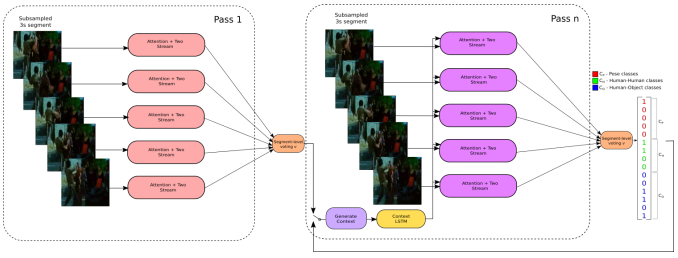


Fig. 14. Our proposed two pass testing scheme when the further passes use the concatenation fusion method. We omit that each two stream model also receives the optical flow volume corresponding to its frame according to Fig. 9 for simplicity. The predicted labels are merely illustrative.

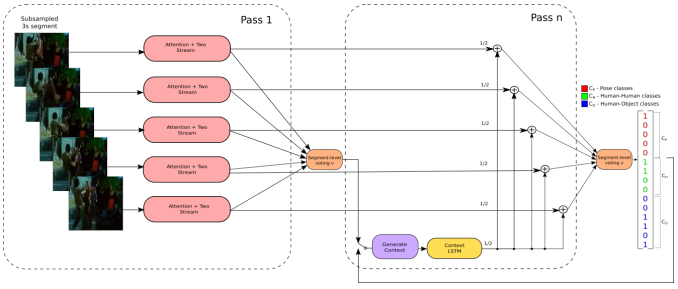


Fig. 15. Our proposed two pass testing scheme when the further passes use the class score fusion method. The predicted labels are merely illustrative.

For the first pass we use the previously explained two stream with attention methods to generate predictions on the test set. These predictions are then used to generate labels that are then used to generate context features and as shown in Fig. 15 and 14 this can be repeated further n times. These are then used by our context models for yet another pass using one of the previously described fusion methods. If the initial two stream model generated perfect predictions, this would be the same scenario of using groundtruth labels to generate context.

IV. Results

All the following experiments were carried out with a single GTX 1080 Ti and 16-32 GB RAM. The AVA task involves localizing the atomic actions in space and time, achieving the highest mAP (at 0.5 IoU) possible on 3s segments extracted around 1 FPS keyframes of 15-minute segments. mAP is the AP (average precision) averaged over all classes. For CNNs and following the recommendations of [36] the training was for 200 epochs, batch size is 32 with a learning rate of 0.001 decaying to 0.0001 after 80% of the epochs, while for the LSTMs the training time was for 100 epochs with a fixed learning rate of 0.001. The RGB and OF streams were ResNet50 [31] networks which were initialized with weights from [22] from the respective RGB and OF networks trained on the UCF101 dataset. For the results we use the same benchmarking tools as those provided for the AVA challenge.

A. Baseline

The results of this experiment are the analogous of the 2D AVA Localization Model baseline but on our smaller dataset. We trained two separate ResNet50 networks, one on RGB frames and one on the extracted optical flow volumes. Then we fused these two networks for a two-stream approach using concatenation fusion and fine-tuned a FC layer to learn spatio-temporal features as shown in Fig. 10.

TABLE II
BASELINE INDIVIDUAL STREAMS AND THEIR FUSION.

Model	mAP@0.5IoU
RGB	5.06%
Flow	5.85%
RGB + Flow	5.00%

We can observe several facts from these results shown in Table II. Firstly, we note how using only OF performs better than RGB. This has been also found the case for some implementations of these types of networks and is often due to the fact that certain actions have very clear motion patterns. Secondly, we highlight how the baseline of the fusion is lower than both approaches, which suggests that the spatiotemporal features being learned are not properly using complementary features from both streams. We believe our small partition is even more challenging as the use of even less data increases the chance of overfitting.

B. Attention for individual streams

This experiment can be divided into two parts: testing attention filtering on RGB frames and attention filtering on Optical Flow volumes and the results are shown in Table III. For the first part we trained individual streams on the outputs of crop filtering, GBB filtering and fovea filtering, while for the second part we trained an individual stream on cropped optical flow as shown in Fig. 9.

Two main conclusions can be drawn from these results. One is that the use of all pre-filtering attention mechanisms improve results, although marginally for Optical Flow. The second is that the filtering techniques we hypothesize would lead to the networks learning artificial edges (i.e not fovea) perform best. This may be due to the fact that these artificial edges might be contributing to the prediction of certain classes, particularly *stand*, which is the most common class in miniAVA.

TABLE III
ATTENTION FILTERING RESULTS ON INDIVIDUAL RGB STREAMS AND INDIVIDUAL OF STREAMS.

Model	mAP@0.5IoU
RGB + GBB	5.63%
RGB + Crop	5.19%
RGB + Fovea	5.12%
Flow + Crop	5.90%

C. Two-Stream Fusion

For this experiment we fused paired combinations of the streams trained on pre-filtered inputs from the previous experiment as shown in Fig. 10 and show their results in Table IV. We fuse them in the same fashion as shown before in Fig. 10. Remember that one of our goals is to analyze how the attention filters impact the joint learning of spatiotemporal filters which is only possible with this type of fusion. We did not fuse RGB and cropped flow as we felt the improvement on the baseline might be minimal, and we prioritized RGB attention filters over flow attention filters due to the results of the last experiments. The first result is that cropped flow seems

TABLE IV
TESTING OF SEVERAL COMBINATIONS OF STREAMS AND THEIR RESPECTIVE ATTENTION FILTERS

mAP@0.5IoU	Flow	Flow + Crop
RGB + GBB	3.59%	4.16%
RGB + Crop	5.01%	5.06%
RGB + Fovea	**5.94**%	4.95%

to worsen results when fused with all other streams except the RGB crop stream. This suggests some synergy in the learned features. A second result is how the fovea filter outperforms all others. At first we hypothesized this might be due to the fact that the artificial edges introduced by the other filters would harm their performance when later merged with flow features, upon further analysis we conclude that all two stream approaches seem to be fitting the distribution and learning only from a few relevant classes, however the two stream approach using fovea filtering seems to hit more samples than the others in some of these under represented classes as shown in Table V.

TABLE V
AP ON SOME NOTEWORTHY CLASSES THAT ILLUSTRATE WHY THE FOVEA FILTER SEEMS TO PERFORM BEST.

AP@0.5IoU	Stand	Listen	Touch (an object)
Flow + (RGB + GBB)	53.3%	0.0%	0.0%
Flow + (RGB + Crop)	63.3%	0.0%	0.0%
Flow + (RGB + Fovea)	63.3%	28.4%	6.1%

Nonetheless, while this analysis provides valid hints to the true validity of the methods, they need to be further investigated by themselves on a more balanced dataset where their results are not as affected by the large imbalance.

D. Best Context Architecture

Before discussing the results, whenever we mention the number of hidden units for both our context models we are referring to the sizes of the two input LSTM layers. The size of the next layer, whether a FC or LSTM layer, is always half this value.

In this experience we propose to analyze how well the context architectures shown in Fig. 12 perform by themselves. This has the advantage that the context models we tested were much less computationally heavy and as such we could test the architectures more extensively. We try to find the best context parameters used to generate the data namely the time window $T = (3, 5, 10)$ and the best architecture between model A and model B by testing over several values of the number of hidden units $(64, 128, 256, 512)$. We tested larger versions of the best model A and the best model B (with a larger number of hidden units $(1024, 2048)$) to see if the models required a larger number of hidden units to learn the complexity of the data.

TABLE VI
EVALUATION OF THE BEST CONTEXT GENERATION FOR MODEL A AND MODEL B LSTM ARCHITECTURE, WITH THE COLUMNS CORRESPONDING TO THE NUMBER OF HIDDEN UNITS. ALL RESULTS USE $N = 3$ (I.E THREE CLOSEST NEIGHBOURS)

A	64	128	256	512	1024	2048
T=3	4.99%	5.04%	4.96%	4.80%	–	–
T=5	5.01%	4.97%	4.98%	**5.05%**	5.09%	5.11%
T=10	4.85%	5.00%	4.68%	5.04%	–	–
B	64	128	256	512	1024	2048
T=3	5.01%	5.00%	4.92%	**5.12%**	5.08%	5.09%
T=5	4.93%	4.94%	4.95%	5.04%	–	–
T=10	4.81%	4.90%	4.92%	4.97%	–	–

We can draw several conclusions from Table VI. The first one is that while model B is marginally better, model A seems to operate better with a slightly larger time window, regardless of the fact that there is not a significant difference between the best model A architecture or the best model B architecture. Model A seems to need larger layers to model more complex sequences successfully. Since model A performs better for a larger NHU, this allows us to use it for concatenation context fusion in the next experiment, since in concatenation context fusion (shown in Fig. 13) all streams should have a contribution with the same number of hidden units (1024 NHU in our experiments). As such we chose model A as the best model from this experiment.

Evaluation on a longer time window or more neighbours was not done since we believe actions longer than 21 (10 + 1 + 10) seconds are rare and so are complex interactions between more than 3 people.

E. Context Fusion

Given the discussion in our previous experiments we use LSTM model A and we use the two-stream with fovea filtering and optical flow with no filtering as the basis for all fusions. We test concatenation fusion and class score fusion under two testing schemes: one uses test labels (i.e groundtruth)

to generate context and the other, as shown in 15, uses the predictions generated by a first pass through the best of the two-stream architectures to generate context.

The results of the first are shown in VII where we can see that class score fusion largely outperforms the other methods. We believe this may be because when concatenated, the internal feature representations of the LSTM models are too distinct from the two-stream spatiotemporal features for a FC layer to learn merged features. However, at a class score level the context predictions are able to correct the two-stream predictions. Note that while this situation is an idealized situation the context features only use information of labels from neighbours of actors in frames not from the actors in the frames themselves.

TABLE VII
TESTING CONTEXT FUSION ARCHITECTURES UNDER A GROUNDTRUTH CONTEXT SCENARIO. RESULTS IMPROVE CONSIDERABLY WHEN USING CLASS SCORE FUSION BUT NOT WHEN USING CONCATENATION FUSION.

Model	mAP@0.5IoU
(Groundtruth Context) Concatenation Fusion	5.92%
(Groundtruth Context) Class Score Fusion	**9.11%**

The results of the real-case scenario are shown in Table VIII where similarly to the last experiment we can see that using FC fusion does not seem to improve the model, and we believe the same reason applies.

For the two pass scheme we can see that there is a small improvement of 0.30% mAP. While the improvement may be considered small when comparing to the ground-truth scenario improvement of 3.16% mAP, consider that the context features in this case are being generated from a classifier with only 5.94% mAP while in the other case we could theorize an ideal classifier with 100% mAP. This small improvement is due to the influence of context to force the model to make valid guesses on under represented small classes (like *sit, walk, hold an object*) at the expense of losing AP in the largest *stand* class, thus confirming the effect of negating overfitting to a certain extent we also witnessed in the last experiment. Additionally, using the results of the context fusion to yet generate more context features and test again (i.e Three Pass in Table VIII) only seemed to yield marginal improvements.

TABLE VIII
TESTING CONTEXT FUSION ARCHITECTURES UNDER A REAL CASE SCENARIO, NO ASSUMPTION OF HAVING THE GROUNDTRUTH TEST LABELS AT TESTING TIME. RESULTS OF CLASS SCORE FUSION WITH A TWO PASS AND THREE PASS SCHEME DEMONSTRATED IN FIG. 15.

Model	mAP@0.5IoU
(Two Pass) Concatenation Fusion	5.91%
(Two Pass) Class Score Fusion	**6.24%**
(Three Pass) Class Score Fusion	**6.28%**

F. Balancing via oversampling

In the literature it is referenced that most multi-label datasets suffer from a high level of imbalance [37] since the number of positive training instances with respect to each class label for many labels is far less than its negative counterparts, which may lead to performance degradation for multi-label learning techniques.

A method of counteracting this imbalance is oversampling which is the process of replicating randomly selected samples from minority classes until the dataset is balanced. We chose this option as [38] mention it tends to be the most effective.

Simply applying naive oversampling would lead to disrupting the distribution as mentioned by [17]. To solve this our oversampling repeats all labels in the segments that contain under represented classes. This means that over represented classes will probably also be repeated but the relative imbalance will be reduced. With this method of oversampling the training set can get considerably larger and thus training time becomes considerably longer, we chose to use only a demonstrative study of training a single RGB stream using the Gaussian Background Blur attention filter, the results of which are shown in Fig. 16. While the overall mAP results (4.71% vs the original 5.63%) seem to indicate that the approach was not successful there is an important observation that supports that this result is misleading. On a further analysis if we analyze Fig. 16 we can see that while oversampling loses AP in more common classes like *stand* (refer to Fig. 5 to see which classes are more frequent) or *talk to*, we can see that for many others it outperforms the classifier without oversampling (for example, *sit*, *hold an object*, *walk* and *bow at the waist*). Interestingly enough, these improvements happen on all three types of classes which validates our approach even further. Therefore, while the results do not seem positive we can see that the balancing is indeed working as intended, however, since the test set itself is imbalanced the reported results are that the classifier performs more poorly.

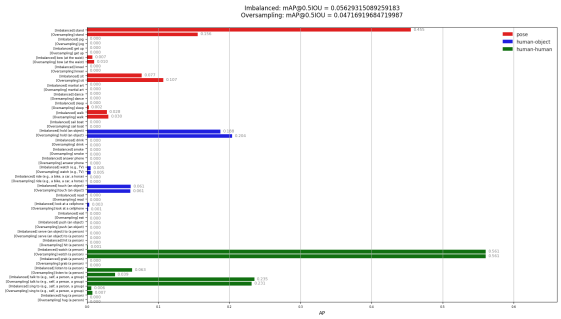


Fig. 16. AP per class on the miniAVA split for the balancing and oversampling experiment. The model used for the oversampling experiment is the same and the repeated frames are all GBB filtered so that the only difference between the two is the oversampling strategy.

V. CONCLUSION

We proposed several architectures, from the initial two stream architectures using only attention filtered inputs as shown in Fig. 10 to our final approach shown in Fig. 15 where we fuse our custom context features with the two-stream approach. We demonstrated improvements on our baseline for

each of the proposed improvements using the aforementioned architectures. Although these results may be deemed provisional, as due to computational resource constraints we had to make our own partition of the larger AVA [9] dataset, we believe it is necessary to conduct further investigation and further testing to truly validate them, not only on the full AVA [9] but also on smaller balanced datasets like [10]. Nonetheless, we believe our work stands on its own as a valid proof of concept for further research on the topic of using alternative human interaction context features and preprocessing attention filtering in the field of action detection and recognition.

REFERENCES

[1] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence - Volume 3*, ser. IAAI'05. AAAI Press, 2005, pp. 1541–1546.

[2] R. K. Tripathi, A. S. Jalal, and S. C. Agrawal, "Suspicious human activity recognition: a review," *Artificial Intelligence Review*, vol. 50, no. 2, pp. 283–339, Aug 2018.

[3] S. Abu-El-Haija, N. Kothari, J. Lee, A. P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," in *arXiv:1609.08675*, 2016.

[4] R. A. Schermerhorn, "Midwest and its children. by roger g. barker and herbert f. wright. evanston, illinois: Row, peterson and company, 1955. 532 pp. 7.50. illustrated," *Social Forces*, vol. 34, no. 4, pp. 390–391, 1956.

[5] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, pp. 90–126, 2006.

[6] M. Wray, D. Moltisanti, and D. Damen, "Towards an unequivocal representation of actions," *CoRR*, vol. abs/1805.04026, 2018.

[7] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, "Hollywood in homes: Crowdsourcing data collection for activity understanding," *CoRR*, vol. abs/1604.01753, 2016.

[8] M. Monfort, B. Zhou, S. A. Bargal, A. Andonian, T. Yan, K. Ramakrishnan, L. M. Brown, Q. Fan, D. Gutfreund, C. Vondrick, and A. Oliva, "Moments in time dataset: one million videos for event understanding," *CoRR*, vol. abs/1801.03150, 2018.

[9] C. Gu, C. Sun, S. Vijayanarasimhan, C. Pantofaru, D. A. Ross, G. Toderici, Y. Li, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik, "AVA: A video dataset of spatio-temporally localized atomic visual actions," *CoRR*, vol. abs/1705.08421, 2017.

[10] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[11] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: a large video database for human motion recognition," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[13] C. Bregler, "Learning and recognizing human dynamics in video sequences," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1997, pp. 568–574.

[14] L. Wang, Y. Qiao, and X. Tang, "Mining motion atoms and phrases for complex action recognition," in *2013 IEEE International Conference on Computer Vision*, Dec 2013, pp. 2680–2687.

[15] S. Herath, M. T. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," *CoRR*, vol. abs/1605.04988, 2016.

[16] S. Kang and R. P. Wildes, "Review of action recognition and detection methods," *CoRR*, vol. abs/1610.06906, 2016.

[17] K. Sozykin, S. Protasov, A. Khan, R. Hussain, and J. Lee, "Multi-label class-imbalanced action recognition in hockey videos via 3d convolutional neural networks," *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 146–151, 2018.

[18] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," *CoRR*, vol. abs/1705.06950, 2017.

[19] D. Tran, J. Ray, Z. Shou, S. Chang, and M. Paluri, "Convnet architecture search for spatiotemporal feature learning," *CoRR*, vol. abs/1708.05038, 2017.

[20] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4724–4733, 2017.

[21] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, pp. 568–576.

[22] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal residual networks for video action recognition," in *NIPS*, 2016.

[23] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, "Temporal segment networks for action recognition in videos," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[24] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ser. ICCV '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 3551–3558.

[25] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "C3D: generic features for video analysis," *CoRR*, vol. abs/1412.0767, 2014.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[27] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, 2018.

[28] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," *CoRR*, vol. abs/1604.06573, 2016.

[29] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l1 optical flow," in *Proceedings of the 29th DAGM Conference on Pattern Recognition*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 214–223.

[30] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 7445–7454.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[32] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[33] A. F. Almeida, R. Figueiredo, A. Bernardino, and J. Santos-Victor, "Deep networks for human visual attention: A hybrid model using foveal vision," in *ROBOT 2017: Third Iberian Robotics Conference*, A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, Eds. Cham: Springer International Publishing, 2018, pp. 117–128.

[34] V. J. Traver and A. Bernardino, "A review of log-polar imaging for visual perception in robotics," *Robotics and Autonomous Systems*, vol. 58, pp. 378–398, 2010.

[35] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532–540, April 1983.

[36] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao, "Towards good practices for very deep two-stream convnets," *CoRR*, vol. abs/1507.02159, 2015.

[37] M.-L. Zhang, Y.-K. Li, and X.-Y. Liu, "Towards class-imbalance aware multi-label learning," in *IJCAI*, 2015.

[38] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *CoRR*, vol. abs/1710.05381, 2017.