Extraction and Classification of Diving Clips from Continuous Video Footage

Aiden Nibali[1] Zhen He[1] Stuart Morgan[1,2] Daniel Greenwood[2]
[1] La Trobe University, [2] Australian Institute of Sport

Abstract

Due to recent advances in technology, the recording and analysis of video data has become an increasingly common component of athlete training programmes. Today it is incredibly easy and affordable to set up a fixed camera and record athletes in a wide range of sports, such as diving, gymnastics, golf, tennis, etc. However, the manual analysis of the obtained footage is a time-consuming task which involves isolating actions of interest and categorizing them using domain-specific knowledge. In order to automate this kind of task, three challenging sub-problems are often encountered: 1) temporally cropping events/actions of interest from continuous video; 2) tracking the object of interest; and 3) classifying the events/actions of interest.

Most previous work has focused on solving just one of the above sub-problems in isolation. In contrast, this paper provides a complete solution to the overall action monitoring task in the context of a challenging real-world exemplar. Specifically, we address the problem of diving classification. This is a challenging problem since the person (diver) of interest typically occupies fewer than 1% of the pixels in each frame. The model is required to learn the temporal boundaries of a dive, even though other divers and bystanders may be in view. Finally, the model must be sensitive to subtle changes in body pose over a large number of frames to determine the classification code. We provide effective solutions to each of the sub-problems which combine to provide a highly functional solution to the task as a whole. The techniques proposed can be easily generalized to video footage recorded from other sports.

1. Introduction

Extracting useful information from video data has become more important in recent years due to the increasing abundance of video data and the low cost of data storage. Much research in this area is compartmentalized into either solving action recognition and classification [28, 38, 14, 11, 46], where the algorithm predicts a discrete class label of actions, or object tracking [24, 2, 49], where continuous pixel coordinates are predicted through time. However, applications in the sports domain often require both problems to be solved together in order to formulate a useful system. For example, isolating individual goal attempts made by a row of training football players in order to find problematic technique, or labeling the actions performed by a gymnast when there are other people moving in the background, or extracting and separating every forehand/backhand for one player in a game of tennis. In each of these examples the person of interest may only occupy a small region of the input frame, there are other people not of interest within the frame, and an accurate understanding of their actions requires an evaluation over an arbitrary temporal span. This aspect of the problem in particular requires a novel approach to learning. We refer to these types of problems as action monitoring problems.

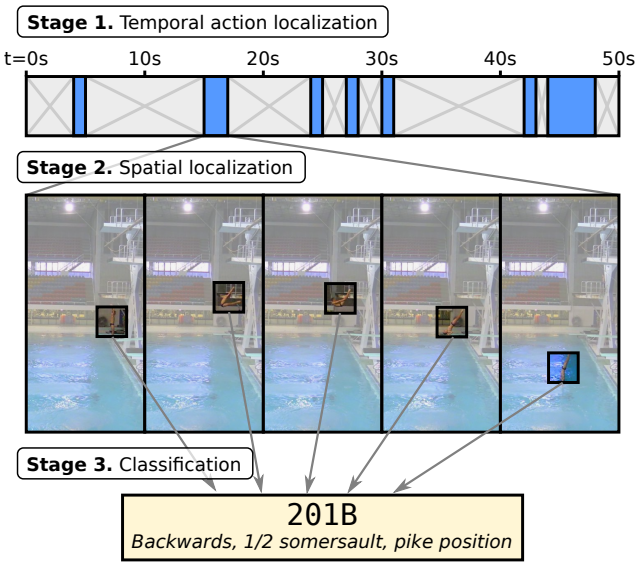


Figure 1: Our action clip extraction and classification system. Each stage drills deeper into the data.

Solving the action monitoring problem requires solutions to the following three sub-problems: 1) temporally cropping events/actions of interest from continuous video; 2) tracking the person/animal of interest; and 3) classifying the event/action of interest. Due to a lack of publicly-available action monitoring data sets, this paper primarily

focuses on solving the diving monitoring problem using a novel data set provided by the Australian Institute of Sport, as illustrated in Figure 1. A solution requires that we first identify the temporal bounds of each dive. We then track the diver of interest to generate suitable spatial crops. Finally, we need to feed the cropped images into a classifier. The solutions presented have general application in the sports domain, and our approach can be applied to solve many other action monitoring problems. The diving monitoring problem is particularly hard since the diver occupies a very small percentage of each frame (typically fewer than 1% of the pixels) and there are thousands of possible different dive codes. So, using just a few pixels per frame we need to consistently separate different dive types which differ only on subtle changes in diver pose. In addition, the system needs to look at the entire dive (which spans around 50-100 frames) to correctly assign a classification code. In contrast, in most public video classification data sets the vast majority of the classes can be assigned by just looking at 1 frame of a video clip (*e.g.* playing tennis versus playing basketball) [15].

We present a 3D convolutional neural network based solution for all three sub-problems of temporal action localization, object tracking, and action recognition. For temporal action localization, we predict the probability that a frame is from the start, middle, and end of a dive. This gives us higher confidence that a dive is correctly detected since all three labels must be detected in sequence. The results show we can correctly extract 98% of dives, with a 26% higher F_1 score than a straightforward baseline approach. For object tracking we present a segmentation based solution to finding the center of a diver in each frame. The results show our segmentation based solution is appreciably more accurate than a more conventional regression based solution. Finally, our proposed classification approach based on dilated convolutions can achieve an average of 93% accuracy for each component of the dive codes.

2. Related Work

Video representation and classification At the heart of video analysis is the way the data is represented. Many techniques extend 2D image representations to 3D by incorporating the temporal dimension, including HOG3D [16] from HOG [3], extended SURF [43] from SURF [1], and 3D-SIFT [28] from SIFT [22]. Other techniques such as optical flow treat the temporal dimension as having properties distinct from spatial dimensions. The work on dense trajectories proposed by Wang *et al*. [38] takes such an approach, and is currently a state-of-the-art hand-crafted feature algorithm for video analysis. Unfortunately, the effectiveness of optical flow-based techniques (including dense trajectories) comes at the price of computational efficiency, which reduces their viability for real-time applications and large-scale datasets.

Using learnt features via convolutional neural networks (CNNs) for video analysis have become more popular since the huge success of AlexNet [17] in the ILSVR 2012 image classification challenge. One of the directions this research took was in finding direct ways of applying 2D CNNs to video data by fusing 2D feature maps at different levels of the network hierarchy. Karpathy *et al*. [15] demonstrated that such fusion schemes only achieve a modest improvement over using only a single frame of input. Another direction taken was to treat video as 3D data (with time being the 3rd dimension), and apply volumetric convolutions [36]. Such networks learn good representations of video data at the cost of a large memory requirement.

There exist multiple more complex solutions for applying CNNs to action recognition [31, 44, 4, 47]. Some of these solutions rely on optical flow [31, 47], which is slow to evaluate. Others rely on a recurrent architecture [44, 4], which is often difficult to train in practice.

Temporal action localization The dominant method for detecting the temporal extent of actions involves sliding windows of several fixed lengths through the video, and classifying each video segment to determine whether it contains an action [25, 42, 30]. The segment classifier can be based on hand-engineered feature descriptors [25], trained CNNs [30], or a combination of the two [42]. In contrast to this segment-based approach, we are able to detect actions of arbitrary length without sliding multiple windows through the video.

Other branches of work related to temporal action localization attempt to solve different variations of the problem, such as detecting temporal extents without explicit temporal annotations [19, 18, 32], or simultaneously detecting temporal and spatial boundaries [13, 8].

Object localization/detection Sermanet *et al*. [29] proposed a neural network called OverFeat for object detection. OverFeat comprises of a convolutional feature extractor and two network "heads" - one for classification, and another for regression. The feature extractor is similar to what is now commonly referred to as a fully-convolutional network. This allows it to efficiently slide a window around the image to extract features for different crops. The classifier is a multi-layer perceptron which takes features from the feature extractor as input and predicts a class as output. This tells us what is in each crop (including confidence), and is already sufficient to produce coarse bounding boxes. However, these boxes are refined further by training a class-specific regression head which outputs bounding box dimensions from image features.

Girshick *et al*. [7] proposed a different strategy called R-CNN (regions with CNN features). They use an existing

algorithm (*e.g*. Selective Search [37] or EdgeBoxes [50]) to produce region proposals, and warp the region of the image described by each proposal to a fixed size. The warped image is run through a CNN, the output features of which are used to prune the proposed regions and generate final predictions. There now exist more efficient works based on R-CNN which improve evaluation time [6, 27].

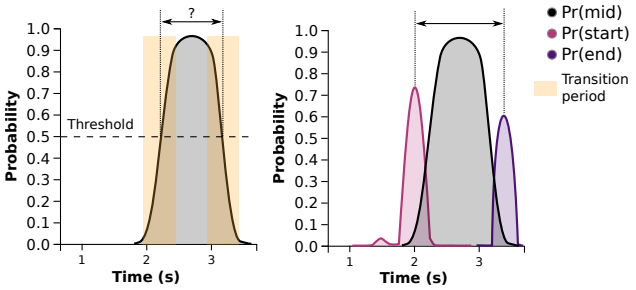
Szegedy *et al*. [34] proposed a segmentation-style approach to object detection. Rather than dealing with region proposals or output coordinates, the network takes the entire image as input and produces a lower resolution "mask" depicting filled-in bounding boxes at the output. The results reported in the paper are considerably worse than R-CNN, but we note that this system is a more natural fit for localization than detection due to complications introduced by overlapping bounding boxes.

Tracking There has been a lot of research in the area of object tracking. In this section we will focus on CNN based solutions [39, 21, 10, 20, 41, 40, 23]. They all take the approach of tracking-by-detection, where a binary classifier is applied to positive and negative samples from each frame. Typically, the object bounding box of just the first frame is provided and the CNN models learnt in an online manner. All methods need to somehow deal with the small number of labeled training samples. [41] pretrains the network using an autoencoder, [39, 10, 40, 23] uses CNNs pretrained on the large ImageNet dataset and [21, 20] uses special loss functions and sampling techniques to cope with the small number of training samples. In contrast to most existing work, our solution first finds location candidates for each frame and then applies global constraints to create the motion trajectory, which is used to provide smoothly tracked output that aids the next stage of the system.

3. Overview

At a high level, our dive detection and classification system consists of three distinct stages, as shown in Figure 1. Each stage uses a convolutional neural network at its core. Firstly, we extract individual video clips of dives from continuous video footage. Secondly, we localize the diver within each frame of the clip to produce a tracking shot of the dive, which allows us to improve the ratio of pixels in the clip which are useful for classification. Thirdly, we use the tracked clip to predict the dive code using a classification network.

The three stages of the system are linked by dependencies on preceding stages. There are a few places where these dependencies led to different design decisions from considering each stage in isolation. For instance, during spatial object localization we fit the motion trajectory by applying global constraints, and crop the images using a fixed-size box to keep the scale consistent between frames. This smooth tracking increases the accuracy of the classifier.



(a) Single probability signal (b) Three probability signals

Figure 2: (a) It's difficult to tell precisely when a dive starts and ends from the middle event probability only due to the transition periods, (b) whereas the start and end probabilities give more obvious time markers.

4. Temporal action localization

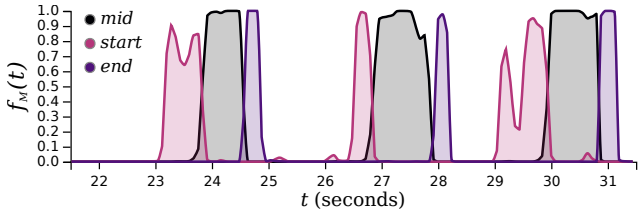
The first stage of the system involves extracting action clips from continuous footage, a task known as *temporal action localization*. Our aim is to predict the temporal extent of each dive as accurately as possible in order to crop the extracted clip tightly, thus maximizing the number of frames which are relevant for classification. Hence our network needs to be able to indicate the start and end times of dives in a dynamic way. This differs from the existing temporal action localization work with CNNs, which slide windows with one of several predetermined lengths through the video [42, 30].

We explicitly identify three event states in the video footage: a diver leaving a platform ($start$), a diver entering the water (end), and any time during which the diver is airborne (mid). Our temporal action localization neural network (TALNN) accepts 21 frames of video as input, and outputs probabilities for the center frame containing each of these events. These probabilities are predicted as independent values (*i.e*. they are not part of a single softmax), which allows the network to output high probabilities for two events at once (*e.g*. start and middle). The network itself is built from volumetric convolutional layers, with one head per probability signal. Table 1a specifies the architecture in detail. Each convolutional layer in the body is followed by batch normalization and a ReLU non-linearity.

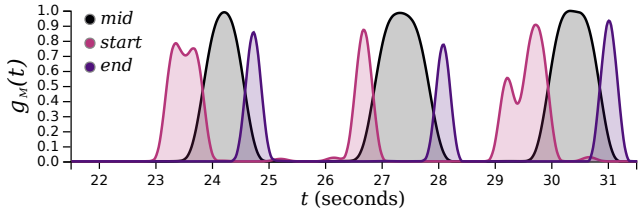
Let $\boldsymbol{x}_t$ be the 21-frame window centered at time t. We now define the time varying probability signals $f_M(t)$ as in Equation 1, where $M \in \{start, mid, end\}$.

$$f_M(t) = \Pr(M|\boldsymbol{x}_t) \tag{1}$$

At first it may seem unusual that we are considering the start and end events at all, since the boundaries of the middle



(a) Raw probability signals



(b) Smoothed probability signals

Figure 3: Raw and smoothed probability signals for a section of video footage containing three dives.

event should be sufficient to determine when a dive starts and ends. Figure 2a shows the problem with that approach. Namely we would need to select some threshold (*e.g.* 0.5) as to when the start and end boundaries are defined. In contrast, Figure 2b shows that using all three events (start, middle, and end) makes finding the start and end of the dive less ambiguous. Note that we could theoretically remove $f_{mid}(t)$ altogether, but we opt to keep it as a way of reducing the likelihood of false positives.

After training the TALNN to identify the different types of events, $f_M(t)$ is obtained by sliding a 21-frame window through the video and evaluating the network. Figure 3a shows that although the output provides a strong indication of when dives occur, it is not perfectly smooth.

Smoothing To make the peaks in the probability signals more pronounced we process them further into smoothed probability signals, $g_M(t)$ (Figure 3b). This makes the final dive extraction step more robust. A common way of smoothing signals is to apply a window function, as in Equation 2.

$$g_M(t) = \frac{\int_{-\infty}^{\infty} f_M(\tau) w(\tau - t + T/2) d\tau}{\int_{-\infty}^{\infty} w(\tau) d\tau} \quad (2)$$

We use the Hann window function (Equation 3) for smoothing, which gives us the formula for calculating $g_M(t)$ described in Equation 4.

$$w(t) = \begin{cases} \sin^2\left(\frac{\pi t}{T}\right) & \text{if } 0 \leq t \leq T \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Body	Head
conv3-32, strided	conv1-1
conv3-32	avgpool
conv3-64, strided	sigmoid
conv3-64	
conv3-128, strided	
conv3-128	
conv3-256, strided	
conv3-256	

(a) Temporal localization

Kernel	Dilation	Maps
3x3x3	1x1x1	2
3x3x3	1x1x1	2
3x3x3	2x2x2	4
3x3x3	4x4x4	4
3x3x3	8x8x8	4/8
3x3x3	1x16x16	8/16
3x3x3	1x1x1	8/16
1x1x1	1x1x1	1/3

(b) Spatial localization context net

Table 1: CNN localization architectures.

$$g_M(t) = \frac{2}{T} \int_{t-T/2}^{t+T/2} f_M(\tau) \sin^2\left(\frac{\pi(\tau - t)}{T} + \frac{\pi}{2}\right) d\tau \quad (4)$$

Extraction Given the three smoothed probability signals, we can apply a simple algorithm to extract concrete dive intervals. Firstly, identify candidate dives by locating peaks in $g_{mid}(t)$. Secondly, perform a limited scan forwards and backwards through time (we use 1 second) to locate the dive's start and end from peaks in their respective probability signals. If there are no strong nearby peaks in the start and end probability signals, discard the dive candidate.

5. Spatial localization

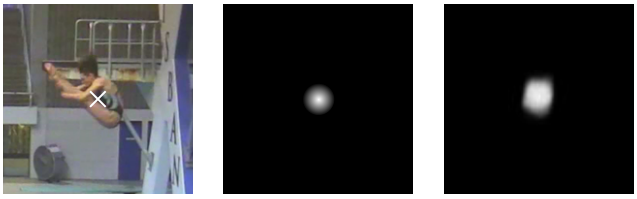
The aim of the spatial localization stage is to produce a trajectory consisting of the diver centroid in each frame of the input clip. Given a list of centroids we can then take a fixed size crop from each frame to produce a tracking clip, which will supply the classifier with fixed size input of a consistent scale that excludes most of the background.

We take a tracking-by-detection approach to spatial localization, which is separated into two steps. The first step is to find object location candidates which indicate potential locations for the diver in each frame. The second step is to take these candidate locations and apply global constraints to construct a motion trajectory.

5.1. Object location candidates

Here we compare three possible solutions to the object location candidate proposal step which we refer to as full regression, partial regression, and segmentation.

Full regression Perhaps the most straightforward approach to spatial localization is to take a complete video clip as input, and attempt to train a network which outputs the object location coordinates (l_x, l_y) directly for each frame. We call this approach "full regression". One advantage of



(a) Input (b) Target (c) Output

Figure 4: Using a "hot-spot" for localization via segmentation. Diver location is marked on input for reference only.

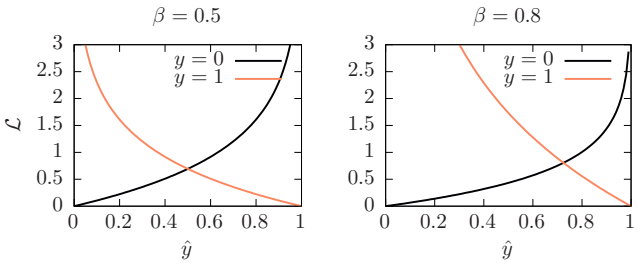


Figure 5: Side-by-side comparison of standard BCE (left) and weighted BCE (right).

full regression is that it gives a single location per frame, which removes the need for a second step to construct the motion trajectory. In practice we found full regression to yield very poor accuracy of predicted locations, with a high amount of location "jitter" between neighboring frames.

Partial regression An alternative to full regression is to only consider a small crop of the input at a time (an "input patch"), and train a network to predict whether the object is contained in the patch. The network is also trained to output its location relative to the input patch's frame of reference, though patches which do not contain the object exclude the location from loss calculations. This is an approach used successfully in prominent object detection systems including OverFeat [29] and Fast R-CNN [6].

The network used in this paper for partial regression is a stack of two context networks [45] followed by an average pooling layer. Table 1b specifies our configuration for the context networks. The use of dilated convolutions improves the scale invariance of the network, which helps with the fact that divers are at different distances from the camera. Furthermore, we were able to construct the network with very few feature maps, resulting in a compact model.

An important aspect of our implementation of partial regression is that the network is fully convolutional. This means that at inference time we can provide the entire image as input (rather than patches). The network will then implicitly slide a window through the image, but do so in a way which shares common intermediate activations. This is much more efficient than explicitly making overlapping crops and feeding them through the network separately. The overlap of the windows can be adjusted by altering the stride of the average pooling layer.

Segmentation We observe that going from full to partial regression resulted in much more accurate location candidates, and that a key difference is that the latter places less emphasis on regressing coordinates. We decided to take a step further in this direction and eliminate regression completely, which is achieved by reframing the problem as a segmentation problem. Instead of using numeric coordinates as the target, we artificially generate target images for each frame where the location of the diver is indicated with a fixed-size "hot-spot" (Figure 4b). The network learns to output blob-like approximations of these hot-spots (Figure 4c) which can then be converted into centroids using existing techniques for blob detection [26]. The main advantage of this approach is that it unburdens the network of transforming spatial activations into numeric coordinates.

The segmentation style of temporal localization has an imbalance in the output, as the hot-spot occupies a small portion of the patch. With a traditional loss function like binary cross-entropy (BCE), this makes the prediction of all zeros an attractive behavior for the network to learn in terms of loss minimization. To counteract this, we modified BCE to weight positive outputs higher, thus penalizing the network more harshly for ignoring them (Equation 5).

$$\mathcal{L} = \frac{-\log(\hat{y})y}{2(1-\beta)} + \frac{-\log(1-\hat{y})(1-y)}{2\beta} \tag{5}$$

When $\beta = 0.5$, weighted BCE is equivalent to the usual BCE formulation. When $\beta \in (0.5, 1)$, the positive example term of the loss function is weighted higher. Figure 5 illustrates how weighted BCE imposes a greater loss for misclassified positive examples than negative ones when $\beta > 0.5$. We found $\beta = 0.8$ to work well in practice.

As with partial regression, we train the segmentation network on input patches. The network architecture is similar, the main difference being that the average pooling layer is removed and second context network adjusted such that there is 1 output per pixel.

5.2. Global constraints

Neither the partial regression nor the segmentation approach is able to produce a proper motion trajectory alone, as there can be many (or zero) locations output for each frame. We get around this by using a second step which applies global constraints to refine the location candidates and produce a motion trajectory. Ultimately this produces one location per frame to center the crop on when constructing a tracking clip. During this step bad location candidates are rejected and missing locations are interpolated.

Algorithm 1 Creating a motion trajectory model.

function CREATEMODEL(ts[], xs[], ys[])
  $a_0, a_1 \leftarrow LinearRegression(ts, xs)$
  $b_0, b_1, b_2 \leftarrow QuadraticRegression(xs, ys)$
  function MODEL(t)
    $x \leftarrow a_0 + a_1 t$
    $y \leftarrow b_0 + b_1 x + b_2 x^2$
    return x, y
  return Model

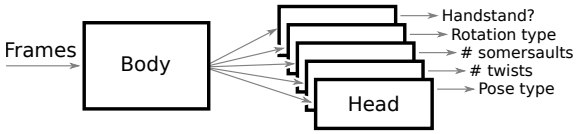


Figure 6: High-level view of the classifier architecture with a head for each part of the dive code.

The appropriate constraints to apply when constructing a motion trajectory will depend on the problem. For diving, we have the ability to apply very strong constraints derived from basic kinematic formulae. In fact, we can go so far as to specify a model for the trajectory which has only five parameters – two for a linear mapping from time to horizontal location, and three for a quadratic mapping from horizontal location to vertical location. Algorithm 1 describes how the model is constructed. Once we have a known model we can use the RANSAC [5] algorithm to find the instance of the model which best fits the location candidates. RANSAC is an iterative algorithm which fits the data by repeatedly creating model instances for random subsets of points and selecting whichever one fits the complete set of points best. The main benefit of using RANSAC is that it is very robust to contamination from outliers, and is therefore able to ignore bad location candidates. In practice we used the improved MSAC [35] variant of RANSAC which generally fits the model in fewer iterations.

For less tightly constrained problems, an alternative method for constructing the motion trajectory must be employed. Although we did not explore this space ourselves, one approach would be to use local feature descriptors to track candidate locations through time.

6. Classification

Classifying dives involves outputting a five-part code, where each part represents a different property of the dive. An example of a dive code is `201B`, where the 2 means backwards rotation, the 1 means one half-somersault, the B means pike position, and the overall code implies that there are no twists and no handstand start. We could try to classify the entire code using a single output representation, which equates to a 1-in-k classification problem where k is the number of combinations of all properties. This would result in thousands of possible output classes, most of which would have just a few or zero training examples.

C3D	C3D (alt.)	Dilated
Body		
conv3-64	conv3-32, BN	conv3-32, BN
1x2x2 maxpool	1x2x2 maxpool	1x2x2 maxpool
conv3-128	conv3-64, BN	conv3-64, BN
2x2x2 maxpool	2x2x2 maxpool	2x2x2 maxpool
conv3-256 (×2)	conv3-128, BN (×2)	conv3-128, BN (×2)
2x2x2 maxpool	2x2x2 maxpool	2x2x2 maxpool
conv3-512 (×2)	conv3-256, BN (×2)	conv3-256, BN (×2)
2x2x2 maxpool	2x2x2 maxpool	-
conv3-512 (×2)	conv3-256, BN (×2)	conv3-d2-256, BN (×2)
2x2x2 maxpool	2x2x2 maxpool	-
-	dropout-0.5	dropout-0.5
Head		
fc-4096	fc-2048, BN	conv1-12
dropout-0.5	-	context net
fc-4096	fc-2048, BN	2x2x2 maxpool
dropout-0.5	-	conv3-12, BN
fc-output	fc-output	conv3-output, avgpool

Table 2: Architectural differences between vanilla C3D and our variations used for classification.

Instead, we propose using multi-task learning consisting of a single network with 5 heads, each outputting a separate property (Figure 6). One way to reason about the architecture is that the network body learns to extract features from the input which are relevant for predicting one or more parts of the dive code. The heads take these features and use them to predict a particular part of the dive code. Our hypothesis is that some intermediate features can be shared between heads, making it easier for the network to rule out unlikely dive code combinations. We use a deep convolutional network for the model body, and multi-layer perceptrons for the heads. The internal structure of each head is identical, except for the number of outputs.

The classification network takes tracked video clips as input. Since the clips are now cropped around the diver, we can use a higher resolution than the previous networks under the same memory constraints. To keep the input size constant we always temporally downsample the clip to a length of 16 frames, which we verified is sufficient to solve the classification task as a human annotator.

In this paper we consider three classifier architectures (Table 2), all of which are based on the "C3D" volumetric convolutional network proposed by Tran *et al.* [36]. The first is a direct implementation of the C3D architecture which follows the original work closely. All layers up until and excluding the first fully connected layer form the model body, and the rest form a head. The second is an altered

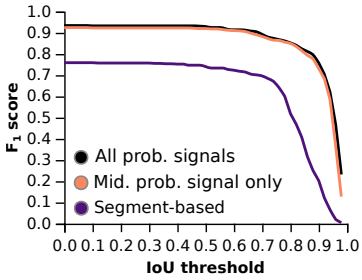


Figure 7: Temporal action localization results as the IoU threshold was varied.

version of C3D which makes room for batch normalization (BN) [12] by halving the number of features throughout the network. The third architecture introduces dilated convolutions for scale invariance [45]. Pooling in the latter half of the body is removed, and the last two convolutional layers given a dilation of 2 (conv3-d2) to maintain receptive field size. A context network [45] with layers 5 and 6 removed is introduced into the head for multi-scale aggregation.

7. Data set

The data set consists of 25 hours of video footage containing 4716 non-overlapping sport dives. The video was recorded over 10 days of athlete training at the Brisbane Aquatic Centre. The scene is observed from the perspective of a fixed camera which has 9 platforms and springboards at varying heights and distances in view. Each dive is labeled with a start and end time, along with a code representing the type of dive performed. 20% of the dives are also labeled with a quadratic curve describing the location of the athlete in each frame of the dive. An additional day's worth of footage containing 612 dives is kept aside as the test set.

The dive code encodes 5 distinct properties of the dive: rotation type, pose type, number of somersaults, number of twists, and whether the dive began with a handstand. These properties are not all represented uniformly in the data set. For instance, dives involving twists are uncommon, and dives starting with a handstand are even rarer.

8. Experiments

8.1. Temporal action localization

As a point of comparison we implement a segment-based temporal action localization method based on the work of Shou *et al*. [30]. We use a single C3D-based network to directly predict how well a particular segment matches any sort of dive. We incorporate batch normalization into the network in the same way as the classification network, and do not perform any pretraining. Although we did not explicitly gather time metrics, we will note that performing inference on multiple segment lengths did make the segment-based system very slow to evaluate.

	Precision	Recall	F_1 score
Segment-based [30]	0.7671	0.7157	0.7405
Ours	**0.8825**	**0.9829**	**0.9296**

Table 3: Action clip extraction results.

With the segment-based approach established as a baseline, we consider two of our own approaches to temporal action localization as discussed in Section 4. The first approach uses only a single probability signal indicating the middle of a dive (Figure 2a), with the transition threshold set to 0.5. The second approach uses three probability signals for the start, middle, and end (Figure 2b). Each network was trained to convergence using ADADELTA [48]. A predicted dive interval is deemed "correct" if it matches a labeled dive interval with an IoU (intersection over union) above a certain threshold. A "false positive" is a predicted interval without a corresponding labeled dive, and a "false negative" is a labeled dive not predicted by the system.

Figure 7 shows a plot of the F_1 score for the different approaches as the IoU threshold was varied. Both of our approaches (all probability signals and middle probability signal only) perform much better than the segment-based approach, which is unable to reach an F_1 score of 0.8 for any IoU threshold. Although the performance of our own two approaches are similar, we advocate using all three signals since doing so shows slightly better results, and in other situations it may be more difficult to threshold the middle probability signal.

Table 3 shows the precision, accuracy, and F_1 score for our three-signal approach and the segment-based approach, with the IoU threshold set to 0.5. At first it seems as if the precision of the TALNN is much worse than its recall. However, upon examining the false positives it was found that the vast majority did in fact contain dives that were simply not labeled in the data set. During our manual inspection of the false positives we did not find a single example that wasn't a labeling mistake. On the other hand, dives which were missed by the TALNN were mostly legitimate oversights, with dives from the furthest springboard being the most common culprit.

The results of the TALNN stage are very convincing, and provide a solid starting point for the rest of the system. The segment-based approach does not achieve performance metrics which are as strong. We believe that the main reason for this is the fixed segment lengths – any dive which does not perfectly match a segment length will inevitably incur error from the difference.

8.2. Spatial localization

Each network was trained to convergence using ADADELTA [48]. The hot-spot to location conversion for

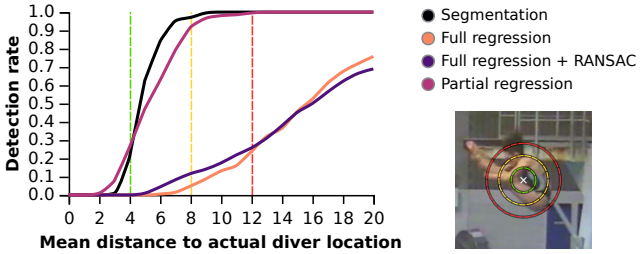


Figure 8: Spatial localization network results. Three distances are marked on a video frame for reference.

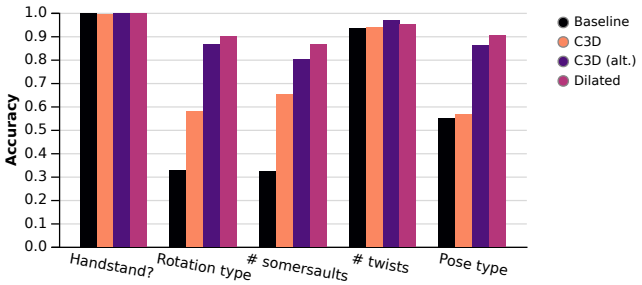


Figure 9: Classifier network accuracy.

the segmentation approach was handled using OpenCV's blob detector [26], which leverages the contour finding algorithms proposed by Suzuki *et al*. [33].

Figure 8 shows, for a range of distance error thresholds, the percentage of dive clips that had a mean error distance below that threshold. Closer to the top-left is better, as this indicates high detection rate within a strict distance limit. The results show just how poorly the full regression approach performs, even when global constraints are applied using RANSAC. Upon inspecting individual examples, we found that the full regression network often seemed to ignore subtleties of the current dive instance in favor of some learnt statistical average across the training set location labels. The margin between the partial regression approach and our novel segmentation approach is less pronounced, but shows that segmentation does indeed work best.

In practice we found that partial regression resulted in many more candidate locations than segmentation. This was not an issue for RANSAC due to its speed and robustness to contamination, but we note that other techniques for constructing motion trajectories may benefit heavily from the reduced number of candidates produced by segmentation.

8.3. Classification

Since the data set does not contain an equal number of examples for each type of dive, we include a baseline to help visualize this skew. The baseline shows the results of always outputting the statistical mode for each part of the dive code. Gains in accuracy above this baseline are indicative of the system's ability to discriminate between classes.

Table 2 specified the architecture of each classifier model. The networks make heavy use of volumetric convolutions with $3 \times 3 \times 3$ kernels and use ReLU non-linearities. Regularization is provided by dropout [9] and, for two of the architectures, batch normalization. Each network was trained until convergence using stochastic gradient descent with a momentum of 0.9 and an initial learning rate of 0.006 (0.003 for vanilla C3D), which is halved every 30 epochs.

	Isolated	Combined
Handstand?	100.00%	99.67%
Rotation type	89.81%	77.54%
# somersaults	86.89%	66.72%
# twists	95.15%	93.51%
Pose type	90.78%	82.36%

Table 4: Combined classification accuracy.

Figure 9 shows accuracy results for the classification networks when isolated from the other stages (*i.e*. using ground truth labels for diver locations). Despite halving the number of feature maps in order to fit batch normalization, we observe that doing so still leads to a marked improvement in accuracy. We suspect that the increased regularization provided by batch normalization is contributing a lot to the performance of the network, as our data set is relatively small in comparison to existing large-scale public image data sets.

Adding dilated convolutions to the altered C3D network resulted in a boost to classification accuracy for all parts of the dive code except the twist count. We theorize that the dilations increase the network's ability to recognize features irrespective of the distance of the diver from the camera.

8.3.1 Combined classification

In order to measure the impact that errors introduced in the temporal and spatial localization stages have on classification, we conducted a combined classification experiment using three-signal temporal action localization, localization by segmentation, and classification with dilations. Table 4 shows that although error from the earlier stages does have a negative impact on classification accuracy, the complete system is still viable.

9. Conclusions

There are challenges involved with composing multiple stages of deep learning computer vision processing together. Using dive classification as a case study, we have demonstrated that such a composite system can be successfully constructed for sports action monitoring of continuous video. Novel techniques for extracting action clips and localizing an object of interest were presented with strong results. As future work we would like to modify our system to assign dive scores like a judge, which is a difficult problem due to the subtle and subjective nature of the task.

References

[1] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded Up Robust Features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006*, number 3951 in Lecture Notes in Computer Science, pages 404–417. Springer Berlin Heidelberg, May 2006. DOI: 10.1007/11744023_32.

[2] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition, 2000. Proceedings*, volume 2, pages 142–149 vol.2, 2000.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[4] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

[5] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[6] R. Girshick. Fast R-CNN. pages 1440–1448, 2015.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. pages 580–587, 2014.

[8] G. Gkioxari and J. Malik. Finding action tubes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 759–768, 2015.

[9] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580 [cs]*, July 2012. arXiv: 1207.0580.

[10] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. *arXiv preprint arXiv:1502.06796*, 2015.

[11] E. P. Ijjina and K. M. Chalavadi. Human action recognition using genetic algorithms and convolutional neural networks. *Pattern Recognition*, 59:199–212, Nov. 2016.

[12] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 448–456, 2015.

[13] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek. Action localization with tubelets from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 740–747, 2014.

[14] S. Ji, W. Xu, M. Yang, and K. Yu. 3d Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, Jan. 2013.

[15] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-Scale Video Classification with Convolutional Neural Networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, June 2014.

[16] A. Klaser, M. Marszałek, and C. Schmid. A Spatio-Temporal Descriptor Based on 3d-Gradients. pages 275:1–10. British Machine Vision Association, Sept. 2008.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, page 2012.

[18] K.-T. Lai, X. Y. Felix, M.-S. Chen, and S.-F. Chang. Video event detection by inferring temporal instance labels. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2251–2258. IEEE, 2014.

[19] K.-T. Lai, D. Liu, M.-S. Chen, and S.-F. Chang. Recognizing complex events in videos by learning key static-dynamic evidences. In *European Conference on Computer Vision*, pages 675–688. Springer, 2014.

[20] H. Li, Y. Li, and F. Porikli. Robust online visual tracking with a single convolutional neural network. In *Asian Conference on Computer Vision*, pages 194–209. Springer, 2014.

[21] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *IEEE Transactions on Image Processing*, 25(4):1834–1848, 2016.

[22] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[23] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3074–3082, 2015.

[24] L. Mihaylova, P. Brasnett, N. Canagarajah, and D. Bull. *Object Tracking by Particle Filtering Techniques in Video Sequences*.

[25] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1817–1824, 2013.

[26] OpenCV. cv::SimpleBlobDetector class reference. Available: http://docs.opencv.org/3.1.0/d0/d7a/classcv_1_1SimpleBlobDetector.html.

[27] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497 [cs]*, June 2015. arXiv: 1506.01497.

[28] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional Sift Descriptor and Its Application to Action Recognition. In *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07, pages 357–360, New York, NY, USA, 2007. ACM.

[29] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv:1312.6229 [cs]*, Dec. 2013. arXiv: 1312.6229.

[30] Z. Shou, D. Wang, and S.-F. Chang. Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs. *arXiv:1601.02129 [cs]*, Jan. 2016. arXiv: 1601.02129.

[31] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[32] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 371–380. ACM, 2015.

[33] S. Suzuki and K. be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, Apr. 1985.

[34] C. Szegedy, A. Toshev, and D. Erhan. Deep Neural Networks for Object Detection. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2553–2561. Curran Associates, Inc., 2013.

[35] P. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78(1):138–156, Apr. 2000.

[36] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning Spatiotemporal Features with 3d Convolutional Networks. *arXiv:1412.0767 [cs]*, Dec. 2014. arXiv: 1412.0767.

[37] J. R. R. Uijlings, K. E. A. v. d. Sande, T. Gevers, and A. W. M. Smeulders. Selective Search for Object Recognition. *Int J Comput Vis*, 104(2):154–171, Sept. 2013.

[38] H. Wang, A. Kläser, C. Schmid, and C. L. Liu. Action recognition by dense trajectories. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176, June 2011.

[39] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3119–3127, 2015.

[40] L. Wang, W. Ouyang, X. Wang, and H. Lu. Stct: Sequentially training convolutional networks for visual tracking. CVPR, 2016.

[41] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *Advances in neural information processing systems*, pages 809–817, 2013.

[42] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. *arXiv:1506.01929 [cs]*, June 2015. arXiv: 1506.01929.

[43] G. Willems, T. Tuytelaars, and L. V. Gool. An Efficient Dense and Scale-Invariant Spatio-Temporal Interest Point Detector. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision – ECCV 2008*, number 5303 in Lecture Notes in Computer Science, pages 650–663. Springer Berlin Heidelberg, Oct. 2008. DOI: 10.1007/978-3-540-88688-4_48.

[44] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. *arXiv preprint arXiv:1511.06984*, 2015.

[45] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. *arXiv:1511.07122 [cs]*, Nov. 2015. arXiv: 1511.07122.

[46] Y. Yuan, L. Qi, and X. Lu. Action recognition by joint learning. *Image and Vision Computing*, 55, Part 2:77–85, Nov. 2016.

[47] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer*

Vision and Pattern Recognition, pages 4694–4702, 2015.

[48] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701 [cs]*, Dec. 2012. arXiv: 1212.5701.

[49] K. Zhang, Q. Liu, Y. Wu, and M.-H. Yang. Robust Visual Tracking via Convolutional Networks. *arXiv:1501.04505 [cs]*, Jan. 2015. arXiv: 1501.04505.

[50] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, pages 391–405. Springer International Publishing, Cham, 2014.