Monitoring Tool Usage in Cataract Surgery Videos using Boosted Convolutional and Recurrent Neural Networks

Hassan Al Hajj\textsuperscript{a}, Mathieu Lamard\textsuperscript{b,a}, Pierre-Henri Conze\textsuperscript{c,a}, Béatrice Cochener\textsuperscript{b,a,d}, Gwenolé Quellec\textsuperscript{a,*}

\textsuperscript{a}Inserm, UMR 1101, Brest, F-29200 France
\textsuperscript{b}Univ Bretagne Occidentale, Brest, F-29200 France
\textsuperscript{c}Institut Mines-Télécom Atlantique, Brest, F-29200 France
\textsuperscript{d}Service d’Ophtalmologie, CHRU Brest, Brest, F-29200 France

Abstract

With an estimated 19 million operations performed annually, cataract surgery is the most common surgical procedure. This paper investigates the automatic monitoring of tool usage during a cataract surgery, with potential applications in report generation, surgical training and real-time decision support. In this study, tool usage is monitored in videos recorded through the surgical microscope. Following state-of-the-art video analysis solutions, each frame of the video is analyzed by convolutional neural networks (CNNs) whose outputs are fed to recurrent neural networks (RNNs) in order to take temporal relationships between events into account. Novelty lies in the way those CNNs and RNNs are trained. Computational complexity prevents the end-to-end training of “CNN+RNN” systems. Therefore, CNNs are usually trained first, independently from the RNNs. This approach is clearly suboptimal for surgical tool analysis: many tools are very similar to one another, but they can generally be differentiated based on past events. CNNs should be trained to extract the most useful visual features in combination with the temporal context. A novel boosting strategy is proposed to achieve this goal: the CNN and RNN parts of the system are simultaneously enriched by progressively adding weak classifiers (either CNNs or RNNs) trained to improve the overall classification accuracy. Experiments were performed in a new dataset of 50 cataract surgery videos where the usage of 21 surgical tools was manually annotated. Very good classification performance are achieved in this dataset: tool usage could be labeled with an average area under the ROC curve of $A_z = 0.9717$ in offline mode (using past, present and future information) and $A_z = 0.9696$ in online mode (using past and present information only).

Keywords: cataract surgery, tool usage monitoring, video analysis, Convolutional Neural Networks, Recurrent Neural Networks, boosting

1. Introduction

Cataract surgery is the most common surgical procedure worldwide: 19 million cataract surgeries are performed annually (Trikha et al., 2013). A cataract is a clouding of the lens, inside the eye, which leads to a decrease in vision. The purpose of the surgery is to remove this lens and replace it with an artificial lens. During the surgery, the patient’s eye is under a microscope and the output of the microscope is video-recorded. With the emergence of imaging devices in the operating room, the automated analysis of videos recorded during the surgery is becoming a hot research topic. Potential applications include report generation, surgical workflow optimization, surgical skill evaluation, surgical training and real-time warning generation (Quellec et al., 2015). One solution to monitor the surgery is to recognize which tools are being used at each time instant. Therefore, several tool detection techniques have been proposed in recent years (Bouget et al., 2017). To compare these techniques, a tool detection challenge was organized at the M2CAI 2016 workshop: the goal was to list the visible tools in each frame of laparoscopic surgery videos.\textsuperscript{1} Following the trend in medical image and video analysis (Shen et al., 2017), the best solutions all relied on convolutional neural networks (CNNs) (Raju et al., 2016; Sahu et al., 2016; Twinanda et al., 2017; Zia et al., 2016).

Compared to other computer vision tasks, surgical tool usage annotation has several specificities. First, as opposed to many computer vision tasks, including the popular ImageNet visual recognition challenges,\textsuperscript{2} the problem at hand is not multiclass classification (one correct label per image among multiple classes), but rather multilabel classification (multiple correct labels per image): the number of tools being used in each image varies (from zero to three in cataract surgery). Therefore, multilabel CNNs should be used. Second, taking the temporal sequencing into account is important: knowing which tools have

\textsuperscript{1}Follow:\url{http://camma.u-strasbg.fr/m2cai2016/index.php/tool-presence-detection-challenge-results/}

\textsuperscript{2}Follow:\url{http://www.image-net.org/challenges/LSVRC/2017/index.php}
already been used since the beginning of the surgery greatly helps recognize which tools are currently being used. Therefore, multilabel recurrent neural networks (RNNs) [Hochreiter and Schmidhuber, 1997] may also be used advantageously. In fact, recent machine learning competitions clearly show that ensembles of CNNs outperform single CNNs [Kussakovskiy et al., 2015; Quellec et al., 2017]. Multiple CNNs with different architectures are generally trained independently and their outputs are combined afterward using standard machine learning algorithms (decision trees, random forests, multilayer perceptrons, etc.). However, this simple strategy is suboptimal since difficult samples may be misclassified by all CNNs. And there are many difficult samples to classify in surgery videos: in particular, many tools resemble one other (e.g. two types of cannulae in cataract surgery). Building the ensemble of CNNs using a boosting meta-algorithm can theoretically design CNNs focusing specifically on challenging samples. Boosting an ensemble of RNNs would also make sense as there are difficult samples along the time dimension as well: in particular, some tools or tool usage sequences are very rare and temporal sequencing algorithms tend to misclassify those rare cases. Therefore, we propose to jointly boost an ensemble of CNNs and an ensemble of RNNs for automatic tool usage annotation in cataract surgery videos. In the same way as CNN boosting (or RNN boosting) allows various CNNs (or RNNs) to be complementary, this general boosting solution allows CNNs to be complementary with RNNs. In that sense, it approximates the end-to-end training of a “CNN+RNN” network, which is theoretically ideal but not computationally tractable.

The remainder of this paper is organized as follows. Section 2 reviews the state of the art of related fields, namely 1) deep learning for video analysis, 2) spatial and temporal analysis of surgery videos, and 3) CNN and RNN boosting. Section 3 describes “CNN+RNN” networks. Section 4 introduces the proposed boosting algorithm for these networks. Section 5 presents a new dataset of cataract surgery videos and section 6 reports the experiments performed on that dataset. We end with a discussion and conclusions in section 7.

2. State of the Art

2.1. Deep Learning for Video Analysis

The automatic analysis of dynamic scenes through deep learning has become a very hot research topic. The goal is either to recognize human actions [Simonyan and Zisserman, 2014; Wang et al., 2017] or emotions [Fan et al., 2016; Khorrami et al., 2015] to caption all-purpose videos [Yao et al., 2015; Donahue et al., 2017] or to monitor specific scenes [Jiang et al., 2016; Singh et al., 2016]. Medical applications, ranging from gait analysis [Feng et al., 2016] to surgery monitoring [Bodenstedt et al., 2017; Twiampa et al., 2016], also start to emerge. Different solutions have been proposed to analyze image sequences, instead of analyzing images independently inside the video stream. Most of these solutions derive from one of the following three strategies. The first strategy is to regard videos or video portions as 3-D images and therefore analyze them with 3-D CNNs [Li et al., 2013; Zhu et al., 2016]. The main drawback of this first approach is its complexity, which calls for many training samples and implies longer processing times. A second strategy is to analyze 2-D images as well as the optical flow between consecutive images [Simonyan and Zisserman, 2014; Ye et al., 2015]. Using these simpler, preprocessed inputs leads to a more tractable model. However, this approach only models short-term relationships between images and is prone to optical flow estimation failures. Finally, a third strategy is to combine a CNN, analyzing 2-D (or sometimes 3-D) images, with a RNN analyzing the temporal sequencing [Yao et al., 2015; Donahue et al., 2017]. The main advantage of this “CNN+RNN” approach, which is now the leading video analysis solution, is that long-term relationships between events can be taken into account efficiently without any matching process involved.

Video classification, the task of assigning one class label to each video as a whole, is the most popular application of “CNN+RNN” models [Yao et al., 2015; Sun et al., 2016; Fan et al., 2016; Feng et al., 2016; Li et al., 2016; Wang et al., 2017; Chen et al., 2017; Gammulle et al., 2017]. In this purpose, one RNN, referred to as an encoder, takes as inputs one feature vector per frame (generated by one or several CNNs) and outputs one prediction per frame. Frame-level outputs are then fused in order to obtain a single prediction per video. Another popular task is video captioning: the goal is to generate one sentence describing the video [Cho et al., 2015; Peris et al., 2016; Zhang and Tian, 2016]. In this purpose, a second RNN, referred to as a decoder, generates one sequence of words from the outputs of the encoder. Another popular application of “CNN+RNN” models, which is particularly relevant for our study, is video labeling: the goal is to assign one class label to each frame inside a video [Singh et al., 2016; Jiang et al., 2016; Khorrami et al., 2016]. Two solutions have been proposed for this task. A first solution is simply to use as frame label the frame-level outputs of the RNN encoder [Singh et al., 2016; Jiang et al., 2016]. A second solution is to apply a video classification strategy inside a sliding temporal window [Khorrami et al., 2016]. Other applications of “CNN+RNN” models include video retrieval [Ma et al., 2016] and moving object segmentation in videos [Valipour et al., 2017]. Most of these solutions work with a single 2-D CNN analyzing video frames. However, some solutions rely on 3-D CNNs analyzing temporal sliding windows [Yao et al., 2015; Wang et al., 2017]. Others rely on one 2-D CNN for images and another one for optical flow estimation [Sun et al., 2016; Singh et al., 2016; Chen et al., 2017; Zhang and Tian, 2016] use one 3-D CNN for each of these modalities. Finally, Li et al. (2016) analyses two 2-D CNNs, one for images and one for the optical flow, and two 3-D CNNs, one for temporal sliding windows and one for the entire video, thus providing a multiscale video description. Regarding the RNN part, Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] clearly are the most popular cells (Cho et al., 2015; Wang et al., 2017); these cells include a “forget” gate preventing backpropagated errors from vanishing or exploding. Gated Recurrent Units (GRU) [Cho et al., 2014] are also used sometimes [Jiang et al., 2016; Valipour et al., 2017].
the performance of these lower-complexity cells is often comparable with LSTM. When real-time processing is not required, some encoders include two RNNs: one processing information forward and the other processing information backward (Singh et al., 2016; Peris et al., 2016). The performance of these bidirectional RNNs, which take advantage of past and future information, is generally higher.

2.2. Spatial and Temporal Analysis of Surgery Videos

As evidenced by the M2CAI 2016 challenge on tool presence, the state-of-the-art algorithms for tool detection in surgery videos are CNNs. The best solutions of this challenge relied on a transfer learning strategy: well-known CNNs trained to classify still images in the ImageNet dataset were fine-tuned on images extracted from surgery videos (Raju et al., 2016; Sahu et al., 2016; Twinanda et al., 2017; Zia et al., 2016). In particular, Sahu et al. (2016) and Twinanda et al. (2017) fine-tuned AlexNet (Krizhevsky et al., 2012), Raju et al. (2016) fine-tuned GoogleNet (Szegedy et al., 2015a) and VGG-16 (Simonyan and Zisserman, 2015), and Zia et al. (2016) fine-tuned AlexNet, VGG-16 and Inception-v3 (Szegedy et al., 2015b). Note that temporal information was not exploited in these solutions, like in all other state-of-the-art tool detection algorithms (Bouget et al., 2017). One exception is Al Hajj et al. (2017)’s solution, which relies on a CNN processing sequences of consecutive images, using the optical flow to register and combine local features from consecutive images. The goal is to combine slightly different views on a tool, some of which being affected by motion blur or occlusion. However, long-term relationships between images are not exploited neither in Al Hajj et al. (2017).

In the more general context of surgical workflow analysis, where the goal is to recognize the surgical phases, solutions have been proposed to incorporate temporal information at a global level (Lalys and Jannin, 2014; Charrière et al., 2017). Most solutions rely on statistical models, such as Hidden Markov Models (HMMs) (Cadène et al., 2016), Hidden semi-Markov Models (Dergachyova et al., 2016), Hierarchical HMMs (Twinanda et al., 2017), Linear Dynamical Systems (Zappenda et al., 2013) or Conditional Random Fields (Tao et al., 2013; Quellec et al., 2014; Lea et al., 2016a). Recently, solutions based on RNNs have also been proposed (Jin et al., 2016; Bodenstedt et al., 2017; Twinanda et al., 2016). Following the state-of-the-art video analysis strategy, these RNNs process instant visual features extracted from a CNN from images. In particular, Jin et al. (2016) apply a CNN+LSTM network to a small sliding window of three images. Bodenstedt et al. (2017) apply a CNN+GRU network to larger sliding windows and copy the internal state of the network between consecutive window locations. As for Twinanda et al. (2016), they apply a CNN+LSTM network to full videos. Interestingly, the CNN proposed by Twinanda et al. (2016), namely EndoNet, detects tools as an intermediate step. A challenge on surgical workflow analysis was also organized at M2CAI 2016 (two of the top three solutions relied on RNNs and more specifically on LSTM networks (Jin et al., 2016; Twinanda et al., 2016). It should be noted that successful works on the analysis of kinematics surgery data have also been reported, using an LSTM network (Dipietro et al., 2016) or a CNN along the temporal dimension (Lea et al., 2016b). In all these works, statistical models or RNNs were used to label surgical activities and phases. However, given the strong correlation between surgical activities and tool usage, they can be expected to improve tool recognition as well.

2.3. Boosting Convolutional or Recurrent Neural Networks

Boosting is a family of machine learning meta-algorithms for combining multiple weak learners into a single strong one. These meta-algorithms traditionally operate on decision trees or decision stumps (one-level decision trees) as weak learners. A few years after Freund and Schapire (1997) published AdaBoost, the most popular boosting algorithm, Schwenk and Bengio (2000) showed that AdaBoost could advantageously operate on neural networks (multilayer perceptrons) as well. The use of RNNs as weak learners for text or time series analysis was also experimented (Assaad et al., 2008; Buabin, 2012). The first attempts at combining CNNs and AdaBoost for image analysis relied on decision trees (Karianakis et al., 2015) or decision stumps (Song et al., 2015) operating on features extracted from a pre-trained CNN. Recently, a few authors used CNNs as weak learners for AdaBoost (Liu et al., 2015; Gao et al., 2016; Wang et al., 2016; Han et al., 2016; Chen and Liu, 2017). More recent boosting algorithms, such as AnyBoost (Mason et al., 1999) and Friedman (2001)’s Gradient Boosting Machines (GBM), are formulated as a gradient descent optimization, which integrates nicely with the way neural networks are trained. A few authors thus used CNNs as weak learners for AnyBoost (Moghimi et al., 2016) or GBM (Zhang et al., 2016; Walach and Wolf, 2016). Note that existing works on video analysis involving a boosting algorithm operate on independent frames (Han et al., 2016; Walach and Wolf, 2016) or summarize each video with a few feature images as a preprocessing step (Wang et al., 2016): no temporal model is used. In particular, no boosting algorithm has been proposed for “CNN+RNN” models.

CNN or RNN boosting algorithms have been proposed for regression (Assaad et al., 2008; Walach and Wolf, 2016), binary classification (Gao et al., 2016; Chen and Liu, 2017), multilabel classification (Buabin, 2012; Liu et al., 2015) and multiclass classification (Zhang et al., 2016; Moghimi et al., 2016). When CNNs or RNNs are used as weak learners, the boosting meta-algorithm controls the loss function used to train these learners. Typically, training samples with large classification or regression errors are assigned a larger weight in the updated loss function. Different strategies have been investigated to use RNNs or CNNs as weak learners. First, Wang et al. (2016) and Han et al. (2016) build a single CNN: boosting is used to change the loss function at regular intervals during the course of CNN training. Second, Gao et al. (2016) build an ensemble of CNNs where each weak learner is selected among a set of CNNs with slightly different architectures (the filter sizes vary). All other authors
build an ensemble of CNNs or RNNs where all weak learners have the same architecture; of course, the neuron weights are different since they are trained with different loss functions. As proposed by [Moghimi et al. (2016)], each weak learner can be initialized with the state (i.e. the neuron weights) of the previous weak learner, which speeds up training and leads to higher performance.

2.4. Proposed Solution

In this paper, we propose to design “CNN+RNN” networks, the state-of-the-art video analysis framework, for the task of automatic tool usage annotation. Due to the specific challenges of this task, namely the similarity between some tools and the rarity of some tool usages, we propose to apply the gradient boosting principle to both the CNN part and the RNN part of the network, in a novel and unified manner. Besides addressing the previously mentioned difficult cases, the proposed framework has multiple advantages: 1) it can be used to select the optimal network architectures, an open problem in deep learning, and 2) it can improve the complementarity of CNNs and RNNs, an unsolved problem in “CNN+RNN” models for which end-to-end learning is not tractable (see Fig. [1]). Section 3 briefly describes the networks considered in this paper and the related challenges. Section 3.2 describes the boosting algorithm proposed to address those challenges. In particular, the proposed boosting solution has several novelties:

- The proposed RNN boosting algorithm is novel: in particular, RNN boosting has never been formulated as a gradient descent before. Its use in the medical domain is also novel.

- The proposed algorithm for boosting CNNs, in the absence of RNNs, is related to the one proposed by [Moghimi et al. (2016)]; it mainly differs in that it is designed for multilabel classification rather than multiclass classification. However, to the best of our knowledge, the use of CNN boosting for medical images or videos is novel.

- For computational complexity reasons, CNNs are always trained independently of RNNs in the literature. The main contribution of this paper lies in the design of an optimal CNN or CNN ensemble to be used as input for an RNN or RNN ensemble (through boosting — see section 3.5), which has never been studied before.

3. “CNN+RNN” Networks

3.1. Notations

Let $\Theta$ denote a set of surgical tools that should be detected in videos. Let $D$ denote a collection of training videos. Let $V_t$ denote the $t$-th frame in video $V \in D$. Let $\delta(V_t, \theta) \in \{-1, 1\}$ denote the binary label assigned to frame $V_t$ for tool $\theta \in \Theta$: this label indicates whether or not tool $\theta$ is being used in frame $V_t$.

3.2. CNN and RNN Blocks

Neural networks considered in this paper consist of one or several CNNs working in parallel: this set of CNNs is referred to as the “CNN block”. Let $p(V_t) = \{p(V_t, \theta) \in [0; 1], \theta \in \Theta\}$ denote the instant predictions computed by the CNN block for frame $V_t$. Some of the neural networks considered in this paper also contain one or several RNNs working in parallel: this set of RNNs is referred to as the “RNN block”. Let $q(V_t) = \{q(V_t, \theta) \in [0; 1], \theta \in \Theta\}$ denote the context-aware predictions computed by the RNN block for frame $V_t$. Neural networks containing solely a CNN block are referred to as “CNN alone” networks: they take all pixel values from one frame and return $\Theta$ predictions for that frame. Neural networks containing a CNN block and an RNN block are referred to as “CNN+RNN” networks: they take all pixel values from each frame and return $\Theta$ predictions for each frame (see Fig. [1] (a)). In these networks, the feature vectors taken by the RNN block are the predictions $p(V_t)$ returned by the CNN block.

Any CNN can be used in the proposed framework. Readers are referred to Krizhevsky et al. (2012), Simonyan and Zisserman (2015), Szegedy et al. (2015a) and Szegedy et al. (2015b) for details on the topic. Details on the lesser-known RNNs and how they integrate with CNNs are given in the following section.

3.3. RNNs

The main building blocks of RNNs are “cells”, the most popular being LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014). These cells $C_{it}$ are organized in one or several layers $i = 1..n$ and indexed by a timestamp $t$. Each cell takes one input vector $x_{i,t-1}$ from the lower level ($x_{0,t} = p(V_t)$) and one state vector $c_{i,t-1}$ from the previous cell in the same layer; the “previous” cell may also be indexed by timestamp $t + 1$ if we want information to flow backward in time. Then, the cell passes one output vector $x_{i,t}$ to the next layer and one state vector $c_{i,t}$ to the next cell in the same layer. Depending on the type of cells, other vectors (e.g. the forget gate in LSTM) are passed between cells in the same layer. All cells in the same layer share weights, so two representations are possible for RNNs: “rolled” and “unrolled” (see Fig. [1]). In terms of temporal context, two types of RNNs are considered:

1. In “unidirectional” networks, information between RNN cells flows from timestamp $t$ to timestamp $t + 1$: one feature vector $y_{n,t}$ is produced for each frame $V_t$. If only one RNN is present in the RNN block, $y_{n,t}$ is the $q(V_t)$ vector.

2. In “bidirectional” networks (Schuster and Paliwal, 1997), illustrated in Fig. [2] two independent RNNs are defined: in one of them, information flows from timestamp $t$ to timestamp $t + 1$; in the other one, information flows from timestamp $t$ to timestamp $t - 1$. Two features vectors $y_{n,t}^{(a)}$ and $y_{n,t}^{(b)}$ are thus produced for each frame $V_t$: these two feature vectors are concatenated to form $y_{n,t}$ (i.e. $q(V_t)$ if there is a single RNN in the RNN block).

In both variations, a time-distributed dense layer with $\Theta$ neurons operates on $y_{n,t}$ to produce prediction vectors. By design,
In the literature, RNNs are generally trained using video sequences consisting of a few dozen frames (Chen et al., 2017; Gammulle et al., 2017). In contrast, analyzing all frames of full surgery videos requires the analysis of much longer sequences: for instance, there are at least 10,000 frames per video sequence in our cataract surgery videos (see section 5). Training long-term relationships with RNNs is more computationally intensive using long sequences, so we propose to analyze shorter sequences. In that purpose, \( M \) subsampled versions of each original sequence \( V \), denoted by \( V^{(m)} \), \( m = 1..M \), are generated as follows:

\[
V^{(m)} = \{ V_u | u = m + tM, t \in \mathbb{N}^*, u \leq |V| \}.
\]

During training, this results in a novel kind of data augmentation (Shen et al., 2017): the number of training sequences increases artificially. For simplicity, \( \{ V^{(m)} | V \in \mathcal{D}, m = 1..M \} \) is denoted by \( \mathcal{D} \) in the remainder of this paper. During testing, each of the \( M \) subsequences of \( V \) are analyzed independently and the final prediction sequence for \( V \) is obtained by interleaving the resulting \( M \) prediction sequences.

### 3.5. Training “CNN+RNN” Network

Because CNNs and RNNs are integrated into the same network, it would make sense to train the entire network from end to end, so that features extracted by the CNNs are as relevant as possible to the RNNs that process them further. However, as illustrated in Fig. 1, the complexity of the learning process is very high:

- The error measured for each prediction \( q(V_u, \theta) \) is backpropagated to \( p(V_t) \) but also to all \( p(V_u) \) (such as \( u \leq t \), in unidirectional networks).
- Errors computed for each \( p(V_u) \) are backpropagated further to \( V_u \).

The vast majority of weights in a “CNN+RNN” network are in the CNNs. Therefore, the cost of backpropagating an error measured for one timestamp \( t \) to all frames \( V_u \) in the video sequence (such as \( u \leq t \), in unidirectional networks) is very high. As a consequence, a two-step training process is always preferred in the literature (see section 2.1):

1. A CNN is trained first: errors measured for one timestamp \( t \) are only backpropagated to \( V_t \).
2. Then, a RNN is trained: errors measured for one-time
tamp $t$ are backpropagated to all $p(V_t)$ (such as $u \leq t$
in unidirectional networks) without affecting the CNN
weights. Given the number of weights in a RNN, this process
is tractable.

We propose a solution based on boosting that is able to improve
the CNN block after or while training the RNN block, in order
to achieve the desirable properties of end-to-end training, but at
a reasonable computational cost.

4. Boosted “CNN+RNN” Networks

A boosting algorithm is proposed in this section to design ei-
ther a CNN block or an RNN block. This algorithm is based
on GBM (Friedman, 2001), an advanced boosting algorithm
formulated as a gradient descent, like most state-of-the-art ma-
chine learning algorithms. The same algorithm is used for CNN
boosting in “CNN alone” networks and for RNN boosting in
“CNN+RNN” networks. To ensure the complementarity of the
CNN and RNN blocks in “CNN+RNN” networks, an improved
criterion is proposed for CNN boosting in such networks (see
section 4.5). How to design an adequate neural network ar-
chitecture for a given classification problem remains an open
question. So, generalizing [Gao et al., 2016], multiple architec-
tures of neural networks (CNNs or RNNs) are considered in this
study; let $\mathcal{H}$ denote the set of (CNN or RNN) architectures.

4.1. Gradient Boosting Machine

The purpose of GBM is to build a strong learner $H_L$ by lin-
early combining multiple weak learners $h_l \in \mathcal{H}$, $l=1, L$, with
weights $\alpha_l$. GBM was initially described for binary classifica-
tion problems: a multilabel extension is presented hereafter. Let
$h_l(x) = \{h_l(x, \theta), \theta \in \Theta\}$ denote the predictions of $h_l$ for some
input $x$. The predictions of the strong learner for $x$ are given by:

$$H_L(x) = \sum_{l=1}^L \alpha_l h_l(x) . \tag{2}$$

These predictions are mapped to probabilities using the sigmoid
function $\sigma$:

$$\sigma(y) = \frac{1}{1 + \exp(-y)} . \tag{3}$$

The $\sigma(h_l(x, \theta))$ probability is noted $p_l(x, \theta)$ in CNN boost-
ing and $q_l(x, \theta)$ in RNN boosting. In GBM, weak learners are
added sequentially in order to minimize a loss function. One
possible loss function for binary or multilabel classification is
the negative log-likelihood (Friedman, 2001):

$$\mathcal{L}(h) = - \sum_{x, \theta \in \Theta} \left[ \sum_{l,x,h(x,\theta)=1} \log \sigma(h(x, \theta)) + \sum_{l,x,h(x,\theta)=0} \log [1 - \sigma(h(x, \theta))] \right] , \tag{4}$$

where $\delta(h(x, \theta))$ is the binary label assigned to $x$ for tool $\theta$ (see
section 3.1). At each boosting iteration $L+1$, all weak learn-
ers $h_l, l=1, L$, are backpropagated to $H_L$. Then, a RNN is trained:
errors measured for one times-

4.2. Sample Weights

The main idea behind boosting is to build and select weak
learners focusing on samples incorrectly classified by previ-
ously selected weak learners. In that purpose, a weight $\omega_L(x)$
is assigned to each sample $x$ at boosting iteration $L+1$: this
weight is used to modify the weak learners’ loss function. In
multilabel classification, a weak learner does not necessarily
work equally well for all labels. Therefore, $\omega_L(x)$ should be a
vector: $\omega(x) = \{\omega_l(x, \theta), \theta \in \Theta\}$. These weights are defined as
follows:

\[
\omega_{L+1}(x, \theta) = \begin{cases} 
\delta(x, \theta) & \text{if } L = 0 \\
-\frac{\partial L(H_L)}{\partial H_L(x, \theta)} & \text{if } L > 0
\end{cases} \quad (6)
\]

Samples weights \(\omega_{L+1}, L > 0\), are given by:

\[
\omega_{L+1}(x, \theta) = \begin{cases} 
1 - \sigma(H_L(x, \theta)) & \text{if } \delta(x, \theta) = 1 \\
-\sigma(H_L(x, \theta)) & \text{if } \delta(x, \theta) = -1
\end{cases} \quad (7)
\]

**Proof for Eq. (7)**. This equation can be obtained using the chain rule of derivation, applied to the derivatives of Eq. (5) and (4). Setting \(y = \sigma(H_L(x, \theta))\), the following derivatives are used:

\[
\frac{\partial \sigma(y)}{\partial y} = \sigma(y)(1 - \sigma(y)), \quad (8)
\]

\[
\frac{\partial \log \sigma(y)}{\partial \sigma(y)} = \frac{\partial \log(1 - \sigma(y))}{\partial \sigma(y)} = \frac{-1}{1 - \sigma(y)}. \quad (9)
\]

### 4.3. Loss Functions for Boosting Neural Networks

The loss function for training the strong classifier has been presented in section [4.3] [see Eq. (4)]. We now discuss the choice of adequate loss functions for training neural networks individually, as weak learners. As noted by Friedman [2001], the weak learner selected at boosting iteration \(L + 1 \geq 1\) should ideally return a number \(H_{L+1}(x, \theta)\) proportional to \(\omega_{L+1}(x, \theta) = -\frac{\partial L(H_L)}{\partial H_L(x, \theta)}\).

\[H_{L+1}(x, \theta) = \kappa \omega_{L+1}(x, \theta), \quad \forall x, \forall \theta, \kappa \in \mathbb{R}. \quad (10)\]

With that property, the strong learner’s loss function would decrease directly towards zero. With standard weak classifiers, such as decision stumps, it is generally not possible to achieve such a behavior: instead, weak classifiers are generally trained to minimize classifications losses, such as the negative log likelihood, after weighting each error by the corresponding \(\omega_L\) value. However, neural networks can be trained indiscriminately as classifiers or regressors: in particular, they can be trained to solve Eq. (10) in the least square sense, using \(\kappa = 1\) without loss of generality. Therefore, the following quadratic loss function can be used:

\[\mathcal{L}_2(h, \omega) = \sum_\theta \sum_x (h(x, \theta) - \omega(x, \theta))^2. \quad (11)\]

With a different reasoning, Moghimi et al. [2016] proposed the same optimization criterion for their boosted multiclass classifier based on AnyBoost. In fact, they use this criterion for all boosting iterations, including the first. We argue that this criterion should only be used for iterations \(L + 1 > 1\): the first iteration solves a standard classification task and the quadratic loss function is known to be suboptimal for this task. The proposed solution for choosing the weak learners’ loss function is summarized below:

1. at iteration \(L + 1, L > 0\), each weak learner \(h\) is defined as a regressor, trained to minimize \(\mathcal{L}_2(h, \omega_{L+1})\), the quadratic loss function [see Eq. (11)];

Intuitively, the boosting strategy can be interpreted as follows.

If a sample \(x\) was correctly classified by the previous strong classifier \(H_1\) for label \(\theta\), then we don’t want the classifier’s response to change for the \((x, \theta)\) pair in the \(H_{L+1}\) classifier. This behavior is achieved using the loss function of Eq. (11), provided that \(\omega_{L+1}(x, \theta) = 0\), which is guaranteed when the sample is correctly classified by \(H_L\) [see Eq. (7)]. Similarly, if \(H_1\)’s response for the \((x, \theta)\) pair was too low (respectively too high), then the loss function tends to increase (respectively decrease) \(H_{L+1}\)’s response for that pair. Note, however, that the selected weak classifier \(h_{L+1}\) will have a poor classification performance if used alone: obviously positive samples and obviously negative samples will have the same response (0). Clearly, this behavior, which is theoretically optimal, cannot be achieved by the standard strategy used in computer vision competitions (Russakovsky et al., 2015), namely training several architectures independently and combining their outputs afterward.

### 4.4. Efficiently Training Neural Networks as Weak Learners

Training all weak learners, defined as CNNs or RNNs, from scratch at each boosting iteration would be excessively time consuming. An optimized solution is proposed hereafter:

1. The sample weights \(\omega_0\) are initialized according to Eq. (6).
2. For each architecture \(h \in \mathcal{H}\), neuron weights are initialized at random.
3. For each boosting iteration \(L \geq 1\),
   (a) each architecture \(h \in \mathcal{H}\) is trained using a given loss function [see section 4.3] until convergence on a validation set,
   (b) the best architecture \(h \in \mathcal{H}\) is added to the strong learner [see Eq. (5)],
   (c) the sample weights \(\omega_{L+1}\) are updated [see Eq. (6)] and so is the loss function [see section 4.3].

It should be noted that the neuron weights are only initialized once, before the first boosting iteration: following Moghimi et al. [2016], they are not reinitialized at the beginning of each boosting iteration. This strategy saves time and also improves performance. Indeed, more and more samples receive marginal weights at each boosting iteration. Therefore, the training set somehow becomes smaller and smaller. Starting the optimization from weights obtained during previous boosting iterations can be regarded as transfer learning from a larger dataset, which is known to be beneficial.

### 4.5. Boosting CNNs inside a “CNN+RNN” Network

The boosting solution described in previous sections applies to CNN boosting in a “CNN alone” network and also to RNN boosting in a “CNN+RNN” network. However, it is clearly suboptimal for CNN boosting in a “CNN+RNN” network. Let us assume that one image in a video sequence is wrongly classified by the first selected CNN \(h_1\), due to instant motion blur.
or occlusion for instance, but is correctly classified in the previous and following frame. Based on the temporal context, the RNN block would likely correct this instant classification error. Therefore, building a second CNN \(h_2\) for correcting instant blurry or occlusion problems specifically is not very useful. Instead, CNNs should be trained to maximize the performance of the “CNN+RNN” network as a whole.

Throughout the rest of this paper, let \(H', h', \alpha'\) and \(L'\) denote respectively the strong learner, the weak learners, their weights and their number in the RNN block, in order to avoid confusion with their counterparts in the CNN block. To achieve the desired behavior, the sample weights \(\omega_{t+1}\) should be defined based on \(q_L\), the outputs of the RNN block, rather than \(p_l\), the outputs of the CNN block: the goal should be to minimize \(\mathcal{L}(H_t, h', L')\). In this scenario, \(\omega_{t+1}(V_t, \theta)\), the weight assigned to frame \(V_t\) and label \(\theta \in \Theta\), does not depend solely on instant quantities, namely \(H_t(V_t)\) and \(\delta(V_t, \theta)\). In bidirectional networks (for offline processing), it depends on all pairs such that \(u \geq t\). For \(L = 0\), the definition of sample weights is unchanged: \(\omega_{t+1}(V_t, \theta) = \delta(V_t, \theta)\). For \(L > 0\), these weights become:

\[
\omega_{t+1}(V_t, \theta) = \frac{p_L(V_t, \theta)(1 - p_L(V_t, \theta))}{\sum_{\phi \in \Theta} \sum_{V_u} \Delta \delta(V_u, \phi)(V_t, \theta, V_u, \phi)}
\]

\[
\Delta^+(V_t, \theta, V_u, \phi) = (1 - q_L(V_u, \phi)) \sum_{l=1}^L \alpha'_l \frac{\partial h'_l(V_u, \phi)}{\partial p_L(V_t, \theta)}
\]

\[
\Delta^-(V_t, \theta, V_u, \phi) = -q_L(V_u, \phi) \sum_{l=1}^L \alpha'_l \frac{\partial h'_l(V_u, \phi)}{\partial p_L(V_t, \theta)}
\]

(12)

If a unidirectional RNN network is used, then the \(\partial h'_l(V_u, \phi)/\partial p_L(V_t, \theta)\) partial derivatives equal zero for all \(u < t\). In all other cases, they can be computed automatically by the backpropagation algorithm. Note that the backpropagation algorithm does not compute each \(\partial \sigma/\partial \phi\) term individually, where \(I\) denotes an input tensor whose influence on the output tensor \(O\) should be computed. Instead, it computes:

\[
\sum_i \frac{\partial O_i}{\partial I_j} \nabla_i,
\]

(13)

given a tensor \(\nabla\) weighting each coefficient of the output tensor. However, Eq. (12) can be computed setting:

- \(O_i = h_i'(V_u, \phi), i = (u, \phi),\)
- \(I_j = p_L(V_t, \theta), j = (t, \theta),\)
- \(\nabla_i = 1 - q_L(V_u, \phi)\) or \(\nabla_i = q_L(V_u, \phi)\) depending on \(\Delta \delta(V_u, \phi).\)

Proof for Eq. (12). In this scenario, the partial derivative of the negative log-likelihood function [see Eq. (4)], with respect to \(H_L(V_t, \theta)\), is given by:

\[
\frac{\partial \mathcal{L}(H_t, h', L')}{\partial H_L(V_t, \theta)} = -\sum_{\phi \in \Theta} \sum_{V_u, \delta(V_u, \phi) = 1} \frac{\partial \log q_L(V_u, \phi)}{\partial H_L(V_t, \theta)} + \sum_{V_u, \delta(V_u, \phi) = -1} \frac{\partial \log (1 - q_L(V_u, \phi))}{\partial H_L(V_t, \theta)}.
\]

(14)

Each term in this sum can be decomposed according to the chain rule of derivation, using Eq. (9) and Eq. (15) below:

\[
\frac{\partial q_L(V_u, \phi)}{\partial H_L(V_t, \theta)} = \frac{\partial q_L(V_u, \phi)}{\partial \sigma(H_L(V_t, \theta))} \frac{\partial \sigma(H_L(V_t, \theta))}{\partial H_L(V_t, \theta)}.
\]

(15)

The first factor on the right hand side of Eq. (15) can be decomposed as follows using Eq. (9):

\[
\frac{\partial q_L(V_u, \phi)}{\partial H_L(V_t, \theta)} = q_L(V_u, \phi)(1 - q_L(V_u, \phi)) \sum_{l=1}^L \alpha'_l \frac{\partial h'_l(V_u, \phi)}{\partial p_L(V_t, \theta)}.
\]

(16)

The second factor on the right hand side of Eq. (15) can also be decomposed using Eq. (8).

The sample weights we have defined for CNN boosting inside a “CNN+RNN” network are more complex than the general case [see Eq. (3)]. However, they are only computed once per boosting iteration. Therefore, they do not make the optimization problem significantly less tractable, as opposed to the end-to-end training of a “CNN+RNN” network. But, like end-to-end training, they ensure a good complementarity between the CNN and RNN blocks.

4.6. Joint CNN and RNN Boosting

We have seen how to design the CNN and RNN blocks using gradient boosting. What remains to be seen is the order in which these blocks should be designed, in order to design optimal “CNN+RNN” architectures. Two strategies are proposed below.

4.6.1. “Sequential” strategy

The most straightforward solution is to boost the CNN block while \(\mathcal{L}(H_L)\) decreases, and then to boost the RNN block while \(\mathcal{L}(H_L')\) decreases. Besides the use of boosting, this is the standard approach for designing “CNN+RNN” networks (see section 2.1). However, this solution suffers from the limitation described in the previous section, namely the lack of complementarity between the CNN and RNN blocks.

4.6.2. “Joint” strategy

To overcome this limitation, we propose to design the CNN and RNN blocks inside a single boosting loop, using a single strong learner’s loss function, namely \(\mathcal{L}(H_L, h', H_L')\). At each boosting iteration, all CNN architectures \(h \in \mathcal{H}\) and all RNN architectures \(h' \in \mathcal{H}'\) are trained (or re-trained) and only one CNN or one RNN is added to the network: the one minimizing

\[
\mathcal{L}(H_L + ah, H_L') \cup \mathcal{L}(H_L', h' + a'h') \mid h \in \mathcal{H}, a \geq 0 \cup \mathcal{L}(H_L, h' + a'h') \mid h' \in \mathcal{H}', a' \geq 0
\]

(17)
Of course, in the first boosting iteration, only CNN architectures are considered: RNNs need at least one feature extractor to operate. Eq. (12) is used to define the sample weights for CNN boosting as soon as $L' \geq 1$. Boosting stops when the joint loss function stops decreasing.

5. CATARACTS Dataset

The purpose of cataract surgeries is to remove a clouded natural lens and replace it with an artificial lens. The entire procedure can be performed with small incisions only. The natural lens is indeed broken into small pieces with the help of a high-frequency ultrasound device before leaving the eye. As for the artificial lens, it is rolled up inside an injector before entering the eye. This procedure typically lasts 15 minutes. We recently released a dataset, called CATARACTS

5.1. Video Collection

The dataset consists of 50 videos of cataract surgeries performed in Brest University Hospital between January 22, 2015 and September 10, 2015. Reasons for surgery included age-related cataract, traumatic cataract and refractive errors. Patients were 61 years old on average (minimum: 23, maximum: 83, standard deviation: 10). There were 38 females and 12 males. Informed consent was obtained from all patients. Surgeries were performed by three surgeons: a renowned expert (48 surgeries), a one-year experienced surgeon (1 surgery) and an intern (1 surgery). Surgeries were performed under an OPMI Lumera T microscope (Carl Zeiss Meditec, Jena, Germany). Videos were recorded with a 180I camera (Toshiba, Tokyo, Japan) and a MediCap USB200 recorder (MediCapture, Plymouth Meeting, USA). The frame definition was 1920x1080 pixels and the frame rate was approximately 30 frames per second. Videos had a duration of 10 minutes and 56 s on average (minimum: 6 minutes 23 s, maximum: 40 minutes 34 s, standard deviation: 6 minutes 5 s). In total, more than nine hours of surgery have been video recorded.

5.2. Tool Usage Annotation

All surgical tools visible in microscope videos were first enumerated and labeled by the surgeons: a list of 21 tools was obtained (see Fig. 4). Then, the usage of each tool in videos was annotated independently by two non-clinical experts. A tool was considered to be in use whenever it was in contact with the eyeball. Therefore, a timestamp was recorded by both experts whenever one tool came into contact with the eyeball, and also when it stopped touching the eyeball. Up to three tools may be used simultaneously: two by the surgeon (one per hand) and sometimes one by an assistant. Annotations were performed at the frame level, using a web interface connected to an SQL database. Finally, annotations from both experts were adjudicated: whenever expert 1 annotated that tool A was being used, while expert 2 annotated that tool B was being used instead of A, experts watched the video together and jointly determined the actual tool usage. However, the precise timing of tool/eyeball contacts was not adjudicated. Therefore, a probabilistic reference standard was obtained:

- 0: both experts agree that the tool is not being used,
- 1: both experts agree that the tool is being used,
- 0.5: experts disagree.

Figure 4: Surgical tools annotated in videos

---

4https://cataracts.grand-challenge.org
To be assigned to the training and validation subsets: two videos were assigned to the validation subset, the other 23 were assigned to the training subset. The validation videos were chosen such that all tools appear in the training subset: it was not possible to ensure this property for both subsets.

### 6. Experiments

#### 6.1. Architectures

Regarding CNN boosting, two experiments were performed:

1. The first experiment studied the influence of image size. Three weak classifiers based on $O_O$’s ‘net B’ network [Quellec et al., 2017] were defined. The original $O_O$’s ‘net B, or $O_O$ for short, was applied to images of $480 \times 270$ pixels. $O_O$ was slightly modified as illustrated on Fig. 5 in order to process other image sizes as well: $320 \times 180$ pixels and $720 \times 405$ pixels.

2. The second experiment studied the influence of network architecture. Each weak classifier was defined as a CNN architecture from the literature: $O_O$, but also Inception-v4 and Inception-ResNet-v2 (Szegedy et al., 2017). These architectures were used without modifications and were therefore applied to image sizes close to the original ones: $480 \times 270$ pixels for $O_O$, $342 \times 192$ pixels for Inception-v4 and Inception-ResNet-v2.

All CNNs were implemented using TensorFlow.

Regarding RNN boosting, two types of RNN cells were used: LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Cho et al., 2014]. To limit complexity and computation times, the number of layers in RNNs was set to $n = 2$. Three different values were used for $C$, the number of neurons per cell, in order to define six weak classifiers (three based on LSTM, three based on GRU): $C = 64$, $C = 128$, $C = 256$. A subsampling factor of $M = 30$ was used in all RNN boosting experiments: this number was found to be optimal in initial experiments on the validation subset (see Fig. 6). RNNs were implemented using Keras.

Inference times for CNNs, the most computationally intensive parts of the system, are reported in Table 2.

#### 6.2. Experimental Results

Table 2: Inference times of CNNs using one GeForce GTX 1070 GPU by Nvidia. Inference times are given for batch processing (mini-batches of 32 images), which can be used for offline video labeling, and for single image processing, which must be used for online video labeling.

| Tool | CNN | single image | batch processing |
|------|-----|--------------|-----------------|
| $O_O$ (small) | 7.4 ms/image | 2.1 ms/image |
| $O_O$ (medium) | 8.3 ms/image | 4.0 ms/image |
| $O_O$ (large) | 10.8 ms/image | 7.3 ms/image |
| Inception-v4 | 12.7 ms/image | 7.6 ms/image |
| Inception-ResNet-v2 | 17.4 ms/image | 8.1 ms/image |

Table 1: Statistics about tool usage annotation in the CATARACTS dataset. The first two columns indicate inter-rater agreement (Cohen’s kappa) before and after adjudication; the largest changes are in bold. The last column indicates the prevalence of each tool in the training subset, ignoring the frames where experts disagree about the usage of that tool, even after adjudication.

Inter-rater agreement, before and after adjudication, is reported in Table 1. Binary labels are needed to supervise classifiers and assess their performance. During training, some tool $\theta \in \Theta$ is considered to be in use if the probabilistic reference for $\theta$ is greater than or equal to 0.5. The classification performance for $\theta$ is assessed only in frames where experts agree about the usage of $\theta$.

#### 5.3. Training and Test Sets

The dataset was divided into a training set (25 videos) and a test set (25 videos). Division was made in such a way that 1) each tool appears in the same number of videos from both subsets (plus or minus one) and 2) the test set only contains videos from surgeries performed by the renowned expert. Apart from that, division was made at random. In total, the training set contains 4 hours and 42 minutes of video and the test set contains 4 hours and 24 minutes of video.

#### 5.4. Training and Validation Subsets

For training CNNs, 1/30th of the training set was used for validation: one image per second of video was assigned to the validation subset, the other 29 were assigned to the training subset. This ensures that all surgical tools, including the rarest, are present in both subsets. For training RNNs, full videos need to be assigned to the training and validation subsets: two videos were assigned to the validation subset, the other 23 were assigned to the training subset. The validation videos were chosen such that all tools appear in the training subset: it was not possible to ensure this property for both subsets.
6.2. Performance of Boosted Video Labelers

The architectures of video labelers obtained by CNN or “CNN+RNN” boosting in the “three image sizes” and “three CNN architectures” experiments are reported in Table 3. Their performance is reported in Table 4. Boosting experiments reported in these tables involve bidirectional RNNs only. ROC curves for the best setup, namely joint “CNN+RNN” Boosting using three CNN architectures, are reported in Fig. 6, and one sequence labeling example obtained with this setup is illustrated in Fig. 5.

Results for unidirectional and bidirectional “CNN+RNN” boosting using three CNN architectures from the literature (o_O, Inception-v4 and Inception-ResNet-v2) are compared to baseline methods in Table 5. Two baseline methods were defined. The first one (“three CNN architectures + RF”) relies on random forests (RFs) combining the outputs of o_O (medium size), Inception-v4 and Inception-ResNet-v2 (trained as described above). One RF of 50 decision trees is built for each tool. All 21 outputs of each CNN is fed to each RF, so each RF has 63 inputs and 1 output. For each tool, the maximum depth \( d_{\text{max}} \) of each decision tree is learned by cross-validation in the training set (\( d_{\text{max}} = 10 \), typically). The second baseline method (“Inception-v4 + LSTM”) relies on a bidirectional LSTM network processing the outputs of Inception-v4 (trained as described above). It is similar to the solution by Twinanda et al. (2017) for tool detection, but with a more recent CNN (Inception-v4 instead of AlexNet). The parameters of the LSTM (number of layers \( n \) and number of neurons per cell \( C \) ) are learned in the validation set. Setting \( M = 1 \), following the state of the art, an average score of \( \bar{A}_C = 0.9520 \) was obtained (using \( n = 1 \) and \( C = 128 \)) after 29 epochs. Setting \( M = 30 \), as proposed in this paper, an average score of \( \bar{A}_C = 0.9622 \) was obtained after only 2 epochs (using \( n = 2 \) and \( C = 64 \)). Ideally, we would also compare the proposed solution with the end-to-end training of a “CNN+RNN” network, but the complexity of that model prevents any experimentation.

6.3. Analysis of the Boosted Video Labelers

To visualize what the CNNs have learned, one can rely on sensitivity analysis (Simonyan et al., 2014) and related metrics. Sensitivity is the gradient of the CNN predictions with respect to the pixel values: the pixel values influencing most the CNN predictions are highlighted. Fig. 7 reports hue-constrained sensitivity analysis maps (Quellec et al., 2017) for the o_O network: the interpretation is similar, except that the three color components of a pixel are analyzed jointly rather than independently. This figure shows that the CNN does not consider solely the tools, but also the anterior segment of the eye: the lens, which is modified by tools, the cornea, which is temporarily deformed by tools as they move, and the corneoscleral junction, where tools are inserted. One explanation is that each tool interacts differently with the eye and, therefore, analyzing the eye structures helps differentiating tools. Another explanation is that the target labels are not related to tool presence, but rather
to tool usage. So CNNs must be able to recognize whenever each tool is in contact with the eye. In particular, the fourth example shows that the forceps at the bottom left corner, away from the cornea, are not detected.

Because our joint “CNN+RNN” boosting algorithm relies on the gradients of RNN predictions with respect to CNN predictions [see Eq. (12)], sensitivity analysis is also useful for RNNs in our case. These gradients are illustrated in a condensed form in Fig. [10] given an RNN $h'$, this figure shows $\nabla_{\phi} h'(h')$, where:

$$\nabla_{\phi} h'(h') = \sum_{V \in D} \sum_{t} \sum_{u} \frac{\partial h'(V_u,\phi)}{\partial p_t(V_t,\theta)} .$$

For a lazy RNN, all coefficients outside the diagonal would be zero. Here, we observe that the diagonal is not even dominant. This is particularly true for tools that are not very well detected by CNNs, such as Charleux cannulae or needle holders (see Table 3). The last column indicates the Pearson correlation between $A_c$ in the test set and tool prevalence in the training set (see Table 4). The best results are in bold.

### Table 3: Composition of the boosted sequence labelers relying on bidirectional RNNs. The order in which weak learners are added is determined automatically by the boosting algorithm. M, S and L stand for small ($320 \times 180$ pixels), medium ($480 \times 270$ pixels) and large ($720 \times 405$ pixels). I-v4 and I-RN-V2 are short for Inception-v4 and Inception-ResNet-v2, respectively. The number of neurons in RNN cells is indicated in brackets.

| Experiment                  | Single CNNs | o-O - three image sizes | Weak classifiers (by order of addition) |
|-----------------------------|-------------|-------------------------|----------------------------------------|
| o-O (small)                 | 0.8543      | 0.9539                  |                                                      |
| o-O (medium)                | 0.9253      | 0.9439                  |                                                      |
| o-O (large)                 | 0.9594      | 0.9594                  |                                                      |
| Inception-v4                | 0.9253      | 0.9439                  |                                                      |
| Inception-ResNet-v2         | 0.9594      | 0.9594                  |                                                      |
| Biomarker                   | 0.9545      | 0.9545                  |                                                      |
| Charleux cannula            | 0.9509      | 0.9509                  |                                                      |
| Hydrodissection cannula     | 0.9509      | 0.9509                  |                                                      |
| Rycroft cannula             | 0.9509      | 0.9509                  |                                                      |
| Viscoelastic cannula        | 0.9509      | 0.9509                  |                                                      |
| Cotton                      | 0.9509      | 0.9509                  |                                                      |
| Capsulorhexis cystotome     | 0.9509      | 0.9509                  |                                                      |
| Bonn forces                 | 0.9509      | 0.9509                  |                                                      |
| Capsulorhexis forceps       | 0.9509      | 0.9509                  |                                                      |
| Troutman forceps            | 0.9509      | 0.9509                  |                                                      |
| Needle holder               | 0.9509      | 0.9509                  |                                                      |
| Irrigation/aspiration HP    | 0.9509      | 0.9509                  |                                                      |
| Phacoemulsifier HP          | 0.9509      | 0.9509                  |                                                      |
| Vitrectomy HP               | 0.9509      | 0.9509                  |                                                      |
| Implant injector            | 0.9509      | 0.9509                  |                                                      |
| Primary incision knife      | 0.9509      | 0.9509                  |                                                      |
| Secondary incision knife    | 0.9509      | 0.9509                  |                                                      |
| Micromanipulator            | 0.9509      | 0.9509                  |                                                      |
| Suture needle               | 0.9509      | 0.9509                  |                                                      |
| Mendez ring                 | 0.9509      | 0.9509                  |                                                      |
| Vannas scissors             | 0.9509      | 0.9509                  |                                                      |
| Average                     | 0.9509      | 0.9509                  |                                                      |
| Corr. with prevalence       | 0.9509      | 0.9509                  |                                                      |

### Table 4: Tool detection performance of the boosted video labelers in terms of average area under the ROC curve ($A_c$) in the test set. All RNNs are bidirectional. The last column indicates the Pearson correlation between $A_c$ in the test set and tool prevalence in the training set (see Table 3). The best results are in bold.
To illustrate what the boosted networks should learn and have actually learnt, Fig. 11 compares the distribution of RNN boosting weights with the distribution of RNN predictions from a weak regressor trained to approximate those weights. We can see that these distributions match closely. The ability of CNNs to approximate boosting weights had been shown by Moghimi et al. (2016), but the transposition to RNNs was not necessarily guaranteed to work. Because boosting weights somehow represent the classification challenge, this figure suggests that RNNs are able to analyze the temporal behavior of labeling challenges. Therefore, RNN boosting is also possible.

7. Discussion and Conclusions

A solution for labeling tool usage in cataract surgery videos has been presented. Following state-of-the-art video analysis solutions, it relies on convolutional neural networks (CNNs) for analyzing each frame in the video and on recurrent neural networks (RNNs) for analyzing the temporal sequencing through-out the entire surgery, based on the outputs of the CNNs. A novel framework for boosting a sequence labeler composed of CNNs and RNNs has been presented. The main motivation for this framework is the fact that “CNN+RNN” labelers cannot be trained from end to end, for complexity reasons. The framework allows to progressively improve the CNN and RNN parts of the system by adding weak classifiers (CNNs or RNNs) designed to improve the overall classification accuracy of the join system. In particular, like the theoretical end-to-end training solution, CNN training is supervised based on the outputs of the RNN block.

The proposed framework has several novelties. The main novelty lies in the boosting algorithm. CNN boosting had been proposed for multiclass classification problems (Moghimi et al. 2016). We adapted it for multilabel classification, showed its applicability to RNN boosting and, more importantly, introduced CNN boosting supervised based on the outputs of the RNN block. A second novelty lies in the proposed temporal sequence augmentation strategy: although very simple, it proved to be quite effective (see Fig. 6).

The proposed framework is quite general and is likely applicable outside the scope of surgery video analysis. However, it is of particular relevance for this application because many

| Tool                        | three CNN architectures + RF | Inception-v4 + LSTM | proposed (unidirectional) | proposed (bidirectional) |
|-----------------------------|-----------------------------|---------------------|---------------------------|-------------------------|
| biomarker                   | 0.9151                      | 0.8741              | 0.9559                    | 0.9278                  |
| Charleux cannula            | 0.8645                      | 0.9490              | 0.9627                    | 0.9594                  |
| hydrodissection cannula     | 0.9662                      | 0.9720              | 0.9848                    | 0.9781                  |
| Rycroft cannula             | 0.9327                      | 0.9951              | 0.9872                    | 0.9954                  |
| viscoelastic cannula        | 0.8696                      | 0.9072              | 0.8766                    | 0.9403                  |
| cotton                      | 0.9616                      | 0.8846              | 0.9726                    | 0.9455                  |
| capsulorhexis cystotome     | 0.9929                      | 0.9909              | 0.9943                    | 0.9947                  |
| Bonn forceps                | 0.8818                      | 0.9573              | 0.9492                    | 0.9658                  |
| capsulorhexis forceps       | 0.9063                      | 0.9720              | 0.9894                    | 0.9794                  |
| Troutman forceps            | 0.8656                      | 0.9569              | 0.9596                    | 0.9671                  |
| needle holder               | 0.8982                      | 0.9831              | 0.9476                    | 0.9756                  |
| irrigation/aspiration HP    | 0.9879                      | 0.9972              | 0.9963                    | 0.9974                  |
| phacoemulsifier HP          | 0.9961                      | 0.9992              | 0.9981                    | 0.9996                  |
| vitrectomy HP               | 0.9347                      | 0.9006              | 0.9463                    | 0.8974                  |
| implant injector            | 0.9624                      | 0.9751              | 0.9507                    | 0.8909                  |
| primary incision knife      | 0.9140                      | 0.9616              | 0.9575                    | 0.9654                  |
| secondary incision knife    | 0.9714                      | 0.9919              | 0.9903                    | 0.9930                  |
| micromanipulator            | 0.9813                      | 0.9933              | 0.9885                    | 0.9949                  |
| suture needle               | 0.9092                      | 0.9752              | 0.9737                    | 0.9829                  |
| Mendez ring                 | 0.9679                      | 0.9847              | 0.9905                    | 0.9837                  |
| Vannas scissors             | 0.9145                      | 0.9847              | 0.9901                    | 0.9815                  |
| Average                     | 0.9330                      | 0.9622              | 0.9696                    | 0.9717                  |
| Corr. with prevalence       | 0.564                       | 0.376               | 0.344                     | 0.397                   |

Table 5: Tool detection performance in terms of average area under the ROC curve — summary of proposed and baseline methods. The proposed methods are based on Joint “CNN+RNN” Boosting using the three CNN architectures and a subsampling factor of $M = 30$. 

Figure 7: Tool detection performance in the test set — ROC curves for the best setup (joint “CNN+RNN” Boosting using three CNN architectures)
tools are very similar to one another (e.g. the cannulae or the forceps — see Fig. 4) but they are often used in a predefined order: using the temporal context (e.g. which tools have been used previously) is quite relevant for differentiating them. Therefore, it seems particularly useful to guide CNN training or boosting based on the temporal context. Experiments on the new CATARACTS dataset for the task of tool usage annotation demonstrated its very good performance: up to $\bar{A}_c = 0.9717$ was achieved on average over a collection of 21 surgical tools. As shown in Table 4, all tools are detected well ($A_c \geq 0.8974$). In particular, detection performance does not depend much on the number of training samples: correlation between $A_c$ and prevalence is usually $< 0.5$.

Besides allowing end-to-end training, boosting is also relevant for feature selection: different architectures are proposed and the most relevant are added sequentially. One way to impact feature extraction is to vary the size of images, another one is to vary the feature extraction algorithm. Therefore, two experiments were performed: one where the same architecture is applied to images of different sizes and one where different architectures are available. Similar conclusions were observed in both experiments. First, CNN boosting alone is disappointing: $\bar{A}_c = 0.9157$ v.s. $\bar{A}_c = 0.9125$ for the best image size in the “three image sizes” experiment; $\bar{A}_c = 0.9509$ v.s. $\bar{A}_c = 0.9483$ for the best architecture in the “three CNN architectures” experiment (see Table 4). One explanation is that much boosting effort is spent on trying to improve tool labeling in frames affected by motion blur, occlusion, or any other challenging conditions, whereas surrounding frames are perfectly labeled: using temporally-filtered outputs for supervising boosting makes more sense. The ability to boost CNNs based on the outputs of RNNs, on the other hand, leads to a noticeable improvement: $\bar{A}_c = 0.9409$ for joint “CNN+RNN” boosting v.s. $\bar{A}_c = 0.9343$ for sequential “CNN+RNN” boosting in the “three image sizes” experiment; $\bar{A}_c = 0.9717$ for joint “CNN+RNN” boosting v.s. $\bar{A}_c = 0.9653$ for sequential “CNN+RNN” boosting in the “three CNN architectures” experiment (see Table 4). These observations support our hypothesis that CNNs should be trained to be complimentary to RNNs.

The proposed framework compares favorably with two state-

---

Figure 8: Sequence labeling for one test video using the best setup (joint “CNN+RNN” Boosting using three CNN architectures): tool usage according to human experts is in black, automatic predictions are in green.
of-the-art computer vision strategies. In particular, we observed that combining multiple CNNs through random forests (RFs) decreases performance: $\bar{A}_c = 0.9330$ v.s. $\bar{A}_c = 0.9483$ for the best CNN. Our explanation is that RFs are trained on CNN predictions computed for training images, which are more ideal than CNN predictions computed for test images. This discrepancy might invalidate the RFs on the test set. However, some tools are only visible in two videos, so the RFs could not be trained on a third independent dataset. One advantage of the proposed boosting strategy is that boosting is governed by tool-independent parameters $\alpha_l$ [see Eq. (5)], which can therefore be tuned on a small validation subsets where the presence of all tools is not required. 

The second baseline, composed of the currently best CNN (Inception-v4) and the state-of-the-art architecture for temporal sequencing (LSTM), was harder to outperform, particularly using the proposed temporal sequence augmentation strategy (see section 3.4): starting from a labeling performance of $\bar{A}_c = 0.9622$ (see Table 5—$\bar{A}_c = 0.9520$ without augmentation) leaves little room for improvement. However, thanks to boosting, we were still able to narrow to gap to the perfect score ($\bar{A}_c = 1$) by one quarter ($\bar{A}_c = 0.9717$).

So far, we have only discussed bidirectional RNNs. As Table 5 shows, unidirectional RNNs are almost as good as bidirectional RNNs for the task under study: $\bar{A}_c = 0.9696$, as opposed to $\bar{A}_c = 0.9717$. It implies that almost equally good performance can be achieved for online video sequencing, using unidirectional RNNs, and offline video sequencing, using bidirectional RNNs. Computation times are of primary importance for online video sequencing. Similarly to the bidirectional version (see Table 3), the unidirectional labeler relies on three weak classifiers: one based on Inception-v4, one based on oO (medium size) and one based on Inception-ResNet-v2. All three together, processing one frame takes 38.4 ms using one GeForce GTX 1070 GPU by Nvidia (see Table 2). Videos of the CATARACTS dataset have a frame rate of 30 image per second (i.e. 33.3 ms per image). It means a slightly faster GPU would be required for real-time video analysis. Alternatively, two GPUs can be used, as the CNN classifiers can be run in parallel.

In the hope to push tool labeling performance even further, we have recently launched a challenge based on the CATARACTS dataset. Whatever comes out of it, in view of the already great performance, automatic monitoring of cataract surgeries can now be envisaged seriously (Charrière et al., 2017). The next step would be to use information collected during the surgery to give feedbacks to the surgeon, either during the surgery or right after it. Cataract surgery being the most common surgical procedure, this is usually the first surgery ophthalmologists have to master. Support to beginners is thus of particular relevance in this context, but many more applications can be envisioned for the near future.

References

Al Hajj, H., Lamard, M., Charrière, K., Cochener, B., Quellec, G., 2017. Surgical tool detection in cataract surgery videos through multi-image fusion inside a convolutional neural network. In: Proc IEEE EMBC. Jeju Island, Korea.

Assaad, M., Boné, R., Cardot, H., Jan. 2008. A new boosting algorithm for improved time-series forecasting with recurrent neural networks. Information Fusion 9 (1), 41–55.

Bodenstedt, S., Wagner, M., Katic, D., Mietkowskii, P., Mayer, B., Kenngott, H., Müller-Stich, B., Dillmann, R., Speidel, S., Feb. 2017. Unsupervised temporal context learning using convolutional neural networks for laparoscopic workflow analysis. Tech. Rep. arXiv:1702.03684 [cs], Karlsruhe Institute of Technology.

6https://cataracts.grand-challenge.org/
Figure 11: Boosting weight regression — illustration for the ‘implant injector’ tool in the “three CNN architectures” experiment based on joint “CNN+RNN” boosting. Fig. (a) shows the distribution of $H^1(x, 'implant injector'). Fig. (b) shows the distribution of $a^1_{\omega}(x, 'implant injector'). Fig. (c) shows the distribution of $a^1_{\omega}(x, 'implant injector').

Bouget, D., Allan, M., Stoyanov, D., Jannin, P., 2017. Vision-based and markerless surgical tool detection and tracking: a review of the literature. Med Image Anal 35, 633–654.

Buabin, E., Oct. 2012. Boosted hybrid recurrent neural classifier for text document classification on the Reuters News text corpus. Int J Mach Learn Comput 2 (5), 588–592.

Cadène, R., Robert, T., Thone, N., Cord, M., Oct. 2016. M2cai workflow challenge: convolutional neural networks with time smoothing and hidden Markov model for video frames classification. Tech. Rep. arXiv:1610.05541 [cs], Université de Pierre et Marie Curie.

Charrière, K., Quellec, G., Lamard, M., Martiano, D., Cazuguel, G., Coatrieux, G., Cochenier, B., 2017. Real-time analysis of cataract surgery videos using statistical models. Multimed Tools Appl. In press.

Chen, H., Chen, J., Hu, R., Chen, C., Wang, Z., Mar. 2017. Action recognition with temporal scale-invariant deep learning framework. China Communications 14 (2), 163–172.

Chen, J.-C., Liu, C.-F., Jun. 2017. Deep net architectures for visual-based clothing image recognition on large database. Soft Comput 21 (11), 2923–2939.

Cho, K., Courville, A., Bengio, Y., Nov. 2015. Describing Multimedia Content Using Attention-Based Encoder-Decoder Networks. IEEE Trans Multimedia 17 (11), 1875–1886.

Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y., Oct. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In: Proc SSST. Doha, Qatar, pp. 103–111, arXiv: 1409.1259.

Dergachyova, O., Bouget, D., Huaülme, A., Morandi, X., Jannin, P., Jun. 2016. Automatic data-driven real-time segmentation and recognition of surgical workflow. Int J Comput Assist Radiol Surg 11 (6), 1081–1089.

Dipietro, R., Lea, C., Malpani, A., Ahmadi, N., Vedula, S., Lee, G., Lee, M., Hager, G., Oct. 2016. Recognizing surgical activities with recurrent neural networks. In: Proc MICCAI. Athens, Greece, pp. 551–558.

Donahue, J., Hendricks, L., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., Darrell, T., Apr. 2017. Long-term recurrent convolutional networks for visual recognition and description. IEEE Trans Pattern Anal Mach Intell 39 (4), 677–691.

Fan, Y., Lu, X., Li, D., Liu, Y., Nov. 2016. Video-based emotion recognition using CNN-RNN and C3d hybrid networks. In: Proc ACM ICMI. Tokyo, Japan, pp. 445–450.

Feng, Y., Li, Y., Luo, J., Dec. 2016. Learning effective gait features using LSTM. In: Proc IEEE ICIAP. Cancun, Mexico, pp. 325–330.

Freund, Y., Schapire, R. E., Aug. 1997. A Decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55 (1), 119–139.

Friedman, J. H., Oct. 2001. Greedy function approximation: A gradient boosting machine. Ann Stat 29 (5), 1189–1232.

Gammulle, H., Denman, S., Sridharan, S., Fookes, C., Mar. 2017. Two stream LSTM: A deep fusion framework for human action recognition. In: Proc IEEE WACV. Santa Rosa, CA, USA, pp. 177–186.

Gao, Y., Rong, W., Shen, Y., Xiong, Z., Jul. 2016. Convolutional neural network based sentiment analysis using Adaboost combination. In: Proc IEEE ICNN. Vancouver, Canada, pp. 1333–1338.

Han, S., Meng, Z., Khan, A., Tong, Y., Dec. 2016. Incremental boosting convolutional neural network for facial action unit recognition. In: Proc NIPS. Vol. 29. Barcelona, Spain, pp. 109–117.

Hochreiter, S., Schmidhuber, J., Nov. 1997. Long short-term memory. Neural Comput 9 (8), 1735–1780.

Li, S., Xu, W., Yang, M., Yu, K., Jan. 2013. 3D convolutional neural networks for human action recognition. IEEE Trans Pattern Mach Intell 35 (1), 221–231.

Jiang, H., Lu, Y., Xue, J., Nov. 2016. Automatic soccer video event detection based on a deep neural network combined CNN and RNN. In: Proc IEEE ICTAI. San Jose, CA, USA, pp. 490–494.

Jin, Y., Dou, Q., Chen, H., Yu, L., Heng, P.-A., Oct. 2016. EndoRCN: recurrent convolutional networks for recognition of surgical workflow in cholecystectomy procedure video. Tech. rep., The Chinese University of Hong Kong.

Karianakis, N., Fuchs, T. J., Soatto, S., Mar. 2015. Boosting convolutional features for robust object proposals. Tech. Rep. arXiv:1503.06350 [cs.CV], University of California Los Angeles.

Khorrami, P., Le, P., Brady, K., Dagli, C., Huang, T., Sep. 2016. How deep neural networks can improve emotion recognition on video data. In: Proc IEEE ICIP. Phoenix, AZ, USA, pp. 619–623.

Krizhevsky, A., Sutskever, I., Hinton, G. E., Dec. 2014. On the properties of deep learning and backpropagation. In: Proc NIPS. Denver, CO, USA, pp. 2849–2857.

Krizhevsky, A., Sutskever, I., Hinton, G. E., Dec. 2012. ImageNet classification with deep convolutional neural networks. In: Proc ICCV. Barcelona, Spain, pp. 1097–1105.

Lalys, F., Jannin, P., May 2014. Surgical process modelling: a review. Int J Comput Assist Radiol Surg 9 (3), 495–511.

Lea, C., Vidal, R., Hager, G. D., May 2016a. Learning convolutional action primitives for fine-grained action recognition. In: Proc IEEE ICRA. Stockholm, Sweden, pp. 1642–1649.

Lea, C., Vidal, R., Reiter, A., Hager, G., Oct. 2016b. Temporal convolutional networks: A unified approach to action segmentation. In: Proc ECCV. Amsterdam, The Netherlands, pp. 47–54.

Li, Q., Qui, Z., Yao, T., Mei, T., Rui, Y., Luo, J., Jun. 2016. Action recognition by learning deep multi-granular spatio-temporal video representation. In: Proc ACM ICMR. New York, NY, USA, pp. 159–166.

Liu, Z., Wang, H., Yan, Y., Guo, G., Sep. 2015. Effective facial expression recognition via the boosted convolutional neural network. In: Proc CCCV. Springer, Berlin, Heidelberg, Xi’an, China, pp. 179–188.

Ma, C., Gu, Y., Liu, W., Yang, J., He, X., Oct. 2016. Unsupervised video hashing by exploiting spatio-temporal feature. In: Proc ICONIP. Kyoto, Japan, pp. 511–518.

Mason, L., Baxter, J., Bartlett, P. L., Frean, M. R., Dec. 1999. Boosting algorithms as gradient descent. In: Proc NIPS. Vol. 12. Denver, CO, USA, pp. 512–518.

Mohgimí, M., Saberián, M., Yang, J., Li, L.-J., Vasconcelos, N., Belongie, S., Sep. 2016. Boosted convolutional neural networks. In: Proc BMVC. York, UK.

Peris, À., Bolaños, M., Ruíz, P., Casacuberta, F., Sep. 2016. Video description using bidirectional recurrent neural networks. In: Proc ICANN.
Quellec, G., Charr`iere, K., Boudi, Y., Cochener, B., Lamard, M., Jul. 2017. Deep image mining for diabetic retinopathy screening. Med Image Anal 39, 178–193.

Quellec, G., Lamard, M., Cochener, B., Cazuguel, G., Dec. 2014. Real-time segmentation and recognition of surgical tasks in cataract surgery videos. IEEE Trans Med Imaging 33 (12), 2352–2360.

Quellec, G., Lamard, M., Cochener, B., Cazuguel, G., Apr. 2015. Real-time task recognition in cataract surgery videos using adaptive spatiotemporal polynomials. IEEE Trans Med Imaging 34 (4), 877–887.

Raju, A., Wang, S., Huang, J., Oct. 2016. M2CAI surgical tool detection challenge report. Tech. rep., University of Texas at Arlington.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., Fei-Fei, L., Apr. 2015. ImageNet large scale visual recognition challenge. Int J Comput Vis 115 (3), 211–252.

Sahu, M., Mukhopadhyay, A., Sengzel, A., Zachow, S., Oct. 2016. Tool and phase recognition using contextual CNN features. Tech. Rep. arXiv:1610.08854 [cs.CV], Zuse Institute Berlin.

Schuster, M., Paliwal, K. K., Nov. 1997. Bidirectional recurrent neural networks. IEEE Trans Signal Process 45 (11), 2673–2681.

Schwenk, H., Bengio, Y., Aug. 2000. Boosting neural networks. Neural Comput 12 (8), 1869–1887.

Shen, D., Wu, G., Suk, H.-I., Jun. 2017. Deep learning in medical image analysis. Annu Rev Biomed Eng 19, 221–248.

Simonoy, K., Vedaldi, A., Zisserman, A., Apr. 2014. Deep inside convolutional networks: visualising image classification models and saliency maps. In: ICLR Workshop. Calgary, Canada.

Simonoy, K., Zisserman, A., Dec. 2014. Two-stream convolutional networks for action recognition in videos. In: Proc NIPS. Vol. 27, Montreal, Canada, pp. 568–576.

Simonoy, K., Zisserman, A., May 2015. Very deep convolutional networks for large-scale image recognition. In: Proc ICLR. San Diego, CA, USA.

Singh, B., Marks, T., Jones, M., Tuzel, O., Shao, M., Jan. 2016. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In: Proc IEEE CVPR. Las Vegas, NV, USA, pp. 1961–1970.

Song, X., Rui, T., Zha, Z., Wang, X., Fang, H., Aug. 2015. The AdaBoost algorithm for vehicle detection based on CNN features. In: Proc ACM ICIMCS. Zhangjiajie, China, pp. 180–184.

Sun, Y., Wu, Z., Wang, X., Arai, H., Kinebuchi, T., Jiang, Y.-G., Oct. 2016. Exploiting objects with LSTMs for video categorization. In: Proc ACM MM. Amsterdam, The Netherlands, pp. 142–146.

Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A., Feb. 2017. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: Proc AAAI. San Francisco, CA, USA, pp. 4278–4284.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., Jun. 2015a. Going deeper with convolutions. In: Proc IEEE CVPR. Boston, MA, USA, pp. 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., Dec. 2015b. Rethinking the Inception architecture for computer vision. Tech. Rep. arXiv:1512.00567 [cs]. Google.

Tao, L., Zappella, L., Hager, G. D., Vidal, R., Sep. 2013. Surgical gesture segmentation and recognition. In: Proc MICCAI. Nagoya, Japan, pp. 339–346.

Trikha, S., Turnbull, A. M. J., Morris, R. J., Anderson, D. F., Hossain, P., Apr. 2013. The journey to femtosecond laser-assisted cataract surgery: new beginnings or a false dawn? Eye (Lond) 27 (4), 461–473.

Twinanda, A. P., Mutter, D., Marescaux, J., de Mathelin, M., Padov, N., Oct. 2016. Single- and multi-task architectures for surgical workflow challenge at M2caI 2016. Tech. Rep. arXiv:1610.08844 [cs]. University of Strasbourg.

Twinanda, A. P., Shehata, S., Mutter, D., Marescaux, J., de Mathelin, M., Padov, N., Jan. 2017. EndoNet: A deep architecture for recognition tasks on laparoscopic videos. IEEE Trans Med Imaging 36 (1), 86–97.

Valipour, S., Siam, M., Jagersand, M., Ray, N., Mar. 2017. Recurrent fully convolutional networks for video segmentation. In: Proc IEEE WACV. Santa Rosa, CA, USA, pp. 29–36.

Walach, E., Wolf, L., Oct. 2016. Learning to count with CNN boosting. In: Proc ECCV. Vol. 9906. Amsterdam, The Netherlands, pp. 660–676.

Wang, L., Zhang, B., Han, J., Shen, L., Qian, C.-S., Sep. 2016. Robust object representation by boosting-like deep learning architecture. Signal Process Image Commun 47, 490–499.

Wang, X., Gao, L., Song, J., Shen, H., 2017. Beyond Frame-level CNN: Saliency-aware 3d CNN with LSTM for Video Action Recognition. IEEE Signal Processing Letters. In press.

Yao, L., Torabi, A., Cho, K., Ballas, N., Pal, C., Larochelle, H., Courville, A., Dec. 2015. Describing videos by exploiting temporal structure. In: Proc IEEE ICCV. Santiago, Chile, pp. 4507–4515.

Ye, H., Wu, Z., Zhao, R.-W., Wang, X., Jiang, Y.-G., Xue, X., Jun. 2015. Evaluating two-stream CNN for video classification. In: Proc ICMR. Shanghai, China, pp. 435–442.

Zappella, L., Béjar, B., Hager, G., Vidal, R., Oct. 2013. Surgical gesture classification from video and kinematic data. Med Image Anal 17 (7), 732–745.

Zhang, C., Tian, Y., Dec. 2016. Automatic video description generation via LSTM with joint two-stream encoding. In: Proc IEEE ICCPR. Cancun, Mexico, pp. 2924–2929.

Zhang, F., Du, B., Zhang, L., Mar. 2016. Scene classification via a gradient boosting random convolutional network framework. IEEE Trans Geosci Remote Sens 54 (3), 1793–1802.

Zhu, W., Hu, J., Sun, G., Cao, X., Qiao, Y., Jun. 2016. A key volume mining deep framework for action recognition. In: Proc IEEE CVPR. Las Vegas, NV, USA, pp. 1991–1999.

Zia, A., Castro, D., Essa, I., Oct. 2016. Fine-tuning deep architectures for surgical tool detection. Tech. rep., Georgia Institute of Technology.