TOWARDS AUTOMATED UNDERSTANDING OF LAPAROSCOPIC VIDEOS

by

BABAK NAMAZI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy at
The University of Texas at Arlington
August, 2019

Arlington, Texas

Supervising Committee:

Venkat Devarajan, Supervising Professor

Ganesh Sankaranarayanan

Michael Manry

Ramtin Madani

Ioannis D. Schizas

Amir Farbin

To my family.

ACKNOWLEDGEMENTS

I have received a great deal of support and assistance throughout writing this dissertation. To begin with, I would like to thank my supervisor Dr. Venkat Devarajan, for his mentor-ship, which was not limited to academic guidance. I consider myself very fortunate for having an advisor who was always willing to share his invaluable expertise, which has been and will be extremely useful for my professional and personal developments.

I would like to appreciate Dr. Ganesh Sankaranayanan for his constant guidance and advice throughout my research. I would also like to thank Baylor University Medical Center at Dallas, and Dr. James Fleshman for providing the funding for this dissertation.

I would like to acknowledge the suggestions and feedback I received from my doctoral committee members Dr. Amir Farbin, Dr. Michael Manry, Dr. Ramtin Madani and Dr. Ioannis Schizas.

I would also like to thank all my colleagues at Virtual Environment Lab, my friends and relatives, especially Mahshid Grooms, Lida Karami, Hamid Parivash, Dr. Kanishka Tyagi, Vivek Nair, and Jennifer Lorraine Ball, all the EE department members and staff, especially Ms. Gail Paniusky, and the office of intenational education, especially Ms. Satu Birch for their various forms of support during my graduate study.

Finally, I would like to thank my family, my parents, my sister and my grandma, for their patience, sacrifice and encouragement. Without all your help and support, this would have never been possible. Thank you.

July 26, 2019

ABSTRACT

TOWARDS AUTOMATED UNDERSTANDING OF LAPAROSCOPIC VIDEOS

Babak Namazi, Ph.D. Candidate

The University of Texas at Arlington, 2019

Supervising Professor: Venkat Devarajan

Despite the advantages of minimally invasive surgeries, the indirect access and lack of the 3D field of view of the area of interest introduce complications in the procedures. Fortunately, the recorded videos from the operation offer the opportunity for intra-operative and post-operative analyses of the procedures, to improve future performance and safety. Such analysis is essential to provide the tools for evaluation and assessment of the surgeries. In this dissertation, we investigate the potential of deep learning techniques in understanding the videos captured during laparoscopic surgeries. To this end, we describe new methods for identifying the surgical instruments and the current phase of the procedure as well as the phase boundaries, which are the key components in understanding the work-flow of surgeries. Furthermore, we describe a method for analyzing and improving the safety in a laparoscopic cholecystectomy procedure by identifying the ”critical view of safety” (CVS), the recognition of which is the gold standard for enhancing the safety in cholecystectomy surgery. The tools developed under the dissertation could be the essential parts of a Surgical Video Analysis System (SVAS).

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Minimally invasive procedures include a variety of surgical techniques that are performed for the diagnosis or treatment for many conditions, with minimum damage to the patients' body. For instance, in minimally invasive surgeries (MIS), the goal is to reduce the size and the number of incisions in comparison to a traditional open surgery. In order to accomplish this objective, the laparoscopic instruments are inserted through hollow tubes called trocars. The procedure is monitored and assisted using specially designed cameras called endoscopes (figure 1.1).

As a result of avoiding large open wounds, the patients benefit from less pain and blood loss, and the recovery time is typically significantly shorter after laparoscopic surgery. Furthermore, operating within the body cavity reduces the risk of potential infections as in conventional open surgeries [88].

Despite these significant advantages, the unique way of manipulating the surgical instruments and the indirect observation of the surgical scene introduce more challenges in a laparoscopic operation [12]. These challenges include lack of depth perception, limited range of motion for the tools and lack of tactile sensation. Therefore, a significant amount of training is required for surgeons to reach the required proficiency.

In addition to the therapeutic benefits of laparoscopic surgery, the real-time monitoring of the procedure offers an opportunity for automatic computer vision techniques for improving the outcome of a surgery. As an example of the potential applications during the operation of laparoscopic cholecystectomy, the video analysis techniques can be used for feedback generation to improve the performance and/or the safety of the procedure.

Figure 1.1: Laparoscopic surgery [wikipedia.org/wiki/Laparoscopy]

Also, the videos recorded from the operations are valuable for the education and training of future surgeons. The analysis and assessment of the content of the videos are essential in information retrieval and future improved protocols.

To take advantage of the recorded videos, they need to be annotated. Manual annotation is a time-consuming task and requires experts' knowledge and assistance. Therefore, an automatic system is needed for performing various surgical video content analysis tasks.

In this dissertation, we investigate the problem of analyzing the laparoscopic procedures using the videos taken during the surgeries. In the following sections, we first review the potential applications of such analysis. The vision-based solutions are then discussed to address different sub-problems.

1.1 Applications of Surgical Video Understanding

The analysis of videos in laparoscopic procedures has numerous intra-operative and post-operative applications. Some of the potential, futuristic applications of an automated video analysis system during the surgery include:

- **Robotic surgeries**: Real-time surgical video /understanding analysis of the surgical scene currently known as computer-aided intervention (CAI) could replace the current human assistance provided to robotic surgical systems [52].
- **Feedback generation**: Surgical procedures are often technical and complex tasks and are susceptible to human errors. An automated system for generating real-time feedback can improve the outcome by giving reliable assistance to the surgeon [101].

The recorded videos can also be used in the following applications after the operation is performed:

- **Assessment**: Routine assessment of operation after the surgery is essential in improving surgical skills. An automated video understanding system [85] can accomplish an objective rating of the videos.
- **Information retrieval**: The recorded videos from the operation and the operating room are usually stored in large databases for future references. Manually organizing such datasets can be extremely costly and tedious. A video analysis system can ease the process of accessing the required information [8].
- **Education**: Videos are the best sources of information. A well-classified database of various tasks, accomplished by automated video analysis, can be an extremely useful tool for educating the future surgeons [62].

1.2 Vision-based Video Analysis

The automated understanding of a surgical video involves invoking several computer vision techniques to solve different sub-problems. Some examples of such analysis include surgical workflow recognition, monitoring the tools usage, surgical key-frame extraction, surgical shot classifications, surgical rating, and skill assessment. Some of these tasks require video frame-level understanding, while in others, the entire videos need to be processed.

In the literature, a plethora of methods has been proposed each addressing a specific problem in analyzing the surgical videos [59]. In most of these methods, a set of hand-crafted visual features are extracted to be used in making decisions for the particular task at hand. These heuristic techniques are based on decades of research in the computer vision area and are often designed for one or multiple specific tasks. Despite their relative success over manual or other signal-based methods [29, 73], most traditional computer vision methods are limited in their ability to perform general feature extraction directly from the data.

Over the past few years, most of the traditional computer vision algorithms have evolved into deep learning-based solutions [55]. This evolution has resulted in state-of-the-art performance in most computer vision tasks such as image [36] and video classification [48], object detection in images [74] and videos [47], activity recognition [41] etc. Therefore, there is a trend in using deep learning in analyzing the content of the videos from laparoscopic surgeries [104]. Some of the advantages of such methods include:

- Deep learning systems require minimal help from experts in designing a reliable model.
- The performance of most deep learning methods improve by introducing larger training data.

- Avoiding the manual feature extraction, deep learning models are more efficient and are suitable for real-time applications.

Inspired by their recent success, we investigate deep learning as a powerful solution for different sub-problems in laparoscopic video understanding.

1.3 Problem Statement

In this dissertation, we focus on simplifying the video analysis of laparoscopic surgeries using deep learning techniques. To prove the effectiveness of the proposed algorithms, we chose the laparoscopic cholecystectomy procedure or the gallbladder removal surgery, which is the most common laparoscopic procedure.

It turns out that deep learning can be employed in at least three or more sub-problems under the overall problem of cholecystectomy video analysis. Therefore, we proposed and developed four novel methods for solving these sub-problems. They are:

- Detecting the presence and the type of different surgical tools in all the frames of a video for intra-operative and post-operative applications.
- Identifying the current task or phase. Separate models are proposed for a) detecting the phase to which each video frame belongs b) segmenting the entire video by identifying the transition time of each phase.
- Detecting the critical view of safety (CVS), which is the gold standard for ensuring safety in a cholecystectomy procedure.

The methods are evaluated and proven to be state of the art, using large datasets containing videos from cholecystectomy surgery that are manually annotated by the experts. All the experiments in this dissertation are implemented in Tensorflow [1], and using Nvidia's Titan XP and 1080ti GPUs.

A broader objective is to investigate the potential for combining the methods to have an integrated future Surgical Video Analysis System that we called SVAS. An example use case for SVAS will be as follows: An attending surgeon would like to assess the performance of a resident on a rotation. The video corresponding to that surgery would then be loaded from the cloud. SVAS would be invoked to provide the options for the surgeon to find the phases of the operation, examine the correct use of the appropriate tools in various phases and grade the resident on the successful achievement of the critical view of safety. In the future, SVAS could produce detailed automated reports on the multiple aspects of the surgery, and generate real-time feedback during the operation.

1.4 Outline

The remainder of the dissertation is organized as follows: In chapter 2 the fundamentals of the deep learning techniques are reviewed. Chapter 3 describes the problem of tool detection in detail, provides relevant previous work, derives the logic of the development of our method and the corresponding results. Chapter 4 similarly describes the problem of detecting the workflow in separate sections for the identification of surgical phases for all the frames and the detection of the boundaries of each phase. Chapter 5 describes the critical view of safety detection. The final chapter concludes the dissertation and provides suggestions for future research.

CHAPTER 2

DEEP LEARNING

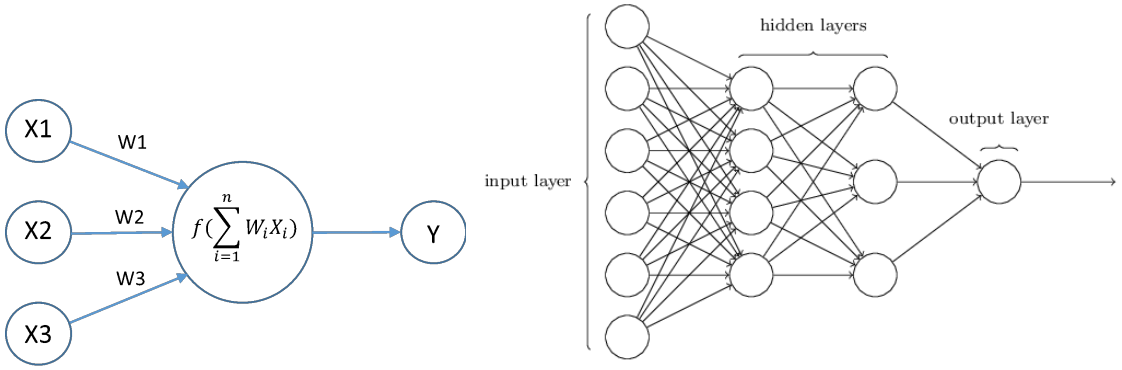
As a part of a broader family of machine learning, deep learning encompasses a variety of methods that are aimed at learning representations directly from data [55]. A deep learning model is composed of multiple layers for extracting multiple levels of representations, to be used for detection or classification. Over the past decade, these methods have improved the state-of-the-art in most challenging problems in the areas of computer vision [36, 51], speech processing [70] and natural language processing [116], and in different domains such as medical imaging [57].

This chapter provides fundamental knowledge regarding deep learning techniques used in this dissertation. The main components of most deep learning systems such as Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are described, followed by deep learning applications such as classification and regression, and video processing.

2.1 Deep Learning Components

2.1.1 Artificial Neural Networks

ANNs are computing systems that are inspired by the neural system in human brain. Each ANN consists of multiple neurons (*units*), which perform computation on their inputs. An example of an artificial neuron is shown in figure 2.1(a). The function f is called *activation function*. The output of a neuron is the weighted sum of the inputs plus a *bias*, which is then activated by the activation function.



(a) a single neuron (b) A deep neural network

Figure 2.1: Artificial Neural Networks

In a multi-layer neural network, for example, a Multi-Layer Perceptron (MLP), the neurons are formed in multiple layers, which pass messages from the previous layer to the next layer. An example of an ANN is shown in figure 2.1(b). The layers following the input layer are called *hidden layers* and the number of hidden layers is the *depth* of the network.

For an L-hidden-layer ANN, with m_l being the number of units in the lth layer, the output is calculated as:

$$F_W(x) = W_o^T f[W_L^T f(W_{L-1}^T ... f(W_1^T))] \tag{2.1}$$

where $W_l = (w_{l,0}, ..., w_{l,m_l})$ is the weight matrix for layer $0 < l \leq L$ ($w_{l,0}$ is the bias), and W_o is the output weight matrix. The collection of all weights W is the total trainable weights of an ANN.

The main role of the activation function f is to add non-linearity to form a more general mapping function. However, due to the requirements of the optimization process (to be explained in section 2.1.1.1) or the application, these functions need to have certain specifications such as differentiability. The most common activation functions include sigmoid,

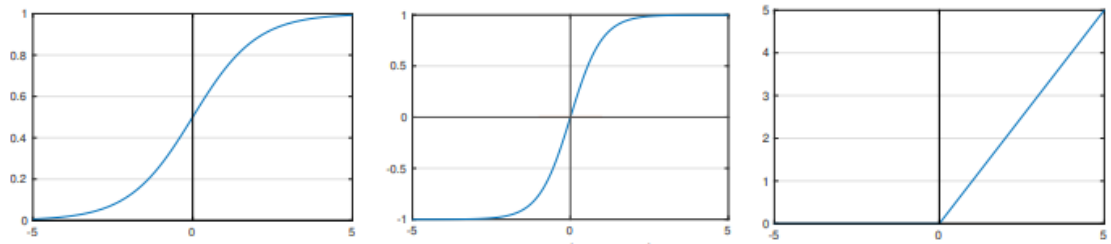


Figure 2.2: Common activation functions: sigmoid (left), tanh (middle), RELU (right)

hyperbolic tangent (tanh), and Rectified Linear unit (RELU). These functions are shown in figure 2.2.

2.1.1.1 Optimization

Of all of the different forms of deep learning algorithms, supervised learning has proven to be the most promising. The goal of any supervised learning algorithm is to find the best mapping from given inputs to their corresponding outputs, when the *ground-truth* is available. For ANNs, this is achieved by finding the best weights W that minimize an empirical cost (*loss*) function. Such optimization process is called *training*.

Given n training samples $(x_1, y_1)...(x_n, y_n)$ with y_i being the ground-truth for the ith sample x_i, the empirical cost minimization problem is formulated as:

$$\min_{W} \mathcal{L}(W) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, F_W(x_i)) \tag{2.2}$$

The function can be minimized by the Gradient Descent (GD) method, which updates the network parameters by computing the gradient of the loss function and taking steps proportional to the negative of the gradients at each iteration of the optimization process.

Given the initial weights matrices $W^{(0)}$, the update process at step k for each layer l can be written as:

$$W_l^{(k)} = W_l^{(k-1)} - \eta \nabla_{W_l} \mathcal{L}(W_l^{(k-1)}), \quad l = 1, ..., L \tag{2.3}$$

where $\eta > 0$ is the step size (*learning rate*), and ∇ is the gradient operation.

The gradients in the GD-based algorithms are calculated through the *backpropagation* method [75], which uses the chain rule to compute the gradients for each layer l starting from the last layer. In other words, the process of updating the weights is accomplished by repeatedly calculating the loss (error) through a forward pass and updating the gradients of each layer in a backward pass until convergence to a local or a global minimum is achieved.

The update rules in equation 2.3 require the cost function to be computed for all of the samples in the training set. Therefore, this method might not be efficient in terms of memory usage, when the size of the training set is large. In contrast, the Stochastic Gradient Descent (SGD) method updates the gradients for each sample or a *mini-batch* of size n_b, which is randomly selected from the total batch n. This method is proven to be faster and more efficient than GD for larger datasets.

The SGD algorithm is a first-order algorithm and has slower convergence due to the noisy nature of the gradients, especially if the *batch size* is too small. The inclusion of the *momentum* has been proven to increase the rate of convergence by taking the exponential weighted moving average of the update term of the current and previous step [71]. By adding the hyper-parameter β as the momentum, the method of SGD with momentum is formulated as:

$$\begin{aligned} W^{(k)} &= W^{(k-1)} - \eta \Delta W^{(k)} \\ \Delta W^{(k)} &= \beta \Delta W^{(k-1)} + (1 - \beta) \nabla_W \mathcal{L}(W^{(k)}) \end{aligned} \tag{2.4}$$

In order to improve the convergence of SGD based methods, several techniques have been proposed based on adaptive learning rate and momentum, such as AdaGrad [28], AdaDelta [117], RMSProp , ADAM [50], NADAM [26] etc. For instance, the update rules for ADAM are as follows:

$$W^{(k)} = W^{(k-1)} - \frac{\eta}{\sqrt{\hat{V}^{(k-1)}} - \epsilon}\hat{M}^{(k-1)} \tag{2.5}$$

where $\hat{V}$ and $\hat{M}$ are calculated as:

$$\begin{aligned} \hat{M}^{(k-1)} &= \frac{M^{(k-1)}}{1-\beta_1} \\ \hat{V}^{(k-1)} &= \frac{V^{(k-1)}}{1-\beta_2} \end{aligned} \tag{2.6}$$

and the estimate of the mean and the variance are:

$$\begin{aligned} M^{(k-1)} &= \beta_1 M^{(k-2)} + (1-\beta_1)\nabla_W \mathcal{L}(W^{(k-1)}) \\ V^{(k-1)} &= \beta_2 V^{(k-2)} + (1-\beta_2)[\nabla_W \mathcal{L}(W^{(k-1)})]^2 \end{aligned} \tag{2.7}$$

The hyper-parameters β_1 and β_2 are selected to be close to 1.

2.1.1.2 Regularization

Over-parameterization in deep neural networks happens because, in comparison to the parameters needed to create the approximation function 2.1, the number of input-output pairs aren't usually sufficient. Thus, deep learning models are prone to overfitting, which results when the model performs exceptionally well on the training set but fails to generalize for the unseen samples. In order to address this issue, several *regularization* techniques

are proposed to reduce the generalization error. The goal of almost all of the regularization methods is to reduce the *capacity* of the network or the optimization process.

L1 and L2 are the most common regularizers and are added to the cost function as below:

$$\mathcal{L}(W) = \mathcal{L}(W) + \lambda \sum \|W\|^n \tag{2.8}$$

where λ is the regularization parameter, and n can be 1 or 2. L2 is also known as weight decay (in most SGD based methods), since it reduces the weights' value.

As another popular regularization method, the Dropout [93] randomly turns off (drops out) some of the neurons during training. The Dropout technique temporarily splits the network into multiple parallel networks and therefore, is similar to the well-known ensembling methods [35].

Other regularization methods include but are not limited to data augmentation, which is applying random transformations and noise to the input to enhance the diversity of the training set, early stopping of the training process before the loss completely settles or starts to increase, and label smoothing [68] by randomly changing the confidence of the ground-truth.

2.1.1.3 Normalization

The goal of normalization techniques is to reduce the range of the input or activation maps at the layers of a deep neural network. Normalization methods have proven to be an essential component in improving the stability and convergence of the optimizer [33].

The simplest form of normalization methods is batch normalization [40], which performs normalization on the activation maps across each mini-batch. In order to accomplish this, the mean and variance of that feature are calculated. After that, the mean is subtracted from the features, and the result is divided by the standard deviation of the mini batch.

Despite it's early success, batch normalization has some limitations [83]. To address these limitations, other normalization methods such as weight normalization [80], layer normalization [10], instance normalization [106], group normalization [114] etc. have been proposed.

2.1.2 Convolutional Neural Networks

As a subset of ANNs, CNNs are designed to process arrays of data [54]. For example, a 1D CNN can be used for audio recognition [2], 2D for image processing [51] and 3D CNN for video processing [41]. A CNN is composed of multiple convolutional layers, pooling layers, and fully connected layers. The architecture of a typical CNN is shown in figure 2.3.

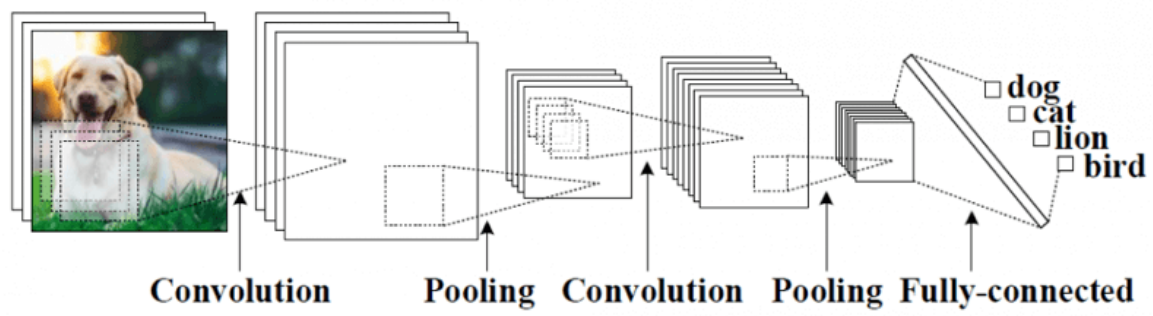


Figure 2.3: A generic convolutional neural network [20]

In a deep CNN, each layer is responsible for extracting features at a certain level. The lower level features such as edges are detected by the first layers, followed by more abstract and higher-level features towards the end of the network. The architecture of a CNN is inspired by the hierarchical structure of our visual cortex [39].

Convolution is at the core of most image processing methods such as edge detection, deblurring, etc. The convolution operation on an image is performed by applying square-

shaped *kernels*. Similarly, in a convolutional layer of a CNN, multiple kernels are stacked together, forming a trainable *filter*. The main ideas behind CNNs are the local connections of the convolutional filters to the input and the sharing of the trainable weights among them. The output of each convolutional layer is a set of *feature maps*. The non-linearity is then added to the feature maps using an activation function such as RELU.

Besides the size of the convolutional filters, the *stride* and the *padding* parameters have to be chosen before adding a convolutional layer. Stride determines how the filter is being slid over a feature map. The stride value of higher than one is used for down-sampling as well. The padding is applied to the borders to control the size of the output.

The role of pooling in CNN is to reduce the number of trainable parameters in a network. For instance, max-pooling is a common pooling technique, which aims at reducing the dimensionality of the feature maps while keeping the most dominant features. Another form of pooling layer is average pooling. The choice of pooling layer depends on the application.

A fully connected layer is usually added at the end of a CNN and after the convolutional and pooling layers. As was mentioned before, the extracted feature maps are the higher-level representations of the input and contain the discriminative features useful for a particular task such as image classification. The impact of each part of the extracted feature in the classification problem is learned using a fully connected layer.

2.1.2.1 CNN architectures

Despite the great advantages of CNNs, it took more than a decade since the introduction of the first CNN architecture [54], to find the popularity they have today. The two main factors that hindered such progress were the lack of computational power and large labeled datasets. The recent advances in hardware and parallel processing, especially in Graphical Processing Units (GPUs) have had a great impact on the success of deep learn-

ing. Furthermore, online datasets and competitions such as Imagenet (Large Scale Vision Recognition Challenge) [76] provide a great opportunity for researchers to develop their ideas using large, manually annotated datasets.

Alexnet [51] was one of the biggest breakthroughs in the area of pattern recognition and machine learning. The architecture of Alexnet consists of five convolutional layers, followed by three fully connected layers. The architecture is shown in figure 2.4. Since Alexnet, several architectures have been proposed, such as VGGnet [92], which introduced changes such as adding more convolutional layers or better training techniques.

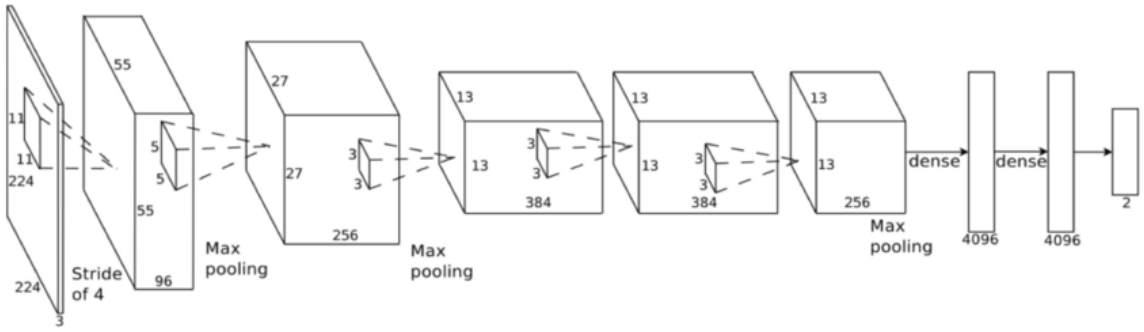


Figure 2.4: Alexnet architecture [67]

In 2014, a new architecture was proposed by [98]. The main idea was to learn the data representations through multiple paths by introducing convolutional structures called Inception blocks. On each Inception block, several convolutional layers with different kernel sizes are placed. The structure of the blocks are shown in figure 2.5 [98]. The architecture is further refined in Inception-v3 [99] and Inception-v4 architectures [97].

Another breakthrough in deep learning and CNN happened in 2015 with the introduction of the skip (residual) connections in the Resnet architecture [36]. The main advantage of residual connections is that it provides a shortcut path for the gradients to flow to the first layers in the backpropagation steps. This prevents the vanishing gradient phenomenon

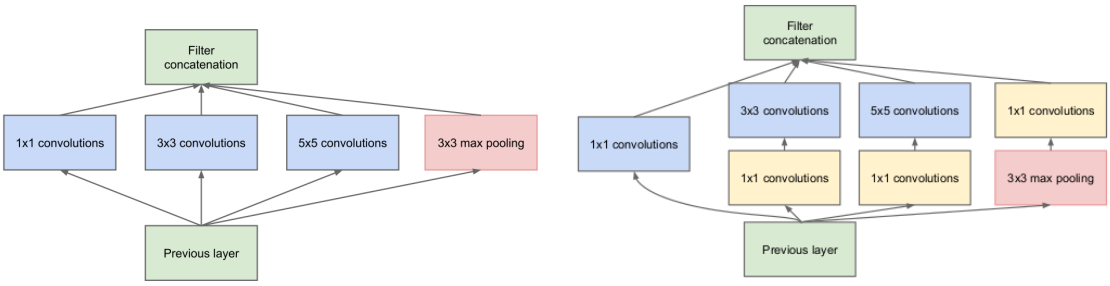


Figure 2.5: Inception blocks a) naive structure b) with dimension reductions [98]

from occurring, and therefore, deeper architectures of even up to 1000 layers, is possible with residual connections.

2.1.3 Recurrent Neural Networks

Similar to CNNs, RNNs benefit from sharing the weights for processing arrays of data [34]. The difference is that using RNNs, the elements of the data such as frames of a video are processed one at a time and thus, they are useful for data that has sequential forms such as text [11], audio [46], video [25], and time series [22]. The key idea is that the information from the previous elements are maintained in RNNs' hidden state and are used for extracting the correlation among the elements in order to make predictions on the current element. Another advantage of RNNs is that unlike feed-forward networks such as MLP and CNN, the architecture of the model does not depend on the size of the input and, a single RNN can process sequences with variable length.

The block diagram of a generic RNN is shown in figure 2.7. From the unfolded diagram, it can be seen that each RNN cell has two inputs; one coming directly from the inputs and the other one is the output (hidden state) of the previous time step (state). The output of an RNN layer can be either the hidden states of individual elements or the last hidden state. One example of the former paradigm is the classification of an individual

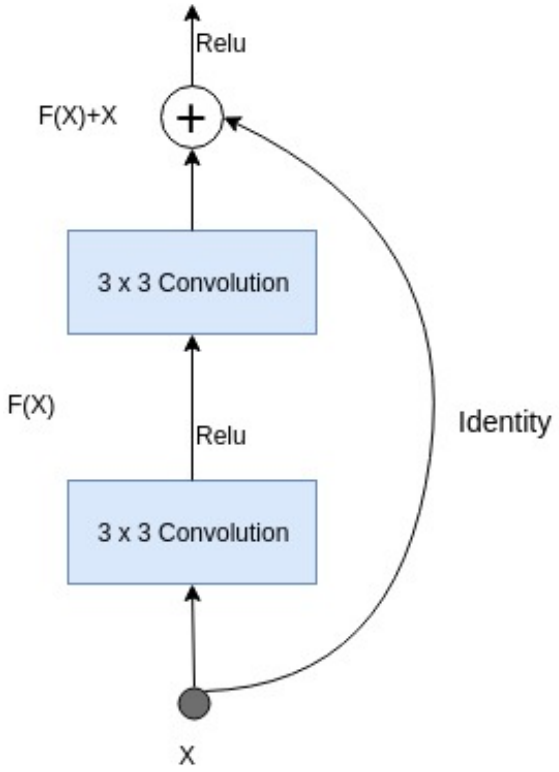


Figure 2.6: Residual connection in Resnet CNN

sentence in a text, while the latter design can be used for the classification of the whole text.

The training of an RNN is performed by a method called backpropagation through time (BPTT) [111]. Similar to the conventional BP method, the gradients are calculated using chain rules and based on the errors in the output sequence, given a sequence ground truth. To this end, the RNN is unrolled (figure b), and the gradients are calculated for each time step. The network is then rolled back and the weights are updated using the optimization equations 3.6. The issue with BPTT is that, as the number of elements in the input sequences increases, the computation and memory needed for processing the gradients increases as well. Moreover, the training might get affected by the exploding or vanishing gradients problems. In order to address this issue, truncated BPTT (TBPTT) is proposed

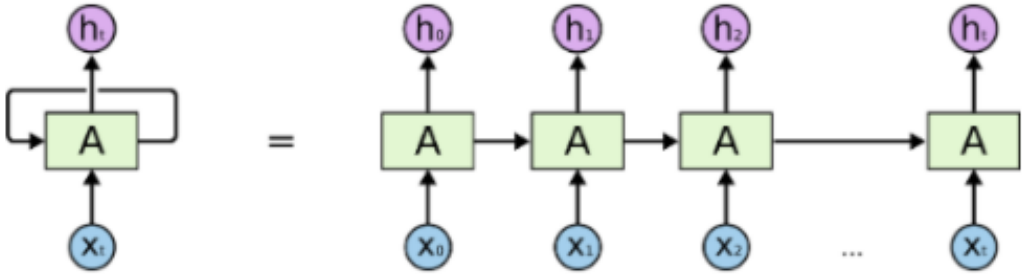


Figure 2.7: A generic RNN [18]

which updates the gradients every k_1 steps for k_2 timesteps.

2.1.3.1 RNN architectures

One of the main issues with RNNs is that the long term dependencies might not be captured using the generic RNN explained in the previous section. The reason is the vanishing gradient [14]. To address this issue, several modifications have been proposed in the literature [23, 37]. The key idea behind all of them is the residual connection between the states.

One of the most widely used RNN architectures is called long short-term memory (LSTM) [37], which is designed for extracting the pattern in longer input sequences. LSTM introduces the ideas of cell states and soft gates, which are used to decide on the usefulness of the information from the past. The cell state works as a skip connection and is responsible for propagating the derivatives to the earlier states. The architecture is shown in figure 2.8.

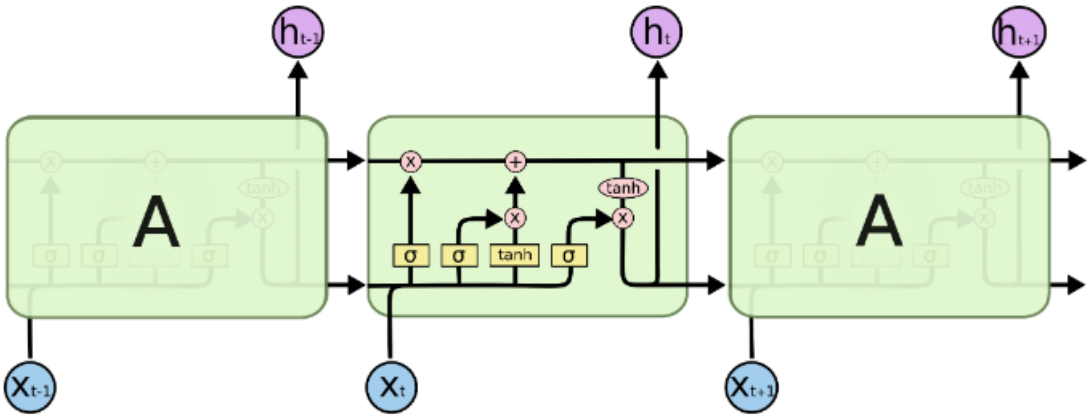


Figure 2.8: LSTM architecture [18]

Given input x_t at time step t, and the hidden state from the previous step h_{t-1}, the output at state t is calculated as:

$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= tanh(W_C[h_{t-1}, x_t] + b_C) \\
C_t &= f_t \star C_{t-1} + i_t \star \tilde{C}_t \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
h_t &= o_t \star tanh(C_t)
\end{aligned} \tag{2.9}
$$

f is called the forget gate and using the sigmoid function, it decides which information to be removed from the previous cell state C_{t-1}. The input gate i is responsible for deciding which information is to be retained in the cell state. Based on the candidate values $\tilde{C}_t$, the input and forget gates and the previous cell state, the cell state C_t is updated. o is called the output gate and is responsible for deciding the output of the cell, which is the hidden state

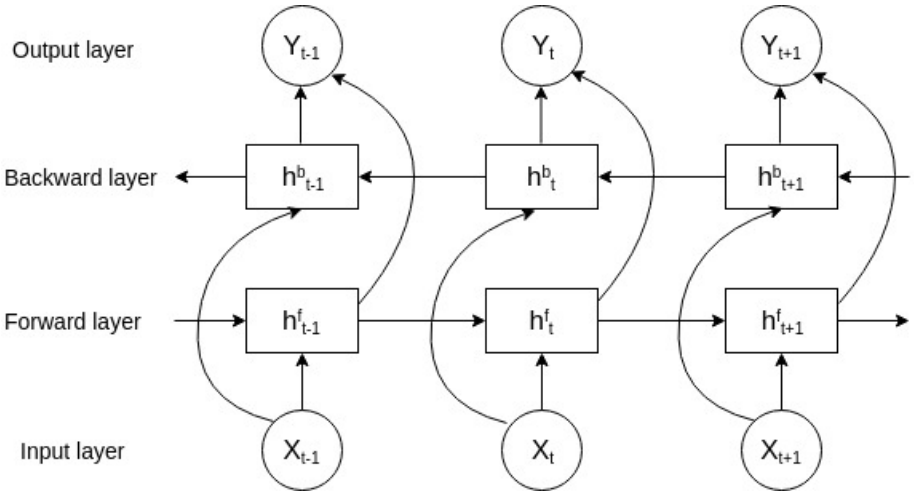


Figure 2.9: A bi-directional RNN

h_t, based on the current cell state. W_f, W_i, W_C and W_o are the trainable weights, and b_f, b_i, b_C and b_o are the biases for the forget, input, cell and output gates respectively.

A modified version of LSTM is called Gated Recurrent Unit (GRU) [23], which merges the cell state and the hidden state. The input gate and the forget gate are combined to decide which information is to be retained from the previous state. The candidate hidden states are updated based on the reset gate. The current hidden state is then calculated based on the previous hidden state, update gate and the hidden state candidates. Using the three-gates GRU is simpler than the LSTM and has shown superior performance in some cases.

In the RNN models discussed above, the output at each time step depends on the previous element of a sequence. To take advantage of the correlation among all of the elements of sequences from both the past and the future of the current input, a bi-directional RNN has been proposed [86]. A bi-directional RNN is simply a concatenation of two uni-directional RNNs where one of the sequences is the reverse of the other one. The block diagram of a bi-directional RNN is shown in figure 2.9. Bi-RNNs are useful in an offline and non-casual application when the entire sequence is available for training and making predictions.

2.2 Deep Learning Applications

In the following section, some applications of deep learning are described.

2.2.1 Regression and Classification

Most of the problems in computer vision can be defined as a regression or classification problem. The goal of a regression problem is to find the best mapping function to make a prediction from a continuous space. The widely used loss function for a regression problem is the mean square error and is defined as:

$$\mathcal{L}_{MSE} = \frac{1}{n}\sum_{i=1}^{n}[F_W(x_i) - y_i]^2 \tag{2.10}$$

For classification problems, logistic regression is used. Cross-entropy loss is the most widely used cost function for a classification problem:

$$\mathcal{L}_{CE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=0}^{C} y_i^{(c)} \log F_W(x_i)^{(c)} \tag{2.11}$$

where C is the total number of classes and y is one-hot vector of the ground truth (only the correct class is 1).

The general case for a classification problem is multi-class classification. The common strategy for a multi-class problem is one-vs-all, where only one class can be accepted as positive, while the rest of the classes are negative. In neural networks, the last layer is usually activated by the *softmax* function, which is the normalized logistic function resulting in a probability distribution and is formulated as:

$$\sigma(Z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad for i = 1, ..., K. \tag{2.12}$$

and $Z = [z_1, ...z_K]$. Softmax is a smooth (soft) approximation of the maximum function and the final prediction of ANN model is the argument of the maximum probability.

2.2.2 Video Processing

One of the most important applications of deep learning techniques is video processing and is the focus of this dissertation. Some of the sub-problems of a video processing system include video classification [48], action recognition and localization [90, 91], video segmentation, video frames' classification, object detection and segmentation in videos [31, 47], video generation [19], etc.

In a typical video, there is a high correlation in the sequence of images and among the neighboring frames. To take advantage of the structure of the information in a video, a deep learning system needs to consider the spatial visual information, while utilizing the dependencies among the frames. For this purpose, several approaches are accepted in the literature. For instance, RNNs can be used to capture the correlations in sequences of frames in a video.

Another solution for extracting spatiotemporal features in a video is to use 3D CNNs [100]. Using a 3D CNN, the pattern can be extracted in three dimensions of the width and height of the image and time. The main challenge in using 3D CNN is that it is computationally expensive.

While RNNs and 3D CNNs utilize the dependencies for the spatial features, they don't directly consider the temporal features such as motion. To capture the motion feature in a video, optical flow and trajectory are the conventional solutions. Recently, deep learning methods have been used for calculating the optical flows as well [102]. In order to incorporate the motion features, the two-stream model [91] has been proposed. In two-stream models, one stream is responsible for extracting spatial features using a CNN,

whereas, the other stream extracts motion features from a stack of neighboring frames. The streams are fused, and the decision is made based on the fused features.

CHAPTER 3

SURGICAL TOOL DETECTION

Tracking surgical tools is essential in understanding the workflow of a procedure and is important in the assessment and rating of the videos. Manual annotation of long videos from surgeries is a time-consuming and expensive task. A vision-based algorithm for automated detection of the presence, location, or movement of surgical tools can be extremely useful in designing a fast and objective surgical evaluation system. A well-annotated database of surgical videos can also be used in information retrieval and is a reliable source for education and training of the future surgeons.

During the operation, monitoring the usage of surgical tools can provide real-time feedback to the surgeons and operating room staff. Furthermore, in computer-aided intervention, the surgical tools are controlled by a surgeon with the aid of a specially designed robot [9], which requires a real-time understanding of the current task. Therefore, detecting the presence, location, and pose of the surgical instruments may be useful in robotic surgeries as well [27], [7], [6]. Finally, an automated tool usage detector can be useful in generating an operative summary.

To track the surgical instruments, several approaches have been introduced, which use the signals collected during the procedure [29], [73]. For instance, in vision-based methods, the instruments can be localized using the videos captured during the operation [112]. These methods are generally reliable and inexpensive. Traditional vision-based methods rely on extracted features such as shape, color, the histogram of oriented gradients, etc., along with a classification or regression method to estimate the presence, location or pose of the instrument in the captured images or videos [15]. However, these methods are

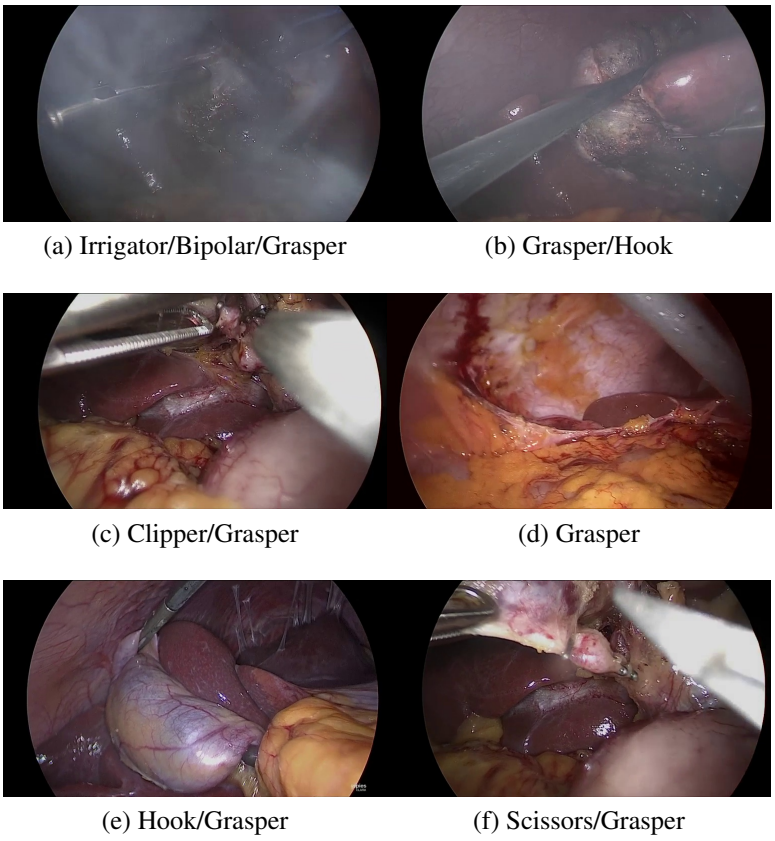
dependent on pre-defined and painstakingly extracted hand-crafted features. Just logically defining and extracting such features alone is a major part of the detection process. Thus, these hand-crafted features and designs are not suitable for real-time applications.

Compared with the other surgical video tasks, detecting the presence and usage of surgical instruments in laparoscopic videos has certain challenges that need to be considered.

Firstly, since multiple instruments might be present at the same time, detecting the presence of these tools in a video frame is a multilabel (ML) classification problem. In general, ML classification is more challenging compared to the well-studied multiclass (MC) problem, where every instance is related to only one output. These challenges include but are not limited to using correlation and co-existence of different objects/concepts with each other and the background/context and the variations in the occurrence of different objects.

Secondly, as opposed to other surgical videos, such as cataract surgery [5], robot-assisted surgery [84] or videos from a simulation [120], where the camera is stationary or moving smoothly, in laparoscopic videos, the camera is constantly shaking. Due to the rapid movement and changes in the field of view of the camera, most of the images suffer from motion blur, and the objects can be seen in various sizes and locations. Also, the camera view might be blocked by the smoke caused by burning tissue during cutting or cauterizing to arrest bleeding. Therefore, using still images is not sufficient for detecting the instruments. Some of the challenges caused by still images are illustrated in figure 3.1.

Thirdly, surgical operations follow a specific order of tasks. Although the usage of the tools doesn't strictly adhere to that order, it is nevertheless highly correlated with the task being performed. The performance of the tool detection can be improved with the information about the task and the relative position of the frame with regard to the entire video.



(a) Irrigator/Bipolar/Grasper (b) Grasper/Hook

(c) Clipper/Grasper (d) Grasper

(e) Hook/Grasper (f) Scissors/Grasper

Figure 3.1: Challenges in detecting surgical tools due to smoke (a,b), motion blur (c,f), lack of focus (d) and occlusion (e) in laparoscopic cholecystectomy procedure.

Lastly, since the performance of a deep classifier in a supervised learning method is highly dependent on the size and the quality of the labeled dataset, collecting and annotating a large dataset is a crucial task.

Endonet [104] was the first deep learning model designed for detecting the presence of surgical instruments in laparoscopic videos, wherein Alexnet [51] was used as the CNN, for feature extraction and is trained for the simultaneous detection of surgical phases and instruments. Inspired by this work, other researchers used different and more accurate CNN architectures with transfer learning [78], [69] to classify the frames based on the visual

features. For example, in [118], three CNN architectures are used, and [110] proposed an ensemble of two deep CNNs.

[79] were the first to address the imbalance in the classes in a ML classification of video frames. They balanced the training set according to the combinations of the instruments. The data were re-sampled to have a uniform distribution in label-set space and, class re-weighting was used to balance the data in a single class level. Despite the improvement gained by considering the co-occurrence in balancing the training set, the correlation of the tools usage was not considered directly in the classifier and the decision was made solely based on the presence of single tools. [3] used class weights and re-sampling together to deal with the imbalance issue.

In order to consider the temporal features of the videos, Twinanda et al. employed a hidden Markov model (HMM) in [104] and Recurrent Neural Network (RNN) in [105]. Sahu et al. utilized a Gaussian distribution fitting method in [78] and a temporal smoothing method using a moving average in [79] to improve the classification results after the CNN was trained. [61] were the first to apply an LSTM to a short sequence of frames, to simultaneously extract both spatial and temporal features for detecting the presence of the tools by end-to-end training.

Other papers invoked different approaches to address the issues in detecting the presence of surgical tools. [38] proposed an attention guided method using two deep CNNs to extract local and global spatial features. In [4], a boosting mechanism was employed to combine different CNNs and RNNs. In [42], the tools were localized by Faster RCNN [74] method, after labeling the dataset with bounding boxes containing the surgical tools.

It should be noted that none of the previous methods takes advantage of any knowledge regarding the order of the tasks and, the correlations of the tools are not directly utilized in identifying different surgical instruments.

In this chapter, we propose a novel system called LapTool-Net to detect the presence of surgical instruments in laparoscopic videos. The main features of the proposed model are summarized as follows:

1. Exploiting the spatial discriminating features and the temporal correlation among them by designing a deep Recurrent Convolutional Neural Network (RCNN)
2. Taking advantage of the relationship among the usage of different tools by considering their co-occurrences
3. The end-to-end training of the tool detector using a multitask learning approach
4. Considering the inherent long-term pattern of the tools presence via an RNN in online and offline modes
5. Using a small portion of the labeled samples considering the high correlation of the video frames to avoid overfitting
6. Addressing the imbalance issue using re-sampling and re-weighting methods
7. Providing state-of-the-art performance on a publicly available dataset on laparoscopic cholecystectomy

The remainder of the chapter is organized as follows: the main approach of the proposed model is described in section 3.1, and is elaborated in section 3.2. The performance is evaluated through experiments in section 3.3. Section 3.4 concludes the chapter.

3.1 Approach

The uniqueness of our approach is based on the following three original ideas:

- A novel ML classifier is proposed as a part of LapTool-Net, to take advantage of the co-occurrence of different tools in each frame in other words, the context is taken into account in the detection process. In order to accomplish this objective, each

combination of tools is considered as a separate class during training and testing and, is further used as a decision model for the ML classifier. To the best of our knowledge, this is the first attempt at directly using the information about the co-occurrence of surgical tools in laparoscopic videos in the classifier's decision-making.

- The ML classifier and the decision model are trained in an end-to-end fashion. For this purpose, the training is performed by jointly optimizing the loss functions for the ML classifier and the decision model using a multitask learning approach
- At the post-processing step, the trained model's prediction for each video is sent to another RNN to consider the order of the usage of different tools/tool combinations and long-term temporal dependencies, yet another consideration for the context.

The overview of the proposed model is illustrated in Fig. 1. Let $\mathcal{D} = \{(x_{ij}, Y_{ij}) | 0 \leq i < m, 0 < j < n\}$ be a ML dataset, where $x_{ij} \in \Re^d$ is the ith frame of the jth video and $Y_{ij} \subseteq \mathcal{Y}$ is the corresponding surgical instruments and $\mathcal{Y} \triangleq \{y_1, y_2, ...y_K\}$ is the set of all possible tools. Each subset of $\mathcal{Y}$ is called a label-set and each frame can have a different number of labels $|Y_{ij}|$. The tools associations can also be represented as a K dimensional binary vector $y_{ij} = (y_1, y_2, ..., y_K) = \{0, 1\}^K$, where each element is a 1 if the tool is present and a 0 otherwise. The goal is to design a classifier $F(x)$ that maps the frames of surgical videos, to the tools in the observed scene.

In order to take advantage of the combination of the surgical tools in a laparoscopic video, the well-known label power-set (LP) method is adopted in a novel way. The output of $F(x_{ij})$ is a label-set $\hat{Y}_{ij} \subseteq \hat{\mathcal{Y}}$ (also called a superclass) of size $|\hat{Y}_{ij}| \leq K$, where $\hat{\mathcal{Y}}$ is the set of all possible subsets of $\mathcal{Y}$.

In order to calculate the confidence scores for each tool, along with the final decision, which is the class index in $\hat{\mathcal{Y}}$, the classifier F is decomposed into $F(.) = g(f(.))$, where $g(f(x_{ij})) : \Re^K \rightarrow \Re^{\hat{K}}$ is the decision model, which maps the confidence scores of the frame i of the video j to the label-set $\hat{Y}_{ij}$. The model f takes the video frames as input

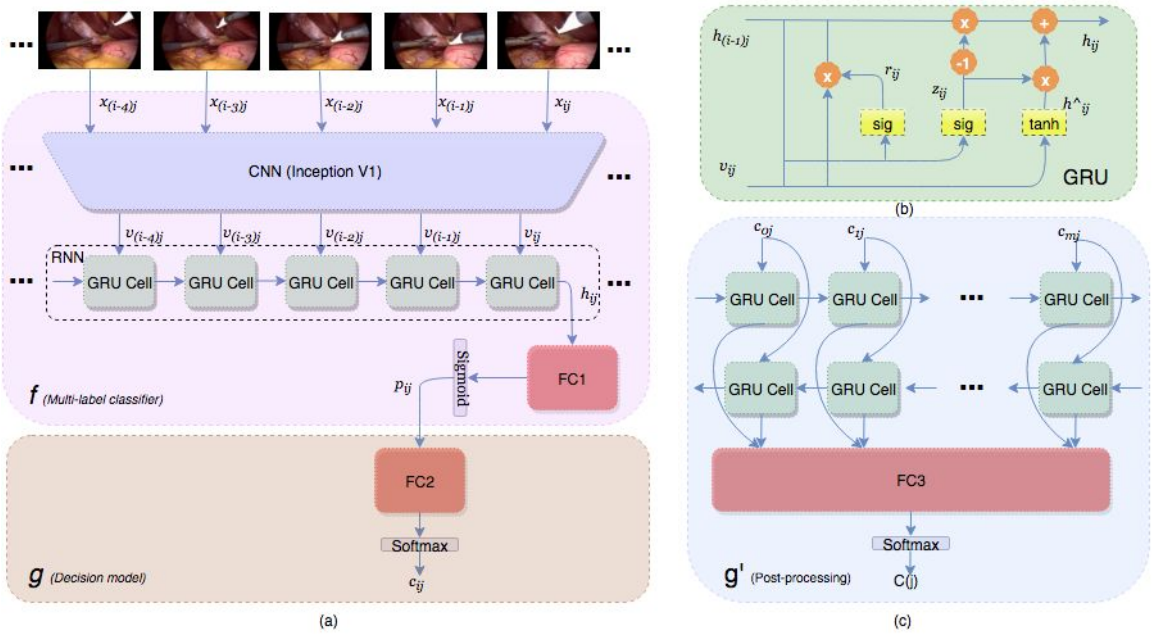


Figure 3.2: Block diagram of a) the proposed multiclass classifier F which consists of f and g, b) the architecture for Gated Recurrent Units (GRU) and c) The bi-directional RNN for post-processing.

and produces the confidence scores $P = (p_1, p_2, ...p_K) = [0, 1]^K$, where each element is the probability of the presence of one tool from the set $\mathcal{Y}$. It's worth mentioning that f is as an ML classifier, while the output of the decision model g is the label-set and therefore, classifier F is an MC classifier based on the LPs.

The ML classifier f consists of a CNN and an RNN. The CNN is responsible for extracting the visual features, while the RNN uses the sequence of features extracted by CNN and calculates the confidence scores P.

The output of the decision model g for all the frames of each video forms a larger sequence $\bar{C}$ of the models predictions. The sequence is used as the input to another RNN, g' to exploit the long-term order of the tool usage.

Figure 3.3: List of the tools used in M2CAI16 cholecystectomy dataset [104]

The overall system is designed and tested using the dataset from M2CAI16[1] tool detection challenge1. The dataset contains 15 videos from cholecystectomy procedure, which is the surgery for removing the gallbladder. All the videos are labeled with seven tools for every 25 frames. The tools are Bipolar, Clipper, Grasper, Hook, Irrigator, Scissors, and Specimen bags, and are shown in figure 3.3 . There are ten videos for training and five videos for validation. The type and shape of all seven tools remain the same for the training and validation sets.

The performance of LapTool-Net is measured through common metrics for ML and MC classification, and a comparison is made with the current methods. The methodology derived from this approach is provided in more detail in the following section.

3.2 Methodology of LapTool-Net

3.2.1 Multi-label Classification

In ML classification, the goal is to assign multiple labels to each image. Higher dimensionality in label space and the correlation between the labels make the ML classification more challenging compared with MC problems. In the literature, two main approaches to deal with such issues in ML classification are accepted [32]. One approach is called adaption, which aims at adapting existing machine learning models to deal with the requirements of ML classification. Since the output of an ML classifier is the confidence

[1] http://camma.u-strasbg.fr/m2cai2016/index.php/program-challenge

scores for each class, a decision policy is needed to make the final prediction. This decision is usually made based on top-k or thresholding methods.

The second paradigm for ML classification is based on problem transformation. The goal of problem transformation is to transform the ML problem into a more well-defined binary or MC classification. The most popular methods include binary relevance (BR), chain of classifiers [72] and LP. In BR, the problem is transformed into multiple binary classifiers for each class. This method doesn't take the dependencies of the classes into account. On the other hand, in a classifier chain, the binary classifiers are linked in a chain to deal with the classes correlations. In LP, multiple classes are combined into one superclass and the problem is transformed into an MC classification. The advantage of this method is that the class dependencies are automatically considered. However, as the number of classes increases, the complexity of the model increases exponentially. This is not an issue in laparoscopic videos. The reasons are 1) there is a limit for the number of tools in each frame (usually 3 or 4) and 2) the combinations of the tools are known. Since the LP method is more efficient than the classifier chain due to the use of just one classifier, it was determined to be more efficient for detecting the usage of surgical tools. Thus, we propose a novel classifier with LP being the decision layer for an ML classifier.

3.2.2 Spatio-temporal Features

In order to detect the presence of surgical instruments in laparoscopic videos, the visual features (intra-frame spatial and inter-frame temporal features) need to be extracted. We use CNN to extract spatial features. A CNN consists of an input layer, multiple convolutional layers, non-linear activation units, and pooling layers, followed by a fully connected (FC) layer to produce the outputs, which are typically the classification results or confidence scores. Each layer passes the results to the next layer, and the weights for the convolutional and FC layers are trained using backpropagation to minimize a cost function.

The output of the last convolutional layer is a lower dimensional representation of the input and therefore, can be considered as the spatial features. As shown in Fig 3.2, the input frame x_{ij} is sent through the trained CNN and the output of the last convolutional layer (after pooling) forms a fixed size spatial feature vector v_{ij}.

In the literature, several approaches have been proposed for utilizing the temporal features in videos for tasks such as activity recognition and object detection in videos [48], [91]. For instance, when there is a high correlation among video frames, it can be exploited to improve the performance of the tool detection algorithm.

An RNN is typically used to exploit the pattern of the usage of the instruments. It uses its internal memory (states) to process a sequence of inputs for time series and videos processing tasks [44]. Although the motion features are not explicitly extracted when using the RNN, the temporal features are exploited through the correlation of spatial features in the neighboring frames.

Since the point of the RNN along with the CNN is to capture the high correlation among the neighboring frames, short sequences of frames (say five frames) are selected. Also, shorter sequences help the RNN have a better and faster convergence.

For each frame x_{ij}, the sequence of the spatial features $V_{ij} = [v_{(i-\lambda\Delta t)j}...v_{(i-\Delta t)j}v_{ij}]$ is the input for the RNN, where the hyper-parameters λ and Δt are the number of frames in the sequence and the constant inter-frame interval respectively. The total length of the input is no longer than one second, which ensures that the tools remain visible during that time interval. Since the tool detection model is designed to be causal and to perform in real-time, only the previous frames with respect to the current frame can be used with the RNN.

We selected GRU [23] as our RNN for its simplicity. The architecture is illustrated in Fig. 3.2.(b) and formulated as:

$$
\begin{aligned}
z_{ij} &= \sigma(v_{ij}U^z + h_{i-\Delta t,j}W^z), \\
r_{ij} &= \sigma(v_{ij}U^r + h_{i-\Delta t,j}W^r), \\
\tilde{h}_{ij} &= tanh(v_{ij}U^h + (r_{ij} \odot h_{i-\Delta t,j})W^h), \\
h_{ij} &= (1 - z_{ij}) \odot h_{i-\Delta t,j} + z_{ij} \odot \tilde{h}_{ij},
\end{aligned} \tag{3.1}
$$

where U and W are the GRU weights, $\odot$ is the element-wise multiplication, and σ is the sigmoid activation function. z and r are update gate and reset gate respectively. The final hidden state h_{ij} is the output of the GRU and is the input to a fully connected neural network $FC1$. The output layer $FC1$ is of size K (the number of tools) and after applying the sigmoid function, produces the vector of confidence scores $P(ij)$ for all classes.

We designed the above RCNN architecture as the ML classifier model f shown in Fig. 3.2.a, which exploits the spatiotemporal features of a given input frame and produces the vector of confidence scores of all the tools, which in turn is the input to the decision model g.

3.2.3 Decision Model

One of the main challenges in ML classification is effectively utilizing the correlation among different classes. Using LP (as described earlier), uncommon combinations of the classes will automatically be eliminated from the output, and the classifier's attention is directed towards the more possible combinations.

As mentioned before, not all the 2^K combinations are possible in laparoscopic surgery. Figure 3.4 shows the percentage of the most likely combinations in the M2CAI dataset. The first 15 classes out of a possible maximum of 128 span more than 99.95% of the frames in

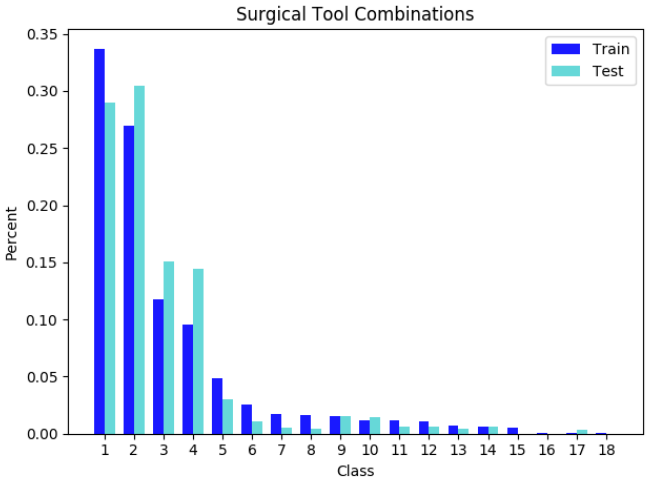


Figure 3.4: The distribution for the combination of the tools in M2CAI dataset

both the training and the validation sets and the tools combinations have almost the same distribution in both cases. The 15 combinations are:

[Hook], [Grasper and Hook], [Grasper], [no tool], [Grasper and Specimen bag], [Clipper], [Grasper and Irrigator], [Bipolar and Grasper], [Grasper and Clipper], [Grasper and Irrigator], [Irrigator], [Bipolar], [Grasper, Irrigator and Specimen bag], [Scissors and Grasper], and [Scissors].

Since an LP classifier is MC, the cost function for training a machine learning algorithm has to be the conventional one-vs-all (categorical) loss. For example, Softmax cross-entropy (CE) is the most popular MC loss function. However, Softmax CE requires the classes to be mutually exclusive, which is not true in the LP method. In other words, while using a Softmax loss, each superclass is treated as a separate class, i.e., separate features activate a superclass. This causes performance degradation in the classifier and therefore, more data is required for training. We address this issue by a novel use of LP as the decision model g, which we apply to the ML classifier f. Our method helps the classifier to consider our superclasses as the combinations of classes rather than separate mutually exclusive classes.

The decision model is a fully connected neural network ($FC2$), which takes the confidence scores of f and maps them to the corresponding superclass. When the Softmax function is applied, the output of $g(.)$ is the probability of each superclass $Q = (q_1, q_2, ..., q_{\hat{K}})$, where $\hat{K}$ is the size of the superclass set. The final prediction of the tool detector F is the index of the superclass with the highest probability and for frame i of video j is calculated as:

$$c_{ij} = argmax(Q_{ij}) \tag{3.2}$$

3.2.4 Class Imbalance

Class imbalance has been a well-studied area in machine learning for a long time [16]. It is known that in skewed datasets, the classifier's decision is inclined towards the majority classes. Therefore, it is always beneficial to have a uniform distribution for the classes during training. Two major approaches have been proposed in the literature to deal with imbalanced datasets. One approach is called cost-sensitive and is mainly based on class re-weighting. In these methods, the output of the classes or the loss function during training is weighted based on the frequency of the classes. Although this approach works in some cases, since the complexity of the input is not known before training, the choice of the weights might not depend solely on the distribution of the data. Thus, class weights are another hyper-parameter that needs to be taken care of. Another solution is to change the distribution in the input of the model. This can be done using over-sampling/up-sampling for the minority classes and under-sampling/down-sampling for the majority classes.

The number of samples for each tool before balancing is shown in table 3.1. Hook has the highest number of frames, due to it's longer usage in dissection tasks, while Scissors have the lowest usage time.

Table 3.1: Number of frames for each tool in M2CAI

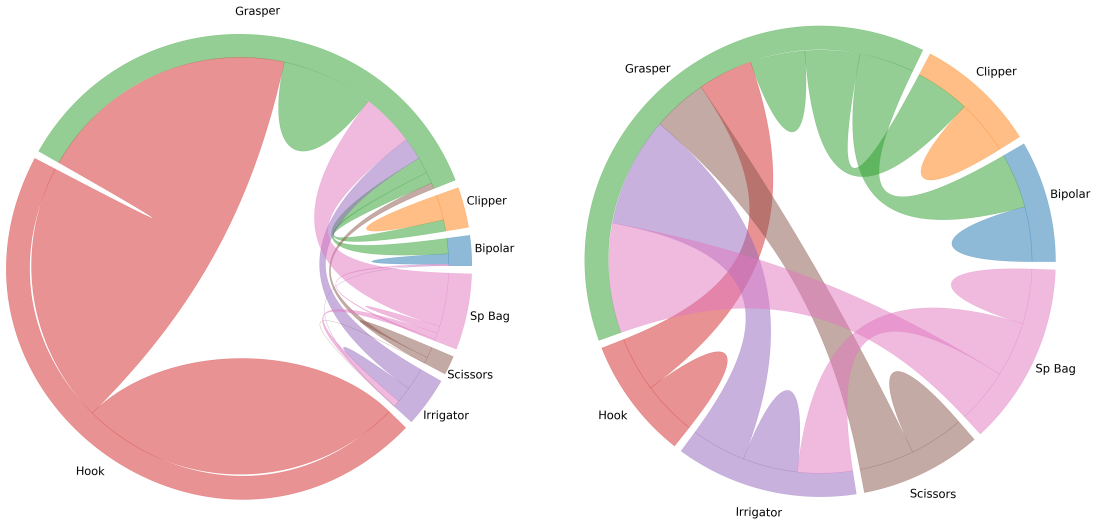
Tool	Train	Validation
Bipolar	631	331
Clipper	878	315
Grasper	10367	6571
Hook	14130	7454
Irrigator	953	131
Scissors	411	158
Specimen Bag	1504	483
no tools	2759	1888
total	23421	12512

In surgical videos, there is a high correlation between the neighboring frames. Therefore, under-sampling can be used to both balance the classes and avoid over-fitting. Using LP, we perform under-sampling to have a uniform distribution of the combination of the labels. Since this approach will not guarantee the balance in each class, a cost-sensitive weighting approach can be used along with a multi-label loss function prior to the LP decision layer; nonetheless, we empirically found that this doesn't affect the performance of the multi-label classifier.

Figure 3.5 shows the relationship of the tools after re-sampling. It can be seen that the LP-based balancing method, not only have a uniform distribution in superclass space, it also improves the balance of the dataset in single class space (with the exception of Grasper which can be used with all the tools).

In multi-label classification, finding the balancing criteria for re-sampling is challenging, since a change in the number of samples for one class might affect other classes as well. To evaluate our re-sampling method, we use the measure for imbalance from [21]. The imbalance ratio per label is defined as below:

$$IRL(y_k) = \frac{\arg\max_y \sum_{i=1}^{d} h(y, y_i)}{\sum_{i=1}^{d} h(y_k, y_i)} \tag{3.3}$$



(a) before balancing

(b) after balancing

Figure 3.5: The chord diagram for the relationship between the tools after balancing with LP method

where $h(y, y_i) = 1$ if $y \in y_i$. Using this metric, the most frequent class has value 1 and the rest of the classes have value greater than 1. The mean and variance of ILR can be used as other measures for comparing the imbalance level of different datasets. The balancing scores of the training set, before and after LP-based balancing method is shown in Table 3.2. It can be seen that the balance of the dataset has dramatically improved, after applying the under-sampling method.

3.2.5 Multi-task Training

Since our tool detector $F(x)$ is decomposed into an ML classifier f and an MC decision model g, the requirement of both models needs to be considered during training. To accomplish this, the model is trained using both ML and MC loss functions.

Having the vector of the confidence scores P, the ML loss L_f is the sigmoid cross-entropy and is formulated as:

Table 3.2: Balancing scores for each tool in M2CAI dataset before and after balancing

Tool	Before	After
Bipolar	22.42	4
Clipper	16.09	4
Grasper	1.36	1
Hook	1	4
Irrigator	14.82	2.6
Scissors	34.38	4
Specimen Bag	9.34	2.6
Mean	14.20	3.17
Variance	120.0	1.64

$$L_f = -\frac{1}{d}\sum_{x\in\mathcal{D}} \log(p_{k=Y}), \qquad P = (\sigma f(x)) \tag{3.4}$$

where Y is the correct label for frame x, d is the total number of frames and $\mathcal{D}$ is the training set.

The Softmax CE loss function L_g for the decision model is formulated as:

$$L_g = -\frac{1}{d}\sum_{x\in\mathcal{D}} \log(q_{k=\hat{Y}}), \qquad Q = (softmax(g(f(x)))) \tag{3.5}$$

We propose to use the joint training paradigm for optimizing the ML and MC losses as a multitask learning approach. In order to do that, two optimizers are defined based on the two losses with separate hyper-parameters such as learning rate and trainable weights. Using Stochastic Gradient Descent (SGD), the weights update at iteration l can be written is:

$$\theta^{(l)} = \theta^{(l-1)} - \sum_{x\in\mathcal{D}_b} [\eta_f \nabla_\theta L_f(\theta^{(l-1)}) + \beta.\eta_g \nabla_\theta L_g(\theta^{(l-1)})] \tag{3.6}$$

where β is a constant weight for adjusting the impact of the two loss functions, $\mathcal{D}_b$ is a randomly selected batch, and η_f and η_g are the learning rates for the ML and MC

loss functions respectively. The training is performed in an end-to-end fashion, and the gradients ($\nabla_\theta L$) are calculated using the backpropagation through time (BPTT) method.

The trainable weights for the ML optimizer are all the weights in the CNN, the weights in the RNN, which are U and W matrices in (3.1), and $FC1$, whereas for the MC optimizer, the CNN, RNN, and $FC2$ are trainable. Note that the shared weights between the two optimizers are the RCNN weights. By keeping the $FC1$ layer untouched by the MC optimizer, the spatiotemporal features are extracted by the RCNN considering both the presence of each tool and the combination of them, and $FC2$ is solely trained as a decision model.

3.2.6 Post-processing

The final decision of the RCNN model from the previous section is made based on the extracted spatiotemporal features from a short sequence of frames. In other words, the model benefits from a short-term memory using the correlation among the neighboring frames. However, due to the high under-sampling rate for the balanced training set, this method might not produce a smooth prediction over the entire duration of the laparoscopic videos. To deal with this issue, we model the order in the usage of the tools with an RNN over all the frames of each video [64].

Due to memory constraints, the final prediction from equation 3.2 of the RCNN, $\bar{C}(j) = [c_{0j}....c_{mj}]]$ for all the videos $0 < j < n$, is selected as the input for the post-processing RNN. Since not all the videos have the same length, the shorter videos are padded with the no-tool class.

Our post-processing occurs in both online and offline modes. In online mode, only the past frames are available for classifying the current frame. Therefore, the RNN in online mode is a uni-directional model. To capture the long-term dependencies, we designed a multi-layer RNN.

In offline mode, future frames can also be used along with past frames to improve the classification results of the current frame. In order to accomplish this, a bi-directional RNN is employed, which consists of two RNNs for the forward and backward sequences. The backward sequence is simply the reverse of $\bar{C}$. The outputs of the last layer of the bi-RNN are concatenated and fed to $FC3$ for the final prediction (g in Fig. 3.2.c).

Since the input frames for the bi-RNN are in a specific order, it's not possible to balance the input through re-sampling. Therefore, class re-weighting is performed to compensate for the minority classes. The class weights are chosen to be proportional to the inverse of the frequency of the superclasses in the training set. The loss function is:

$$L_p = -\frac{1}{d}\sum_{x\in\mathcal{D}}(w_{k=\hat{Y}})\log\bar{q}_{k=\hat{Y}}, \quad \bar{Q} = (softmax(g\prime(\bar{C}))) \tag{3.7}$$

where w_k is the weight for the superclass k, g is a two layer GRU with 128 and 32 units in each layer, and $\bar{Q} = (\bar{q}_1, ..., \bar{q}_{\hat{K}})$ is the superclass probability vector.

The method described in this section is similar to [4] in extracting the long-term temporal features using RNNs. However, in contrast to [4], we used the final predictions of the RCNN model instead of the vector of confidence scores of the tools. Besides containing the information about the co-occurrences, training RNNs can be accomplished easier with a single scalar versus the vector of the size of the total number of tools or the tools' combinations. With the aid of the shorter size input, we were able to train larger sequences, even after performing the temporal data augmentation (to be explained later), using shallower and with fewer units RNNs in online and offline modes.

3.3 Experiments and Results

In this section, the performance of the different parts of the proposed tool detection model on M2CAI dataset is validated through numerous experiments using the appropriate metrics.

The CNN architecture used in all of the experiments is called Inception V1 [98]. In order to have a better generalization, extensive data augmentation such as random cropping, random horizontal and vertical flipping, random rotation and a random change in brightness, contrast, saturation, and hue were used during training. Unless stated otherwise, the initial learning rate was 0.001 with a decay rate of 0.7 after five epochs, and the results are taken after 100 epochs. The batch size was 32 for training the CNN models and 40 for RNN based models.

3.3.1 Metrics

Since the proposed model is MC, the corresponding evaluation metrics were chosen. Due to the high imbalance of the validation dataset, accuracy alone is not sufficient to evaluate the proposed model. Therefore, we used the F1 score to compare the performance of different models in both per-class and overall metrics. These are calculated as:

$$F1_{micro} = 2\frac{P_{pc}.R_{pc}}{P_{pc} + R_{pc}}, F1_{macro} = 2\frac{P_{ov}.R_{ov}}{P_{ov} + R_{ov}} \tag{3.8}$$

where P_{pc}, R_{pc}, P_{ov} and R_{ov} are per-class precision/recall and overall precision/recall respectively and are calculated as:

$$P_{pc} = \frac{1}{K}\sum_{k=1}^{K}\frac{N^c_{y_k}}{N^p_{y_k}}, R_{pc} = \frac{1}{K}\sum_{k=1}^{K}\frac{N^c_{y_k}}{N_{y_k}} \tag{3.9}$$

$$P_{ov} = \frac{\sum_{k=1}^{K} N^c_{y_k}}{\sum_{k=1}^{K} N^p_{y_k}}, R_{ov} = \frac{\sum_{k=1}^{K} N^c_{y_k}}{\sum_{k=1}^{K} N_{y_k}} \tag{3.10}$$

Table 3.3: Results for the multi-label classification of the CNN

Total Frames	Balanced	Acc(%)	mAP(%)	F1-macro(%)
23k	No	77.23	61.02	58.48
150k	Yes	75.90	71.15	70.49
75k	Yes	74.78	77.24	74.81
25k	Yes	75.40	78.58	74.64
6k	Yes	74.36	78.22	74.43
3k	Yes	73.10	73.69	70.85

where $N^c_{y_k}$, $N^p_{y_k}$ and N_{y_k} are the number of correctly predicted frames for class k, the total number of frames predicted as k, and the total number frames for class k. Only frames with all the tools predicted correctly are considered exact matches.

To evaluate the RCNN model f, we used ML metric - mean Average Precision (mAP), which is the mean of average precision (a weighted average of the precision with the recall at different thresholds) for all seven tools.

3.3.2 CNN Results

We first test the performance of the model when only the CNN is used, since CNN is at the core of the proposed tool detector. In other words, we assume that the classifier f is a CNN, and the decision model g is applied to the resulting score from the CNN.

Table 3.3 shows the results of our CNN with different training set sizes for all the tools listed in Table 3.1.

Since the dataset was labeled only for one frame per second (out of 25 frames/sec), there was a possibility of using the unlabeled frames for training, as long as the tools remain the same between two consecutive labeled frames. We used this unlabeled data to balance the training set, according to the LPs. The CNN was trained with the loss function 3.4 with $FC1$ of size 7. Table 3.3 shows the results of our CNN with different training set sizes for the tools listed in Table 3.1.

As was to be expected, the unbalanced training set results shown in the first row of Table 3.3 has the lowest performance on all the metrics. The high exact match accuracy (Acc) of 77.23% and the lower results on per-class metrics, such as F1-macro and mAP show that the model correctly predicted the majority classes (grasper and hook) but has poor performance for the less used tools such as scissors.

In order to balance the datasets, the following steps were taken:

1) 15 superclasses were selected, and the original frames were re-sampled to have a uniform distribution in the set of label-sets $\hat{\mathcal{Y}}$. The numbers of frames for each superclass were randomly selected to be 10,000, 5,000, 1,666, 400, and 200.

2) Multiple copies of some frames were copied and pasted to the final set in the first two training sets, because of the availability of fewer frames in some tools such as scissors. This accomplished the intended over-sampling.

3) Similarly, under-sampling was performed in at least one class in all sets and, in all classes in the last two sets, because too many frames were available for some tools.

Under these conditions, we can discuss the results presented in rows 2 through 6 in Table 3.3. While the exact match accuracy is the highest in the 150K set, it has the lowest score on the per-class metrics. The likely reason is the high over-sampling rate, which causes overfitting for the less frequently occurring classes. On the other hand, a very high under-sampling rate in the 3K set results in lower accuracy, likely due to the lack of informative samples.

The best per-class results are for the 25K/6K versus the 150K/75K, which is due to the lower correlation among the inputs of the CNN during training. We used the 6K dataset for the rest of the experiments versus the 25K, because adding the RNN and decision model to the selected CNN would increase the size of the model (RCNN-LP), and the chances of overfitting increases.

To evaluate the effect of utilizing the co-occurrence of different surgical tools, we tested the LP method as the primary classifier, as well as the decision model, using different training strategies. The configurations for each experiment are shown in Table 3.4.

Table 3.4: Setup configurations for training the multiclass CNN

Experiment number	Loss function	FC1 size	FC2 size	Trainable weights	Training method
1	(3.5)	15	-	all	-
2 (150k)	(3.5)	15	-	all	-
3	(3.4)/(3.5)	15	15	CNN+FC1/ FC2	Sequential
4	(3.4)/(3.5)	7	15	CNN+FC1/ FC2	Sequential
5	(3.4)/(3.5)	7	15	CNN+FC1/ all	Alternate
6	(3.4)/(3.5)	7	15	CNN+FC1/ CNN+FC2	Alternate
7	(3.4)/(3.5)	7	15	CNN+FC1/ CNN+FC2	Joint

In sequential training, the CNN was trained first, and the decision model was added on top of the trained model, while the CNN weights remained unchanged. In alternate training, the trainable weights change with the loss at every other step. The joint training method is explained in the previous section. We used MC metrics; exact match accuracy, micro and macro F1, and average per-class precision and recall. The results are shown in Table 3.5, and the precision and recall for each tool are shown in Table 3.6.

In the first two experiments, the LP method was used directly to map the video frames to the corresponding superclass. In order to accomplish this, the features extracted using CNN were connected to an FC layer of size 15 and the network was trained with the loss function from equation 3.5. We selected the balanced training sets from the previous experiments with 6K and 150K samples. It can be seen from experiment 1 and 2 (Table 3.5),

Table 3.5: Results for the multiclass CNNs

Exp. num	Acc(%)	F1-macro(%)	F1-micro(%)	Mean P(%)	Mean R(%)
1	70.01	69.14	84.57	72.90	67.98
2	76.13	73.77	87.89	86.08	67.24
3	73.18	74.30	86.92	79.65	70.80
4	74.97	75.47	88.04	80.67	73.21
5	72.44	75.23	86.42	87.60	67.25
6	74.42	75.70	87.75	82.37	71.48
7	76.31	78.32	88.53	78.48	78.95

which correspond to 6K sample and 150k samples respectively, that both accuracy and F1 scores increase, when the training set is larger. Also, the precision and recall in Table 3.6 show some improvements in almost all classes. However, compared with the results from Table 3.3, we observe minor improvements in accuracy and F1, when using 150k frames with LP classifier, while the metrics decrease with a smaller training set. Considering both training sets were balanced based on LP, the observation suggests that the LP-based classifier needs more examples for reasonable performance. This is because, in an LP classifier, the superclasses are treated as separate classes with different features from the corresponding single label classes, which requires the classifier to have more training examples to learn the discriminating features. This can also be confirmed by checking the relatively close precision/recall for grasper and hook in Table 3.6, which have more unique frames (due to lower under-sampling rate), in the two experiments.

In experiment 3, ML loss was tried instead of MC for training the LP classifier with 15 superclasses. $FC2$ was added as a decision model and was trained sequentially. As shown in Table 3.5, the per-class F1 score for experiment 3 improves compared to experiment 1, while the exact match accuracy is lower. This is probably because the model is still not aware that a superclass is a combination of multiple classes.

In experiments 4, 5, and 6, the CNN was trained using ML loss 3.4 with seven classes, and the decision layer was added on top of the confidence scores. We evaluated the

Table 3.6: Precision (P) and Recall (R) of each tool for the multiclass CNNs

	Bipolar		Clipper		Grasper		Hook		Irrigator		Scissors		Specimen bag	
Exp. num	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)
1	71.25	66.05	72.72	58.41	90.30	70.90	92.82	90.62	56.73	65.57	61.90	32.91	64.60	91.45
2	76.56	35.76	85.41	52.06	92.01	80.30	95.25	90.04	85.84	74.59	93.82	48.10	73.67	89.83
3	84.48	71.53	78.01	57.46	91.02	75.80	94.63	90.97	57.01	53.27	79.62	54.43	72.81	92.14
4	75.18	75.18	69.25	59.36	90.04	81.54	95.2	90.35	72.88	70.49	87.83	41.11	74.36	94.45
5	91.71	56.56	81.25	53.65	91.32	75.97	97.59	86.19	78.26	59.01	93.18	51.89	79.95	87.52
6	81.51	70.80	80.76	60.01	89.96	79.82	95.31	90.15	63.30	56.55	88.76	50.01	77.05	93.07
7	83.96	72.62	72.67	74.28	91.14	81.25	94.36	91.47	59.74	75.4	71.63	63.92	75.88	93.76

model using different training strategies. All three of these experiments produced better results than experiment 3. This is likely because the model can learn the pattern of the seven tools easier with the ML loss, compared with learning the pattern for the combination classes using 15 classes.

The point of performing the experiments 5 and 6 was to evaluate the effect of the decision model in training the feature extractor and ML classifier. In both experiments, the decision model g, and the CNN were trained alternately. The weights of the CNN were frozen in experiment 4, while in experiments 5 and 6 they were trained at each step. Therefore, in experiment 4, the role of the decision model was to just use the co-occurrence information to find the correct classes (superclasses) using the confidence scores of a trained model. The results show improvement in F1 scores in all three experiments compared with the results from Table 3.3. This is because using LP as the decision model, the co-occurrence of surgical tools in each frame is considered directly in the classification method without learning separate patterns for superclasses.

The difference between experiment 5 and 6 is that is experiment 6, $FC1$ is only trained with the ML loss function, whereas in experiment 5, both ML and MC loss function can affect $FC1$. The results show that the training strategy in experiment 6 is better in F1 and accuracy measures. The reason is that in experiment 6, $FC1$ is only trained for mapping the extracted visual features of the CNN to the confidence scores of the tools.

In experiment 7, the training is accomplished by the weighted sum of the ML and MC training operations (equation 3.6, and the trainable weights are the same as in experiment 6. We can see that the end-to-end training of the CNN-LP produces significantly better results compared with all other training methods, such as sequential and alternate training. The reason is that in end-to-end training, all parts of the model is trained simultaneously to reach better confidence scores and hence, better final decision.

3.3.3 LapTool-Net Results

In this section, the performance of the proposed model is evaluated after considering the spatiotemporal features using an RNN. Similar to the previous section, we tested the model before and after adding the decision model. The dataset for training is the 6K balanced set, and all the models were trained end-to-end. For training the RCNN model, we used five frames at a time (current frame and four previous frames) with an inter-frame interval of 5, which resulted in a total distance of 20 frames between the first and the last frame. The RCNN model was trained with a Stochastic Gradient Descent (SGD) optimizer. The data augmentation for the post-processing model includes adding random noise to the input and randomly dropping frames to change the duration of the sequences; the final predictions of the RCNN model are saved every 20 frames, and the frames are dropped with the probability of 10 to 30%.

Table 3.7 shows the results of the proposed RCNN and LapTool-Net. For ease of comparison, we have copied the results from the previous section for the CNN with and without the LP decision model.

Table 3.7: Final results for the proposed model

	Acc(%)	F1-macro(%)	F1-micro(%)
CNN	74.36	74.43	87.70
CNN-LP	76.31	78.32	88.53
RCNN	77.51	81.95	89.54
RCNN-LP	78.58	84.89	89.79
LapTool-Net(online)	80.95	88.29	91.24
LapTool-Net(offline)	81.84	90.53	91.77

It can be seen that by considering the temporal features through the RCNN model the exact match accuracy and F1-macro were improved by 3.15% and 7.52% respectively. Also, the F1-macro improves by 2.94% after adding the LP decision model. The superiority

of the LP-based decision model over the threshold method is due to the consideration of the co-occurrence patterns and the inclusion of the spatiotemporal feature extraction in the decision model by the end-to-end training method.

To check the effectiveness of the multi-task approach used for the end-to-end training of the RCNN-LP model, we took the output of the ML classifier, after removing the decision model from the trained RCNN-LP. In other words, we replaced the LP-based decision layer of the trained model with the threshold-based decision method. The results are shown in Table 3.8.

Table 3.8: The precision, recall and F1 score of each tool for the ML classifier in RCNN-LP after removing the decision model

Tool	Precision(%)	Recall(%)	F1(%)
Bipolar	77.62	83.57	80.49
Clipper	83.22	81.90	82.56
Grasper	69.99	90.28	78.85
Hook	95.33	93.43	94.37
Irrigator	77.27	83.60	80.31
Scissors	82.91	82.91	82.91
Specimen bag	76.96	94.91	85.00
Mean	80.55	87.22	83.50

Compared with the results for the RCNN model in Table 3.7, we can see 1.55% improvement in F1-macro after training with the proposed method. The reason is that with the help of the joint training strategy, the presence of the tools is detected based on the pattern of the tool combinations and therefore, richer extracted features. The precision for the Grasper is an indicator of the remaining balance in the training set. We believe the could have been better by re-weighting the loss function, especially for Grasper.

The higher performance of the LapTool-Net, shown in Table 3.7 is due to consideration of the long-term order of the usage of the tools. In offline mode, the utilization

of the frames from both the past and the future of the current frame is considered by a bi-directional RNN, and therefore, we can see the improvements over the online model on accuracy and F1 scores.

The precision, recall, and F1 score for each of the tools are shown in Table 3.9 for offline mode and in Table 3.10 for online mode. Compared with the results from Table 3.6, we can see that the F1 score for clippers and scissors have significantly increased because there is a high correlation between the usage of these tools and the tasks, i.e., the order in the occurrence of the tools (e.g., cutting only happens after clipping is completed). The lowest performance is for Irrigator, which is probably because of the irregular pattern in its use (only used for bleeding and coagulation, which can occur any time during the surgery). The higher over-all recall is likely because of the class re-weighting method. We believe the performance could improve with a better choice of the weights.

Table 3.9: The precision, recall and F1 score of each tool for LapTool-Net(offline)

Tool	Precision	Recall	F1
Bipolar	0.86	0.95	0.90
Clipper	0.95	1	0.97
Grasper	0.89	0.90	0.90
Hook	0.93	0.94	0.94
Irrigator	0.72	0.94	0.82
Scissors	0.84	0.99	0.91
Specimen bag	0.88	0.92	0.90
Mean	0.82	0.94	0.90

To show the performance of the model on rare combinations, we performed an experiment on the smaller dataset containing only the rare classes, which are the last 3 combinations in figure 3.4. These classes are [Bipolar, Grasper, Specimen bag], [Irrigator, Specimen bag] and [Bipolar and Specimen bag]. The results are shown in table 3.11.

Table 3.10: The precision, recall and F1 score of each tool for LapTool-Net(online)

Tool	Precision	Recall	F1
Bipolar	0.70	0.89	0.79
Clipper	0.90	0.99	0.94
Grasper	0.90	0.88	0.89
Hook	0.94	0.94	0.94
Irrigator	0.80	0.86	0.83
Scissors	0.81	0.97	0.88
Specimen bag	0.91	0.92	0.91
Mean	0.85	0.92	0.88

Table 3.11: The precision, recall and F1 score of each tool for the dataset of rare combinations

Tool	Precision(%)	Recall(%)	F1(%)
Bipolar	95.23	68.96	80
Grasper	40.9	100	58.06
Irrigator	100	82.14	90.19
Specimen bag	100	85.45	92.15
Mean	84.03	84.13	80.1

The results show that the RCNN-LP classifier can detect the rare classes even when it is not trained on them.

3.3.4 Model Ensemble

Neural networks are highly non-linear models and therefore, have a high variance in their predictions. Ensemble learning methods are a group of techniques to deal with such high variance by combining multiple models. This is achieved by varying the model and/or the training data and using a combination of weaker models to form a stronger and more reliable classifier.

We implemented an ensemble learning method by training the proposed RCNN-LP model on different training sets, which are randomly selected using the method described

earlier. In order to do that, for each model, the final probability vector Q is saved, and the new prediction vector is calculated by averaging the results.

In this experiment, five datasets were created and the performance is calculated by combining 3, 4, and all of them. Different combinations were tested for the first two cases (3 and 4) and the best results are shown in Table 3.12.

Table 3.12: The results for different ensembles of RCNN-LP model

Ensembles	Acc(%)	F1-macro(%)	F1-micro(%)
5	79.93	85.10	90.84
4	79.86	85.55	90.85
3	79.98	86.25	90.91

It can be seen that, compared with the results for single model RCNN-LP from Table 3.7, the ensemble of multiple models improved the performance in all cases. The better results for smaller ensembles is because the weaker classifiers were deleted and the results were taken from the best models.

3.3.5 Tools Localization

In order to localize the predicted tools, the attention maps were visualized using gradient weighted class activation map or grad-CAM method [87]. Using grad-CAM, all the channels of the final activation map (the last convolutional layer) are weighted by the gradient of a class of interest with respect to the channel, to produce a coarse heat map of the important parts of the image in predicting the class.

The results for some of the frames are shown in figure 3.6. To avoid confusion for frames with multiple tools, only the activation map of a single tool is shown based on the prediction of the model. The results show that the visualization of the attention of the proposed model can also be used in reliably identifying the location of each tool.

Table 3.13: Comparison of tool presence detection methods on M2CAI

Method	CNN	mAP(%)	F1-Macro(%)
LapTool-Net(offline	**Inception-V1**	-	**90.53**
LapTool-Net(online)	**Inception-V1**	-	**88.29**
RCNN (ours)	**Inception-V1**	**87.88**	**81.95**
[38]	Resnet-101 [36]	86.9	-
[42]	VGG	81.8	-
[79]	Alexnet	65	-
[110]	Inception-V3 [99]	63.8	-
[78]	Alexnet	61.5	-
[103]	Alexnet	52.5	-

3.3.6 Comparison

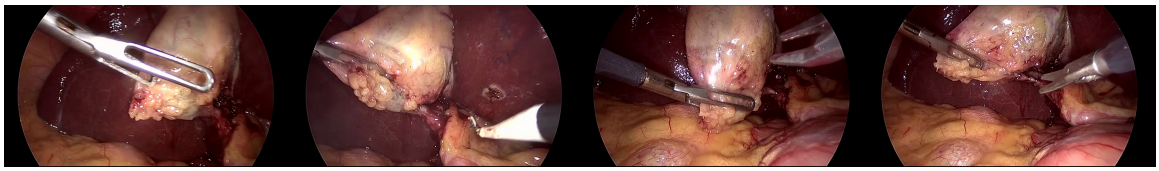
In order to validate the proposed model, we compared it with previously published research on the M2CAI dataset. The result is shown in Table 3.13. Since all the methods reported their results using ML metrics such as mAP, we compared our ML classifier f, which is the RCNN model, along with the final model. We show that our model out-performed previous methods by a significant margin even when choosing a relatively shallower model (Inception V1) and while using less than 25% of the labeled images.

3.4 Conclusions

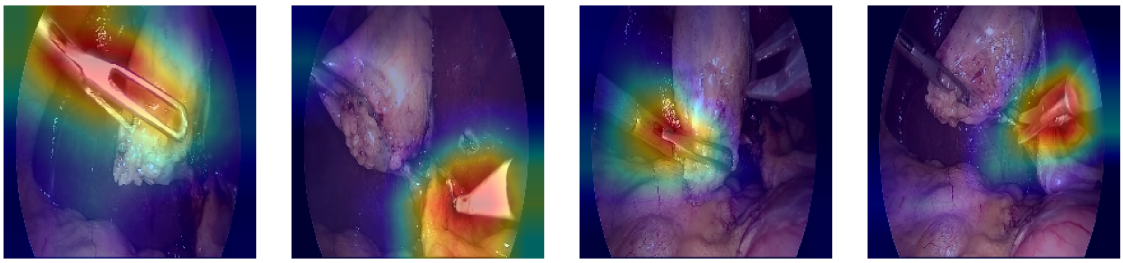
The observation by surgical residents of the usage of specific surgical instruments and the duration of their usage in laparoscopic procedures gives great insight into how the surgery is performed. While identifying the tools in a recorded video of surgery is a trivial albeit tedious task for an average human, there are certain challenges in detecting the tools using computer vision algorithms. To tackle these challenges, in this paper, we proposed a novel deep learning system called LapTool-Net, for automatically detecting the presence of surgical tools in every frame of a laparoscopic video. The main feature of the proposed RCNN model is the context-awareness, i.e. the model learns the short-term and

long-term patterns of the usages of the tools by utilizing the correlation between the usage of the tools with each other and, with the surgical steps, which follow a specific order. To achieve this goal, an LP-based model is used as a decision layer for the ML classifier, and the training is performed in an end-to-end fashion. The advantage of this paradigm over direct LP classifier is that the training can be accomplished with a smaller dataset, due to having fewer classes and avoiding learning separate (and probably not useful) patterns for the superclasses. Furthermore, the order of occurrence of the tools is extracted through training a bi-RNN with the final prediction of a trained RCNN model. To overcome the high imbalance in the occurrence of the tools, we used under-sampling based on the tools combinations and the LP model. In addition to having a balanced dataset, the high under-sampling rate reduces the generalization error by avoiding overfitting, which is the main challenge in tool detection, due to the high correlation among videos frames. Our method outperformed all previously published results on M2CAI dataset while using less than 1% of the total frames in the training set.

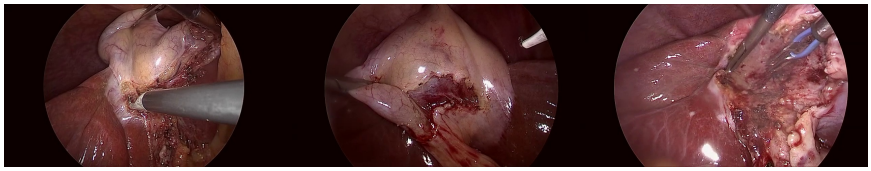
While our model is designed based on the previous knowledge of the cholecystectomy procedure, it doesn't require any domain-specific knowledge from experts and can be effectively applied to any video captured from laparoscopic or even other forms of surgeries. Also, the relatively small dataset after under-sampling suggests that the labeling process can be accomplished faster by using fewer frames (e.g., one frame every 5 seconds). Moreover, the simple architecture of the proposed LP-based classifier makes it easy to use it with other proposed models such as [4] and [38], or with weakly supervised models [65, 107]to localize the tools in the frames. Moreover, the offline design can be useful in generating a summary report, assessment and procedure rating etc. Also, the proposed model in online mode has a processing time of less than 0.01 seconds/frame, which makes it suitable for real-time applications such as feedback generation during surgery.



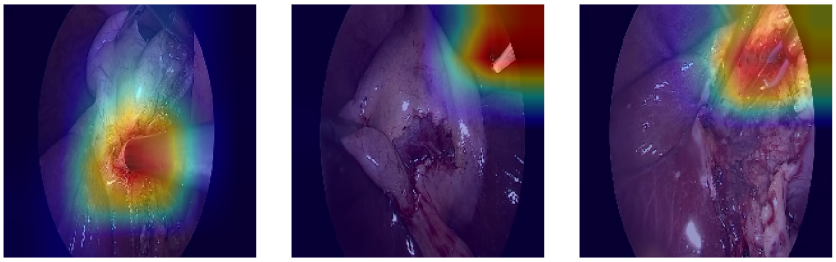
(a) Grasper (b) Grasper/Hook (c) Grasper/Clipper (d) Grasper/Scissor



(e) Grasper (f) Grasper/Hook (g) Grasper (h) Grasper/Clipper



(i) Grasper/Hook (j) Grasper/Hook (k) Grasper/Bipolar



(l) Grasper/Hook (m) Grasper/Hook (n) Grasper/Bipolar

Figure 3.6: The visualization of the class activation maps for some example based on the prediction of the model

CHAPTER 4

SURGICAL WORKFLOW DETECTION

Routine assessment of open surgical skills requires expert supervision in the operating room (OR) during surgery. Alternatively, in laparoscopic procedures, evaluation can be accomplished using a video recorded from the operation, which can be multiple hours long [119]. However, this requires a significant amount of the surgeon's time and can be subject to human error. Automatically detecting and separating the videos into the corresponding surgical phases can help the surgeons quickly find and evaluate the videos. However, the most productive evaluation approach results if the segmented parts can be extracted in an automated system to perform the evaluation using objective measures. The final output videos will then have all the phases identified, enabling easy access to the information stored in video databases during the training of the surgical residents.

In this chapter, we study the problem of analyzing the surgical workflow in a laparoscopic video by proposing two different approaches. In the first method, which is described in section 4.1, the detection of surgical phases is accomplished by classifying all of the frames of a video. In section 4.2, a novel model is designed for finding the borders of each phase.

4.1 Frame-level Phase Detection

Automatic detection of surgical steps using still frames from the videos is quite challenging as the rapid movement of the camera causes blurry views and low-quality frames. Also, the camera is not always focused on the scene. This is shown in figure 4.1.a. Thus, using still frames might not be sufficient for distinguishing the surgical phases.

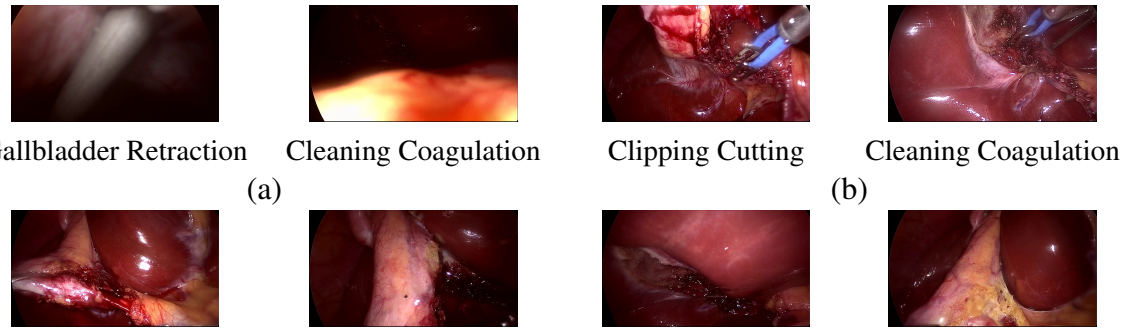


Figure 4.1: Illustration of the challenges in the detection of the surgical work flow using still images. a) examples of blurry images and the lack of focus in camera, b) Using the same bipolar tool in two different phases, c) the absence of surgical instruments in different steps

Traditionally, identifying various surgical phases was accomplished using the signals from the surgical tools that are used during the procedure [66] or from the manually annotated videos with handwritten notes such as ”tools”, ”anatomical structures”, and ”surgical tasks” [30]. However, such annotation is very time-consuming. These shortcomings have been recently addressed after the introduction of deep learning systems and CNNs.

Endonet [104] was the first that used visual features from a CNN architecture called Alexnet [51]]. In their work, the CNN was trained for extracting multi-level features, for simultaneously identifying surgical instruments and surgical phases. The output of the tool detection system was concatenated with the features, and the detection of phases were accomplished using the fused features. The result of the last layer of the CNN was then used as the input to a Hierarchical Hidden Markov Model (HHMM) to incorporate the temporal features into the phase detection model. Following their work, in [78], a Gaussian fitting model was used to deal with the correlation among the video frames. They further used a random forest classifier to improve the classification accuracy.

In their other work, Twinanda et al. [103] replaced the HMM model with an LSTM model. Although using tool information showed some improvement in some of the mea-

sures that they used for evaluation, the accuracy remained the same when using LSTM. The reason is that not all of the surgical phases correlate with the tools that are used. As some examples of the tool-phase discrepancies, figure 4.1b illustrates two frames with the same tool but different phases, and in figure 4.1c no tool is present in four distinct phases. We intentionally didn't use the tool information, because we wanted to have a fair comparison with the previous works, which use only the phase information and, the labels for the instruments are not always available.

In order to train a CNN with LSTM in an end-to-end fashion, EndoRCN [43] employed three consecutive frames as the input sequence to an RNN. Similarly, SVRCN [44] used a small number of successive frames for simultaneously training a CNN and an LSTM. Although this model has the benefits of simultaneous end-to-end training of both CNN and LSTM and, the smoother results due to the small number of frames, it misses the long-term relationship between the video phases. In order to tackle this issue, they introduced a probabilistic method called prior knowledge inference (PKI) during test time, which took the probability of the previous frames as input and updated the predictions according to the previously known ordering of the surgical phases in the videos. Using this technique, they reached state-of-the-art results in cholecystectomy datasets. One drawback of this method was that it ignored the future frames, and it required prior knowledge regarding the ordering of the surgical workflow to get the best performance.

In our method called Surgical Phase Detection using a Deep Learning System (SPD-DLS), we designed two separate architectures for real-time and offline modes. In both architectures, in addition to the visual features, the frame number was added as a separate feature, before the fully connected layer of the CNN. This method helped the CNN know the actual position of the frame with respect to the other frames of the video. In the offline model, a temporal median filter was applied to a short window of frames. The outputs of both architectures were sent to a separate LSTM network for the final decision making on

the phase detection. The proposed method was tested with a dataset of cholecystectomy procedure. The test showed significant improvement in detecting surgical workflow over current methods. The main contributions of the chapter are as follows:

- Having separate architectures for online and offline detection of surgical phases
- Using short-term temporal features employing a median filtering method
- Taking long-term correlations of features in the entire video into account using LSTM
- Using frame number (time) along with the other high-level visual features to improve the phase detection accuracy
- Addressing the class imbalance problem of surgical phases
- Dealing with the over-fitting issue by separating the training sets of the CNN and LSTM models

The remainder of the chapter is organized as follows: in section 4.1.1 we describe the methodology in detail. In section 4.1.2 we describe the setting used and in section 4.1.3, we evaluate the proposed model and compare the results with those of the recent methods. In the final section, we list the conclusions and some suggestions for future work.

4.1.1 Methodology of SPD

In this section, we describe the method designed for segmenting surgical videos according to their procedural phases. The proposed method consists of two architectures for online and offline cases. Both architectures rely on a CNN for extracting spatial features in each frame of the videos and an RNN for temporal dependencies of the subsequent frames.

4.1.1.1 Visual Features

Extracting high-level features of images is at the heart of most computer vision tasks. Since phase detection is a multi-class problem, we use a softmax cross-entropy loss func-

Table 4.1: Duration of each phase in cholecystectomy procedure

	Phase	duration(sec)
P0	Preparation	125 ± 95
P1	Calot Triangle Dissection	954 ± 538
P2	Clipping and Cutting	168 ± 162
P3	Gallbladder Dissection	857 ± 551
P4	Gallbladder Packing	98 ± 53
P5	Gallbladder Retraction	83 ± 56
P6	Cleaning and Coagulation	178 ± 166

tion to find the probability of each class. The class with the maximum probability is selected as the CNN prediction of the current phase, for sending to the RNN to deal with the temporal correlations of the frames.

4.1.1.2 Imbalance Compensation

As can be seen in table 4.1 [104], the duration of each of the phases of cholecystectomy procedure varies significantly in each video. In order to address this phase count imbalance problem, we used an up-sampling method, which interpolates the video frames for less frequent phases. The resulting dataset has a uniform distribution among all phases.

4.1.1.3 Over-fitting

Recent CNN architecture such as residual networks [36] and Inception [97, 98] are designed to classify a large number of images into 1000 classes. Each model consists of tens of layers, each with thousands of parameters to train. Due to the large number of parameters, these architectures are prone to the well known and significant over-fitting error resulting in lower generalization.

Although surgical videos are recorded at high frame rates (25 or 30 per second), the frames are too highly correlated for high generalization in a deep CNN. In other words, neighboring frames share most of the important visual features used for classification of

these frames. However, even in our case, the set of frames used for training are entirely unlike the set of test frames. Therefore, perfect accuracy during training and potentially large error during testing could occur. While the use of various regularization methods such as l1 and l2, dropout and data augmentation reduces the generalization error, there is still a big gap between training and test accuracies. In order to deal with this issue, we split the training set into two smaller sets for separately training CNN and LSTM. The CNN model was first trained on the CNN training set and the trained model was then applied to a separate set to get the predictions for the input to the RNN model. This ensured that the input sequence used for training the RNN has the same error pattern as the sequence at the test. This way we worked around letting over-fitting affect the results and helped the RNN learn the input sequence better.

4.1.1.4 Temporal Correlations

CNN's are suitable for extracting spatial visual information only from still images. As can be seen in figure 4.1, because of the rapid camera or surgical instrument movements, not all of the images are clear enough to be used with the CNN. We dealt with this problem during test time, by median filtering of the predicted phases. However, due to the high correlation of surgical phases over neighboring frames in a video, an error can be propagated to several frames in a row. Thus, using long-term temporal dependencies is crucial in frame-level classification in a video. To exploit such dependencies, we used a Long Short-Term Memory (LSTM) model [37], which is a type of recurrent neural network suitable for long sequences. LSTMs have been used for time-series modelings such as sequence classification and sequence labeling. To utilize the full potential of an LSTM model, we applied it to the entire video so that the model can learn the order of the surgical phases by "seeing" the full sequence. However, due to memory constraints, it's not possible to train both an RNN and a CNN in an end-to-end fashion.

For the LSTM input, one approach was to use the extracted features from the last convolutional layer of the trained CNN. While these features have a good representation of the visual features, it is somewhat complicated to train an RNN with our large input size. Moreover, the number of features in the last layer varies for each architecture. Alternatively, we could use the outputs of the CNN, which are the probabilities of all classes, or the final predictions. We used the prediction for both online and offline models, which enabled us to have a shallower LSTM without sacrificing accuracy.

4.1.1.5 Time-stamping

Since, often the duration of the phases and their order are known, we can use the frame number to improve the phase predictions of the CNN. We used the frame number or the time of each frame as an additional feature in the last layer of our convolutional network. Thereafter, We used the fused features (the combination of the frame number and the features from the last layer) with the fully connected and softmax layers. This way, the CNN is aware of the relative position of each frame during training.

4.1.1.6 Offline architecture

During the offline mode, we assumed that the model has access to all the frames before and after the current one. Also, the total number of frames in a video is known. For training the CNN, the frame number was divided by the total size of the video (normalized) and was concatenated with the features of the last convolutional layer and right before the fully connected layer. A median filter then determined the output of a separate training set (the final prediction of the CNN). The results were used for training a bi-directional LSTM model. A Bi-LSTM consists of two LSTMs, one in the forward direction and the other

in the backward direction, which enabled us to consider both the previous and the future frames. A block diagram of the method is shown in figure 4.2.

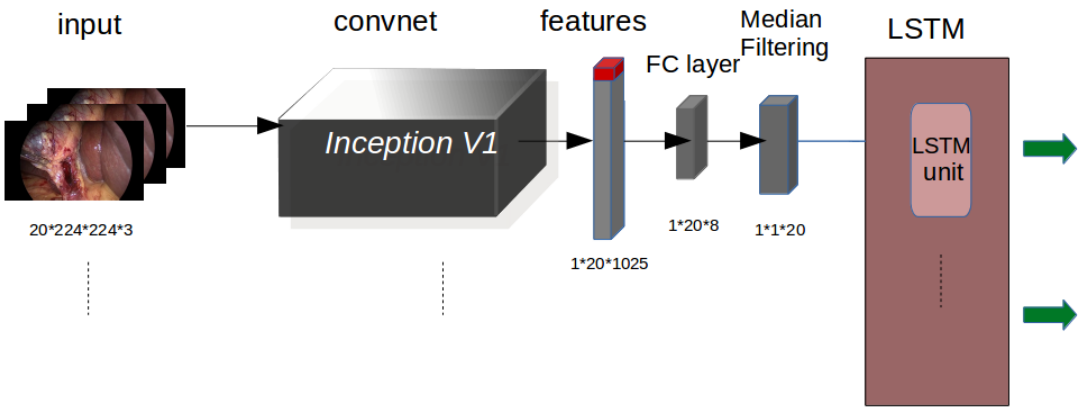


Figure 4.2: The block diagram of the Surgical phase detector (SPD) in offline mode.

4.1.1.7 Online architecture

Unlike the offline mode, in the online detection of surgical workflow, the only information available was the features from the previous frames. Therefore, a uni-directional LSTM was applied to the output of the CNN. On the other hand, since the temporal smoothing method needed the frames information from the past and the future, it could not be used in the online mode. Since the size of the video was unknown at the processing time, the time-stamp feature was just the frame number (unnormalized) concatenated with the feature map of the convolutional model.

4.1.2 Experimental Setup

We used the largest publicly available cholec80 dataset [104] to evaluate our method. The dataset contained 80 videos from cholecystectomy procedure performed by 13 surgeons. All the videos were labeled with the tools and seven surgical phases and were recorded at 25 frames/second. The pixel resolution of each frame was 1920*1080. Since the information about the presence of surgical instruments might be unavailable in other applications, we didn't use it in designing and evaluation of our method. Further, we wanted to compare our results to those of two other recent researchers, who also did not use the tool information. We used 40 videos for training the CNN, 20 for training LSTM model and 20 for testing. The training set was balanced with 50000 samples per class. Thus, we first used one frame/sec and then randomly picked up more frames to reach a balanced dataset. We used the Inception V1 model for extracting the predictions.

Since the images were big and rectangular, they were resized so that the larger dimension was 350 and the aspect ratio was kept the same as the raw data. The images were then cropped and resized randomly to 224 * 224, which is the default input size of the inception v1 model. We used online data augmentations during training, including flipping left to right and up-down, color distortion, random rotation, etc. The model was initialized with the pre-trained weights from the Imagenet dataset.

To reduce generalization error, we used small l2 regularization and dropout rate of 0,7. The Adam optimizer was used for training the CNN with an initial learning rate of 0.001 and a decay rate of 0.7 after three epochs. The window size for our temporal smoothing model in offline mode was set to 20.

The RNN model consisted of two LSTM layers with 128 and 16 units for both online and offline phases (bi-directional LSTM for offline mode). Since the number of videos for training LSTM was not sufficient, we also used data augmentation for the input sequences. The predictions were saved every 20 frames and for each sequence of predictions,

80 percent of the frames were picked randomly at each step during training. The resulting sequences were of slightly different sizes and phase durations from the original videos. A small random noise was added to the input as well to further augment the data. Since the videos have different sizes, we padded the input sequences with a new class to have equal input size for the dynamic LSTM implementation. The same padding was added for Bi-LSTM and in the backward direction. Stochastic gradient decent (SGD) was used for training the LSTMs.

The total processing inference time for the online model was below 0.02 seconds per image, which is less than the requirements of real-time applications.

4.1.3 Results

In this section, the results from the various experiments with SPD-DLS are shown that helped evaluate its performance. The SPD-DLS performance was then compared with previous work.

To check the performance of the spatial visual extractor, we performed various experiments on the CNN model. We chose Inception V1 over existing deep learning models, because of it's lower memory consumption and faster inference in the real-time mode. Table 4.2 summarizes the accuracy of the convolutional neural networks. The accuracy was based on the entire video frames and didn't take per-class accuracy into account. The first column demonstrates the accuracy of pure CNN without adding the frame numbers. It shows that having information about the time of the frames (frame number) improved the performance of the feature extractor model. There is a difference between the results of the CNN after time-stamping in online and offline modes. This is due to the normalization in the offline mode. During online detection, the absolute frame number was used, which took longer to converge and had less accuracy. However, since the duration of each video

and the transmission time between the phases varied from one video to another, using the absolute value for frame number could cause confusion for the CNN.

During the offline mode, before applying the LSTM model, the output was median filtered, and the result is shown in the last column of table 4.2. Since the input to our LSTM network is the prediction from the CNN, a small increase in accuracy from the CNN makes a big difference in the output of the overall model.

Table 4.2: The accuracy of the CNN models (%)

CNN	time-stamped online	time-stamped offline	median filtered
78.2	79.97	80.68	91.2

Since the testing set was not balanced, to show the final results of the proposed model, we needed to separately check the phase prediction performance of the models, for all of the phases. For this purpose, we used precision, recall and F1-score.

The results of the experiments for the online and offline cases are shown in tables 4.3 and 4.4 respectively. The last columns show the number of images used for calculating the precision, recall, and F1. It can be seen that Cleaning and Coagulation phase has the least accuracy in both online and offline modes. This is probably because the order of the last two phases (Cleaning and Retraction of the Gallbladder) were reversed in some videos. On the other hand, Calot Triangle Dissection and Gallbladder Dissection have the highest f1-score in both modes. The reason is likely the higher duration of these phases according to table 4.1.

The overall accuracy in online mode is 90.8% whereas in offline mode the accuracy is 96.5%. The improvement in accuracy is the result of having temporal smoothing and bi-directional LSTM. This also shows that taking the temporal features into account

using LSTM and applying it to the entire video enables the model to learn the long-term dependencies inherent in surgical videos.

Table 4.3: precision, recall and F1 score for online mode

	precision	recall	F1-score	support
Preparation	0.88	0.90	0.89	2162
Calot Triangle Dissection	0.94	0.97	0.96	17456
Clipping and Cutting	0.72	0.81	0.76	2951
Gallbladder Dissection	0.96	0.92	0.94	19144
Gallbladder Packing	0.87	0.84	0.85	2089
Gallbladder Retraction	0.82	0.65	0.72	1530
Cleaning and Coagulation	0.65	0.74	0.69	2293
avg/total	0.91	0.91	0.91	47625

Table 4.4: Precision, recall and F1 for offline mode

	precision	recall	F1-score	support
Preparation	0.91	0.88	0.89	2162
Calot Triangle Dissection	0.97	0.99	0.98	17456
Clipping and Cutting	0.97	0.95	0.96	2951
Gallbladder Dissection	0.99	0.97	0.98	19144
Gallbladder Packing	0.98	0.97	0.97	2089
Gallbladder Retraction	0.93	0.90	0.91	1530
Cleaning and Coagulation	0.85	0.86	0.86	2293
avg/total	0.97	0.97	0.97	47625

To have a better visualization of the per-class results, a confusion matrix is shown in figure 4.3 (Class 7 is the padded class). The plot on the left is the confusion matrix with the absolute number of frames per phase, whereas the right plot shows the normalized confusion matrix. It can be seen that most of the errors are in the neighboring classes. The exception is the last class (P6), which is Cleaning and Coagulation. This is probably because it is very likely that during the Gallbladder Dissection (P3), the resulting bleed-

ing requires a special instrument such as Bipolar and Irrigator to do the coagulation and cleaning in P3 as well. This can be seen in the fourth row, too, which is the Gallbladder Dissection phase. The highest accuracy is obtained in the second phase, which is the Calot Triangle Dissection. The reason is probably the availability of more images for training due to the longer duration. The high performance in Gallbladder Packing phase (P4) is because of the distinct form of the step, which is accomplished using a grasper and packing the gallbladder in a distinctly shaped specimen bag.

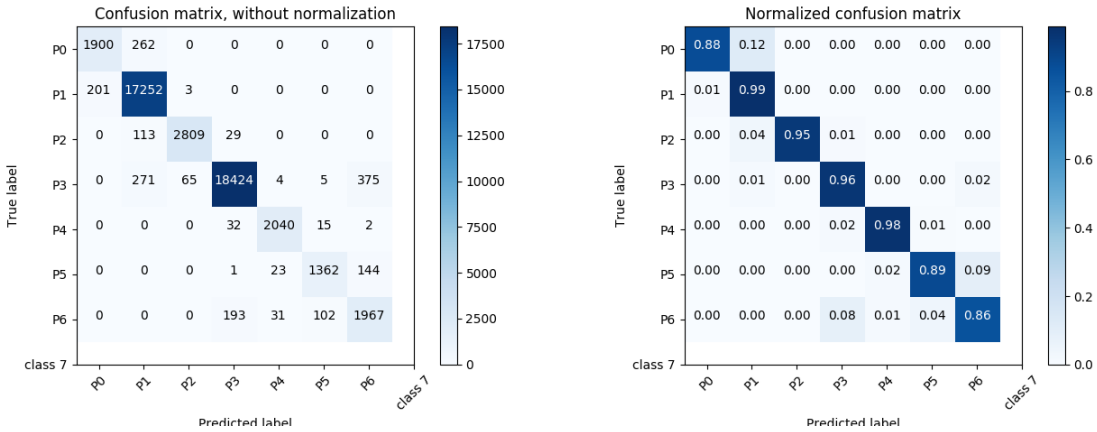


Figure 4.3: Confusion Matrix with (right) and without (left) normalization for the offline mode

To validate the proposed model, we compared our work with the most recent papers. The result is shown table 4.5. It is worth noting that the CNN model used in Endonet model was Alexnet and SVRCN used Resnet50, whereas in SPD-DLS, Inception V1, which a shallower model, is used to compare against Resnet50. On the other hand, all of the previous work used the first 40 videos for training and the other 40 for testing. SPD-DLS used the first 40 videos for training the CNN, videos 40 to 60 for training the LSTM and the last 20 videos for testing.

Table 4.5: Overall Accuracy

model	online	offline
endonet (HMM) [105]	82.0	91.0
endonet (LSTM) [105]	88.6	92.2
SVRCN [44]	N/A	92.4
SPD-DLS (ours)	**90.8**	**96.3**

The higher accuracy in the proposed model, in offline mode, showed that considering both short-term and long-term dependencies of the surgical video frames can significantly improve the performance. Moreover, using the frame number information helps the entire model use the time of the frame being classified. Furthermore, the order of the phases is automatically learned by the LSTM, unlike the other recent approaches. Having the training sets separated for the CNN and LSTM, and the diverse data augmentation techniques are the main reasons for the improvement in the performance of the proposed model compared to the existing methods.

4.2 Phase Boundary Detection

The method described in the previous section for analyzing the workflow of a laparoscopic procedure relies on a multi-class classification for all the frames of a video. Although this approach provides promising results by taking into account the short-term and long-term correlations of the phases, it lacks the capability for explicitly determining the exact transition time (frame) between two consecutive phases. Unless we have perfect accuracy for frame-level predictions, it is not possible to identify the beginning and the ending frames corresponding to a particular phase. This is an important limitation, considering the significance of the information at the boundaries of different phases.

Directly detecting the boundaries of each phase can be seen as an alternative solution for video segmentation. In an information retrieval system, a specific phase or task can be

requested upon a query, to be further analyzed manually or with the aid of an automated system. A video segmentation model based on the automatic detection of the boundaries of each phase can reliably be used, instead of the frame by frame analysis of each phase.

Despite the distinctive features of each phase of a laparoscopic procedure, especially at the borders, the identification of such boundaries using a deep learning technique is far from trivial. The challenges in finding the beginning and the ending frames in laparoscopic cholecystectomy videos include the following considerations.

Firstly, laparoscopic cholecystectomy is typically a lengthy procedure and might take up to one and a half hours. For instance, the duration of the videos in the validation set of the cholec80 dataset that we used for the experiments in this chapter, has an average of 2454 and a standard deviation of 1054 seconds, a minimum of 739 and a maximum of 5993 seconds. Since the identification of the starting and ending frames of each phase requires access to the entire video, the longer duration of the videos is the main challenge in designing an end-to-end model. As was described earlier, the processing of all the frames to capture the visual features using a CNN requires a huge amount of memory to perform in a reasonable time. Furthermore, the long-term dependencies between the frames of a video need to be extracted, which can be accomplished with an RNN. However, even the best RNN architectures are known to have limitations in processing long sequences. When we use RNNs for classifying a given frame, the impact of different frames, which are much further away, becomes less critical due to the vanishing gradients. This might not be an issue in frame-level classification/segmentation, since for frame level classification, a long short-term memory of a few hundred frames might be sufficient. Furthermore, transition frames can occur anywhere in the video.

Secondly, the duration of each phase varies significantly. For example, a given phase length could be between 20 to 200 frames. We call this intra-phase variation. When a one phase is significantly longer or shorter than another phase, we call it the inter-phase

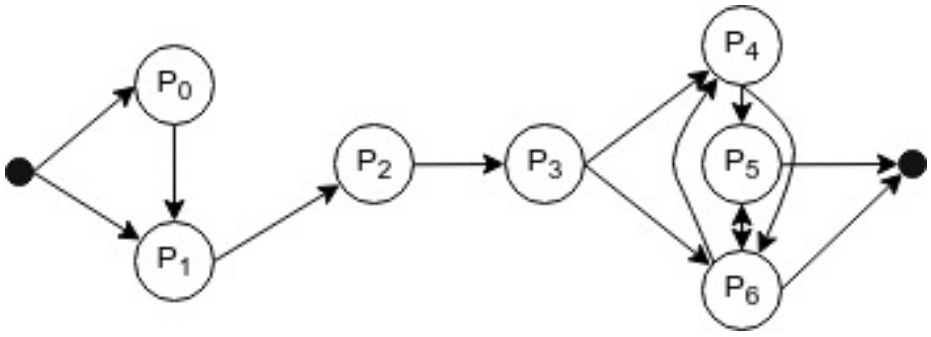


Figure 4.4: The ordering of the phases in cholec80 dataset

variation. This is shown in Table 4.1. While for the frame-level segmentation, inter-phase variations can be addressed by the balancing techniques described in section 4.1, both inter-phase and intra-phase variations introduce challenges in detecting the transition time.

Lastly, the surgical tasks in laparoscopic cholecystectomy might not strictly follow the same order in all the videos. Figure 4.4 shows the diagram of the workflow for cholec80 dataset. It can be seen that the Cleaning and Coagulation phase (P6) can occur before Gallbladder Retraction (P5) or Gallbladder Packing (P4) or after both of them. Moreover, the total number of phases might not be the same in all videos. The two exceptions are Preparation (P0) and Cleaning and Coagulation phases, which might not be present in all videos. The variation in the size and the order of the output also introduces significant challenges in designing an automated approach based on deep learning, since most of the existing approaches require the outputs to have either a fixed size/length or the same size as inputs.

In order to address the above challenges in determining the boundaries of each surgical phase, we proposed a novel method based on the sequence-to-sequence (seq2seq) model [96] and attention mechanism [11].

In the following section, the methodology of the proposed model which we call Attention-based Phase Boundary Detection (APBD) is described.

4.2.1 Methodology of APBD

In order to find phase boundaries in a surgical video, we associate a tuple of integers to each phase, which is the indices (frame numbers) of the first (the beginning) and the last (the ending) frames. Therefore, the desired output is a sequence of integer tuples. Each element of this sequence corresponds to a particular phase.

The length of the output sequence is the total number of phases that are present in a video. As was mention before, the length and the ordering of the output are not necessarily the same in all videos. Furthermore, the lengths and the orders do not depend on the size of the input sequences, which themselves can be variable. Therefore, the problem of finding the phase borders is equivalent to mapping a variable-length sequence to another variable-length sequence, where the length of the output sequence is independent of the length of the input.

We adopted the seq2seq approach [96] to map our variable-length surgical videos to the sequences of phase boundaries. Though this method is an effective solution to similar problems such as machine translation, there are two main limitations.

Firstly, in seq2seq modeling, the entire input sequence is *encoded* in a fixed-size vector, which is typically the last state of an RNN. This vector contains the information of the input sequence. As was described earlier, retaining long-term memory is highly challenging, as surgical videos are usually long.

Another limitation of seq2seq is that each output element is usually independent of the position of the elements of the inputs. A potential solution for this problem is to project each element of the output to a single value and train the system using regression methods. However, only one of the boundaries can be determined using this method. Also, training a regression-based model requires a large number of sample, due to the high inter-phase and intra-phase variations explained before.

To address these issues, we employed the ideas from attention mechanisms [11, 60], which are aimed at extracting the impact of the elements of the input sequence on each element at the output. In the proposed attention-based method, the output of the system is the sequence of phases that are present in the videos, and the attention mechanism finds the *alignments* for each element at the output to each position of the input sequence. These alignments are probability distribution vectors and are called the attention vectors too. Using this technique, the boundaries (we also call them pointers) are chosen from the input sequence according to the alignment vectors. Some of the advantages of our method include:

- The attention vectors connect all the elements of the input, which are the frames of a video to each element at the output, which is the corresponding phase. Thus, the long-term memory of the model is preserved.
- In contrast to regression-based methods, by selecting the indices of phase boundaries as frame numbers and from all the frames, the range of the output is constrained by the length of the videos. Also, the outputs are always integers. By limiting the scope of the outputs, the training can be accomplished easier and with fewer samples.
- By assigning two different attention vectors, the beginning and end frames can be found simultaneously.
- The size of the model does not depend on the size of the output since the number of weights for the attention mechanism remains the same for output elements.
- The variation in the ordering of the phases do not impose any limitation to the model. The reason is that phase, and boundaries associations are accomplished independently for each output, regardless of the order of phases.

In this section, the methodology for designing a deep learning system for detecting the borders of each phase in a laparoscopic video is described. The problem definition is first established in section 4.2.1.1. The fundamentals of the seq2seq model and attention

mechanism is described in details in sections 4.2.1.2 and 4.2.1.3 respectively. The details of the proposed APBD system is explained in section 4.2.1.4 and is followed by the training strategy in section 4.2.1.5.

4.2.1.1 Problem Definition

Let $\mathcal{D} = \{(X^{(i)}, Y^{(i)}) | 0 < i \leq N\}$ be a dataset of N surgical videos, where $X^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_{M_i}^{(i)})$ is the sequence of input frames of the video i and $Y^{(i)}$ is the set of the corresponding phases. $|X^{(i)}| = M_i \leq M$ is the length of video i with M is the length of the longest video, and $|Y^{(i)}| \leq K$ is the number of phases present in the video. The phases in Y can have arbitrary order and lengths for each video. The goal is to design a model F that maps a variable length video X to a set of corresponding phases Y.

Each element of the input sequence $x_j^{(i)} \in \Re^d$ is a fixed size vector of extracted visual features for the jth frame of the ith video, where d is the size of the feature vector and $j \leq M_i$. The visual features are extracted using a CNN, as described earlier in this chapter.

Each phase of a video is defined by a 3-tuple $y_k^{(i)} \triangleq (u_k^{(i)}, v_k^{(i)}, p_k^{(i)})$, where $u_k^{(i)}$ and $v_k^{(i)}$ are the starting and ending frames of the phase k, and are one-hot vectors with the size of M. $p_k^{(i)}$ is the probability that the phase k is present in video i. The phase association for video i can also be represented as a K dimensional vector $P^{(i)} = (p_1^{(i)}, p_2^{(i)}, ..., p_K^{(i)}) = \{0, 1\}^K$, where each element is 1 if the phase is present and 0 otherwise.

We formulate the problem of detecting the boundaries of the phases as a multi-class classification, which maps the sequence of extracted features X to the tuple (U, V, P), where $U = (u_1, ..., u_K)$ and $V = (v_1, ..., v_K)$ are of the same size K.

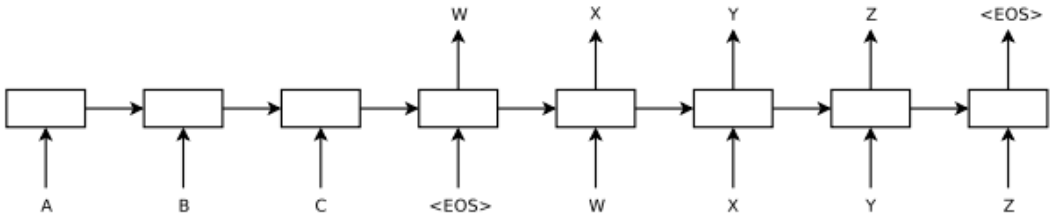


Figure 4.5: Sequence to sequence architecture for language translation [96]

4.2.1.2 Sequence-to-Sequence Modeling

RNNs are powerful tools for processing sequential data such as video, audio, text, etc., and are suitable for variable-size input as well. The output of an RNN can be either the hidden states of all the elements or the final state. However, RNNs can operate on only one sequence as input. In order to address this limitation, the well-known seq2seq from natural language processing is adapted for our application.

Introduced for machine translation [96], sequence to sequence modeling aims at mapping variable-length input sequences to another variable-length target sequences. The model consists of an encoder component and a decoder component and is also called an encoder-decoder architecture. The role of the encoder in a basic seq2seq model is to encode the input into a fixed-size vector called *context* vector. The context vector represents the whole sequence (for example a sentence from the input language) and is used in the decoder as the initial state to generate the output sequence (the corresponding sentence in the second language). The block diagram of a seq2seq model is shown in figure 4.5.

Typically, both the encoder and the decoder in a seq2seq model are RNNs. In the encoder, the last state is taken as the ”context vector” and the initial state of the decoder. The RNN in the decoder part is designed as a generative model (auto-regressive mode). In order to start the generation, two tokens are defined as the start of the sequence (SoS) and the end of sequence (EoS) (See Figure 4.5), which are used to determine the beginning and

the ending of the generated sequence. The input to each unit of the RNN (after the SoS token) in the decoder is taken from the output of the previous step, which is after applying an output layer (usually a fully connected layer) to the hidden state. The generation continues until reaching the EoS output or a pre-defined maximum output sequence length.

The main disadvantage of the seq2seq model is the lack of long-term memory in the RNN of the encoder. In other words, a fixed size context vector might not be sufficient for retaining the information in a long sequence. To address this issue, the attention mechanism has been proposed, which is explained in the following section.

4.2.1.3 Attention Mechanism

Inspired by human's intuition, attention is defined by the importance of each element of input in making a particular decision at the output. With the help of attention, the alignment of the elements in the input and output sequences is used to determine the dependencies between the sequences. Attention mechanism has been widely used in natural language processing providing the state-of-the-art in translation [108], text summarization [89], question answering [56], etc., and in computer vision such as image captioning [115].

In a seq2seq model with attention, the fixed-size context vector is replaced by a dynamic context vector, which is determined by assigning different weights to all the elements of the input using the attention mechanism. At each step of the decoder, a context vector is generated by "attending" at specific positions in the input sequence, and the decision is made based on this context vector and the information from the outputs at the previous steps.

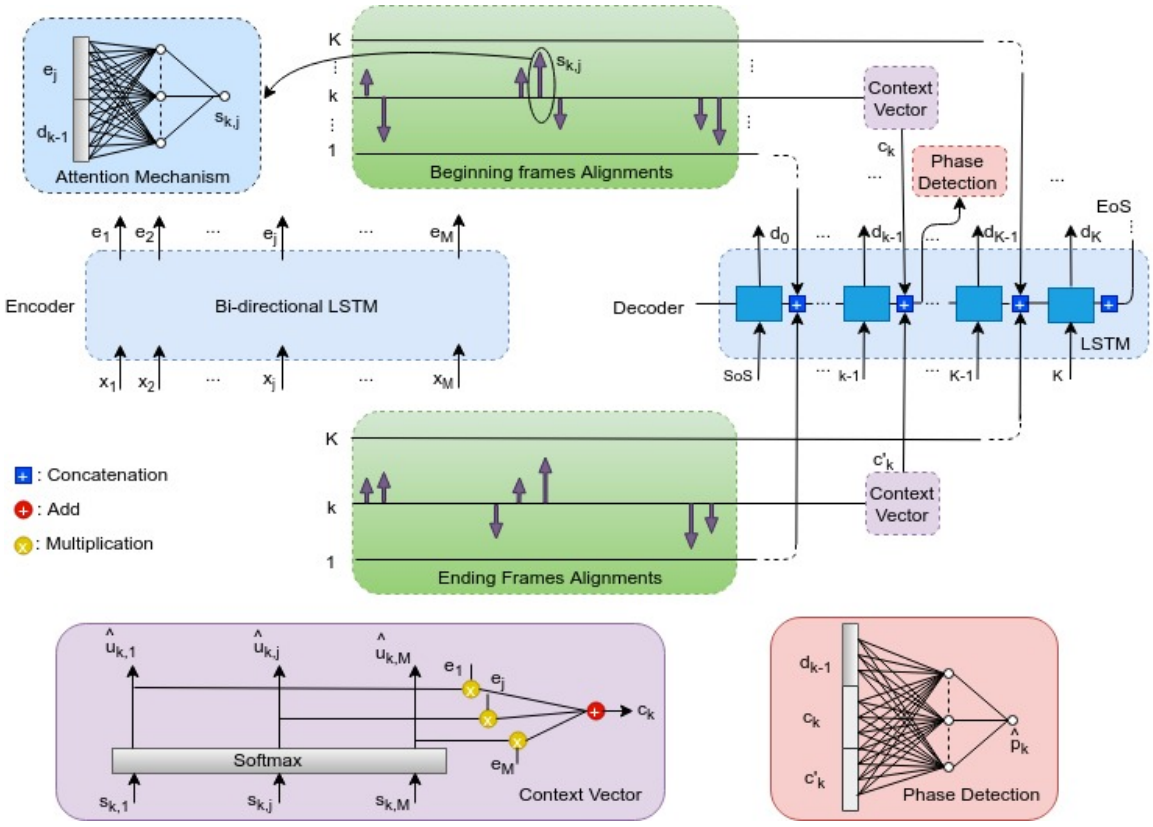


Figure 4.6: The block diagram of the attention-based phase boundary detection model

4.2.1.4 Attention-based Phase Boundary Detection

The proposed model for detecting the boundaries of surgical phases is designed based on a sequence to sequence (seq2seq) model with attention, to map all the frames of each video to a variable-length sequence of the starting and ending frames of each phase. We modify the attention mechanism explained above to have two attention vectors for the beginning and end of the surgical phases, and the outputs of the model are the alignment (attention) vectors. The attention mechanism used in this section is based on [60]. The block diagram of the model is shown in Figure 4.6.

The encoder in the seq2seq model consists of a bi-directional LSTM, with the forward hidden state $\overrightarrow{e_j}$ and the backward hidden state $\overleftarrow{e_j}$, forming the hidden state e_j after

concatenation, for $0 < j \leq M$. The input to the bi-directional LSTM is a sequence of extracted visual features for all the frames of a video.

For the decoder, the hidden state d_k represents the output of an LSTM, which is going to be used for detecting the phases of a video. The decoder state d_k is a function of the input to the state, which is the target of the previous state, the hidden state of the previous state d_{k-1}, and the context vectors c_k and c'_k for the current state. The context vector and the hidden state of the previous state are concatenated $[c_k; c'_k; d_{k-1}]$, and the resulting vector is used as the previous hidden state.

The context vectors c_k and c'_k correspond to the beginning, and the ending frame of phase k and are the weighted sums of the encoder states and are calculated as:

$$c_k = \sum_{j=1}^{M} u_{k,j} e_j \tag{4.1}$$

$$c'_k = \sum_{j=1}^{M} v_{k,j} e_j \tag{4.2}$$

$u_{k,j}$ and $v_{k,j}$ are the attention vectors and determine the *alignments* of the input state at jth time step e_j with the output at kth time step p_k, i.e. the impact of each input frame in making decision about each phase of the video. The context vector c_k encodes the impact of all of the frames in the input and thus, resolves the short-term memory issue in the encoder.

In order to find the attention vectors, an attention score is calculated for each element connecting input and output, using the hidden states of the input and the output at the corresponding time steps. The attention score for the beginning frame for phase k and the frame j can be found by a feed-forward neural network with a single hidden layer and can be formulated as:

$$s_{k,j} = v_a^T tanh(W_a[d_{k-1}; e_j]) \tag{4.3}$$

where v_a and W_a are trainable weights of the attention mechanism. A similar formula is used for the ending frame's attention score $s'_{k,j}$, with the trainable weights v'_a and W'_a.

The attention vectors are then calculated by normalizing the attention scores, resulting in a probability distribution among all the input states. This can be accomplished by applying the Softmax function. For example, for the beginning frame attention vector, we have:

$$u_{k,j} = softmax(s_{k-1,j}) = \frac{e^{s_{k,j}}}{\sum_{j'=1}^{N} e^{s_{k,j'}}} \tag{4.4}$$

The attention vector $u_{k,j}$ from the above equation can be interpreted as the probability of the input at state j to be a beginning boundary for phase k. Since the boundaries are selected from the sequence of frames of the input, the attention vector can directly be trained to output the beginning and end of the phases.

The proposed model is similar to pointer networks [109] in using the attention vector to select an output from an input sequence. However, in our model, the attention mechanism uses the encoder states instead of the inputs. With the help of the encoder states, the dependencies between the elements of the input sequence are considered, and therefore, provides better performance than the pointer networks.

Also, the proposed double attention method is different from multi-head attention in the transformer architecture [108] In multi-head attention, multiple attentions are aggregated into a single one, whereas in our model, the attention vectors are concatenated, forming a longer attention vector to be used in the decoder.

4.2.1.5 Training of APBD

The generative RNN at the decoder of a seq2seq model uses the output from the previous step as input. During training and after the model is trained, the generation of the

output elements starts after the SoS token, and the output is used as input for the subsequent step. The output at each step is typically taken after a fully connected layer on top of the RNN. This method is suitable for inference when the ground truth is not known, but it can cause slow convergence during training and might be unstable.

To remedy this issue, the training can be performed using a method called *teacher forcing* [113]. By teacher forcing, the target output is used instead of the generated output as the input to the new step. While this technique is more stable and effective for training, it might lead to poor performance during inference, as the test sequences diverge from the previously seen sequences. In the literature, there are several approaches to address this limitation, such as [13, 53].

In our model, the role of the decoder is to produce the sequence of attention vectors, which are the pointers to the boundaries of the phases that are determined by the outputs of the decoder. The input sequence can be the fixed-length sequence $(SoS, 1, 2, ..., K)$. The generation of the sequence starts after the SoS token and continues for K+1 steps. This input sequence is the same during training and inference, and therefore, the method is similar to the teacher forcing method.

We can define the loss function for training the attention vectors. The final output of the model is the tuple $(\hat{U}, \hat{V})$ and the loss function is the sum of the Softmax cross-entropy for the two pointers:

$$L_a = -\frac{1}{NK}[\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{M} u_{k,j}^{(i)} \log \hat{u_{k,j}^{(i)}} + \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{M} v_{k,j}^{(i)} \log \hat{v_{k,j}^{(i)}}] \tag{4.5}$$

The decoder states, which are the hidden states of the LSTM cell concatenated with the context vectors, is connected to a fully connected layer, which we call the projection layer to produce the probability of the presence of each phase. The resulting output is formulated as a multi-label classification and is trained using sigmoid cross-entropy:

$$L_p = -\frac{1}{N}\sum_{i=1}^{N} P^{(i)} \log(\sigma \hat{P}^{(i)}) \tag{4.6}$$

where σ is the sigmoid function and $\hat{P}$ is the output of the phase detection block at the decoder.

The training of the model is accomplished by jointly training with the two loss functions in a multi-task learning fashion similar to 3.2.5. In order to accomplish this, two training operations are defined with separate learning rate schedules and trainable weights.

For the loss function of equation 4.5, the trainable weights are all the weights except the weights in the projection layer at the decoder. The weights include the encoder weights (the weights of the bi-directional LSTM) W_e, the attention weights $[W_a, V_a, W'_a, V'_a]$, and the weights of the LSTM at the decoder W_d.

For the loss function in equation 4.6, back-propagation is performed only on the decoder, and the trainable weights are the decoder LSTM weights W_d and the weights of the projection layer W_p. Note that the only weights that are shared between the two training operations are the decoder LSTM weights W_d.

4.2.2 Experiments and Results

In this section, the experiment for validating the proposed model for detecting the boundaries of each phase in a laparoscopic cholecystectomy video is described. We used the cholec80 dataset, with the first 40 videos for training and the last 40 videos for validation.

The inputs to the proposed model were the extracted feature vectors from the frames of each video using a CNN with the Inception V1 architecture. The size of the feature vector was 1024. We created three sets of data; one with no data augmentation for all the frames, one with the same augmentation functions for all the frames (central crop and

Table 4.6: Mean absolute error in seconds for each phase in cholec80 dataset

	P0	P1	P2	P3	P4	P5	P6	Mean
Beginning frame	0.0	27.67	64.65	108.22	46.57	48.07	71.87	52.53
Ending frame	33.25	41.6	65.87	38.82	58.05	0.0	81.22	45.54

horizontal flip), and one with random augmentation similar to section 4.1.3. While using the same augmentation for each video preserves the spatial and temporal relationships, the random augmentation introduces more noise to the original pattern and reduces overfitting.

To apply data augmentation in the time direction, the feature vectors were saved every ten frames, forming sequences that are 2.5 times longer than the original duration of the videos in seconds. At each step of the training, for each video, a sequence was randomly selected that has a length between 30 to 50 percent of the augmented sequence. We ensure the maximum length was less than 6000 frames, which is close to the longest video in the validation set. Using this data augmentation method, we had variable length training input for each video with different boundaries of the phases. At each step, the shorter videos were zero-padded to the longest video of the batch.

The encoder consisted of a two-layer bi-directional LSTM with 64 and 16 hidden units. These numbers were picked by trail and error. A dropout layer with the same mask along all the frames was added after the input layer. At the decoder, the LSTM had 32 hidden units, and the SoS and EoS tokens were 0 and 8 respectively.

To evaluate the performance of the proposed model, we used the Mean Absolute Error (MAE) in seconds or frames (one frame per second). The results for each phase in the cholec80 dataset are shown in Table 4.6. As an example, for P3, the difference between the ground truth and our results for the beginning of P3 is 108.22 seconds and 38.82 secs for the ending of the same phase. The MAE in seconds for the beginning frames is 52.43 and for the ending frames is 45.54.

The zero MAE in The Preparation (P0) phase for the beginning frame pointer is because almost all the videos in test set start with this phase and the beginning of the video is the beginning of P0. For the ending frame pointers, Gallbladder retraction (P5) has zero MAE, which is likely because retracting the gallbladder has a definite ending. As shown in figure 4.4, P5 can also be the last phase of the surgery.

The worst MAE belongs to the Cleaning and Coagulation phase (P6). The reason is likely the irregular pattern in the ordering of the phase, according to Figure 4.4. The high MAE for the Gallbladder Dissection phase (P3) is most probably because of the similarity of the features between the Calot triangle Dissection (P1) and P3.

The proposed model might not guarantee the ordering of the phases that are detected at the decoder. Though the accuracy at the output of the projection layer is 100%, the boundary pointers and the phases might not be correctly associated. On the other hand, the transition time for the surgical phases is not necessarily a single frame as it is annotated in the dataset. Therefore, we evaluate the performance of our model by calculating the percentage of the predicted boundaries within different ranges, i.e., the accuracy of the model when different ranges for error are acceptable. The results are shown for the beginning and the ending frame in Tables 4.7 and 4.8 respectively. For instance, on row 3 of Table 4.7, the probability of the *beginning* frame of the phase P4 before being detected within 10 seconds of the ground-truth is 32.5%. Similarly, on row 3 of Table 4.8, the probability of the *ending* frame of the phase P4 before being detected within 10 seconds of the ground-truth is 52.5%.

It can be seen that more than 48% of the predicted boundary pointers have less than 5 seconds error with respect to the ground truth, whereas the exact match accuracy is less than 19%. This is likely because the transition from one phase to another might take a few seconds. Another possible reason could be that the training was performed using softmax CE loss, which does not produce a range of error in the output.

Table 4.7: The accuracy of the beginning frames detection for different ranges of error in seconds

	P0	P1	P2	P3	P4	P5	P6	Mean
err = 0	1.0	0.075	0.125	0.075	0.0	0.05	0.025	0.19
err $\leq$ 5	1.0	0.525	0.425	0.475	0.175	0.45	0.275	0.475
err $\leq$ 10	1.0	0.55	0.5	0.575	0.325	0.55	0.325	0.54
err $\leq$ 20	1.0	0.675	0.525	0.625	0.575	0.675	0.55	0.66
err $\leq$ 30	1.0	0.775	0.6	0.7	0.675	0.675	0.6	0.72

Table 4.8: The accuracy of the ending frames detection for different ranges of error in seconds

	P0	P1	P2	P3	P4	P5	P6	Mean
err = 0	0.1	0.075	0.05	0.0	0.025	1.0	0.05	0.185
err $\leq$ 5	0.425	0.475	0.575	0.225	0.35	1.0	0.35	0.486
err $\leq$ 10	0.5	0.55	0.675	0.45	0.525	1.0	0.375	0.582
err $\leq$ 20	0.625	0.575	0.75	0.65	0.65	1.0	0.45	0.671
err $\leq$ 30	0.675	0.625	0.825	0.75	0.7	1.0	0.5	0.725

The lowest accuracy for all error ranges belongs to P6, which matches the results in Table 4.6. This might be because cleaning and coagulation can occur during the dissection phases as well, and, using spatial features as the input of the model might not capture the temporal information needed for classifying the frames as a separate phase.

Another observation from the tables is that the error for the beginning of one phase is not the same as the error for the ending of the previous phase, especially for the phases P1 to P4 which always have the same ordering (figure 4.4). This shows that the predictions for the boundaries do not depend on each other, which is because the loss function for the pointers is the sum of the loss functions for the beginning and the ending borders, rather than one loss function for both of them.

4.2.3 Conclusions

A new method called SPD-DLS is proposed in section 4.1 and validated for detecting videos from a laparoscopic procedure into surgical phases. The model takes advantage of

the spatial and temporal features to classify all of the surgical videos. A CNN followed by median filtering extracts useful information to send to an LSTM model for getting the final output of detected phases. In the online mode, the CNN is aware of the frame time and can detect phases in real-time. There are a few possible future improvements to our method. The temporal median filtering method applied to the final prediction of the CNN is blind to the actual visual features of the frames. We are planning on replacing it with an RNN that can capture short-term correlations.

In section 4.2, a novel approach was proposed to detect the transition time of different phases of laparoscopic surgery by learning the beginning and end frames of each phase of the surgery. An attention-based sequence-to-sequence method is utilized to operate on variable-length inputs and outputs. This method is more efficient for segmenting videos than the frame-level classification approaches. The main advantage of the attention-based method is that the size of the model does not depend on the length of the output sequence, which is the number of phases, as the attention mechanism benefits from the weight sharing property, similar to RNNs. Furthermore, the segmentation is accomplished by finding single frames associated with the beginning and the end of a phase. Therefore, the proposed attention-based architecture can be easily used to detect surgical tasks with overlapping or separate sections. Nonetheless, several limitations have to be addressed to improve the performance of the model.

Firstly, using the extracted features from a CNN as the input might not be sufficient, as the CNN doesn't capture the temporal dependencies and the critical information at the boundaries. Future work can include the replacement of the CNN with a CNN-RNN as described in chapter 3.

Secondly, the index of the output determines which phase corresponds to the extracted beginning and ending frames at the decoder. In other words, the phase is not ex-

plicitly learned. This can be addressed by modifying the projection layer and developing a new training strategy with a sequence loss function.

Lastly, due to the smaller size of the training dataset, the performance can vary significantly by different choices of the hyper-parameters. We believe that even with such small data sets, choosing a better training strategy, or loss functions can lead to better results.

CHAPTER 5

CRITICAL VIEW OF SAFETY

Introduced in 1985 [58], laparoscopic cholecystectomy, much like other laparoscopic procedures, has become the preferred choice over open surgery, due to less pain and faster recovery time [49]. Even though laparoscopic cholecystectomy is the most commonly performed procedure and is part of the general surgery residency training, the incidence of bile duct injury is around 0.3% (3 per 1000 procedures) [17]. The bile duct injury is usually caused by the misinterpretation of the anatomy and can have devastating consequence to patients leading to significant morbidity and mortality [17]. The critical view of safety (CVS) is introduced as an effective method to reduce bile duct injuries [94]. An example of the critical view of safety is illustrated in figure 5.1.

The goal of clearly observing the CVS is to avoid misidentifying the common bile duct or an aberrant duct as the cystic duct and therefore, preventing bile duct injuries. Figure 5.2 shows the stylized anatomy of the critical view of safety. The CVS is accomplished by clearly viewing the main anatomical structures that includes the Calot's Triangle, Hepatocystic triangle and the two structures (cystic duct and the cystic artery) [95].The establishment of CVS is critical for ensuring the safety in a laparoscopic cholecystectomy procedure.

Despite the significance of the CVS, there is a lack of a thorough analysis of CVS using visual records. Currently, the rating and evaluation of the quality of CVS is accomplished by manually reviewing short videos or images by expert surgeons or crowdsourcing [24]. However, this evaluation is very time consuming and expensive and, might

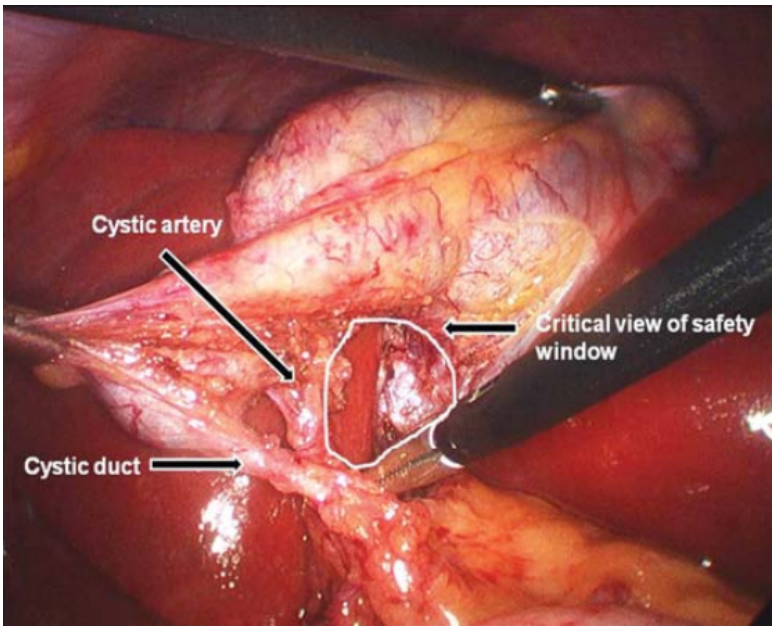


Figure 5.1: The critical view of safety [45]

result in potentially inaccurate subjective conclusions. Thus, an automatic method for analyzing the CVS is desirable in the assessment and rating of the videos.

In this chapter, we provide the results of a deep learning model trained to identify the CVS directly from laparoscopic videos.

The remainder of the chapter is organized as follows: in section 5.1, the main criteria for evaluating the critical view of safety is reviewed. The CNN model for detecting the CVS in a laparoscopic cholecystectomy video is described in section 5.2, followed by the results in section 5.3 . In the last section, we discuss the future direction of the research.

5.1 Criteria for Evaluating the Establishment of CVS

One of the recommended methods for checking the establishment of the critical view of safety in laparoscopic cholecystectomy is through an intra-operative time-out, where the procedure is paused to make decisions. The establishment of CVS can be confirmed by

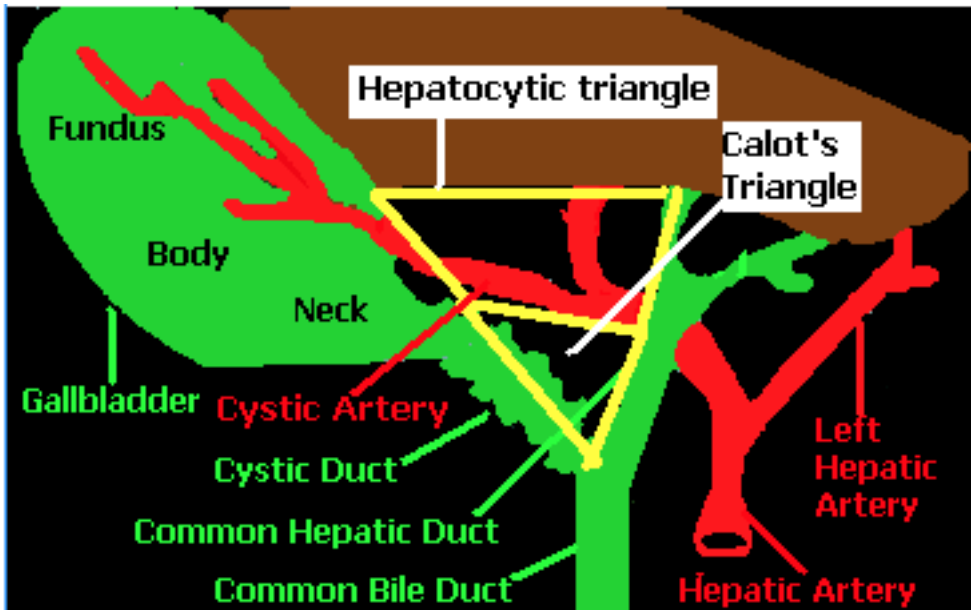


Figure 5.2: The anatomy of gallbladder, cystic duct and cystic artery [81]

using the doublet view method [82], which is visualizing the main components of the CVS using both anterior and posterior views. This is shown in figure 5.3.

Currently, the post surgical evaluation of the achievement of the CVS is accomplished by storing the images from the doublet views and manually assessing the quality by assigning a rating score to each image or video. The examination of the CVS is usually performed by expert laparoscopic surgeons.

The scoring system used by Laparoscopic surgeons for rating the CVS from the images shot during surgery is based on three criteria: hepatocystic triangle is cleared of fat and fibrous tissue, dissection of the lower one third of the gallbladder from the liver to expose the cystic plate and two and only two structures should be seen entering the gallbladder. The CVS with exposed cystic plate is shown in figure 5.4. The point-based scoring system assigned 2 points for clearly viewing the structures, 0 point for not having an immediate view and 1 point for non-optimal view [82].

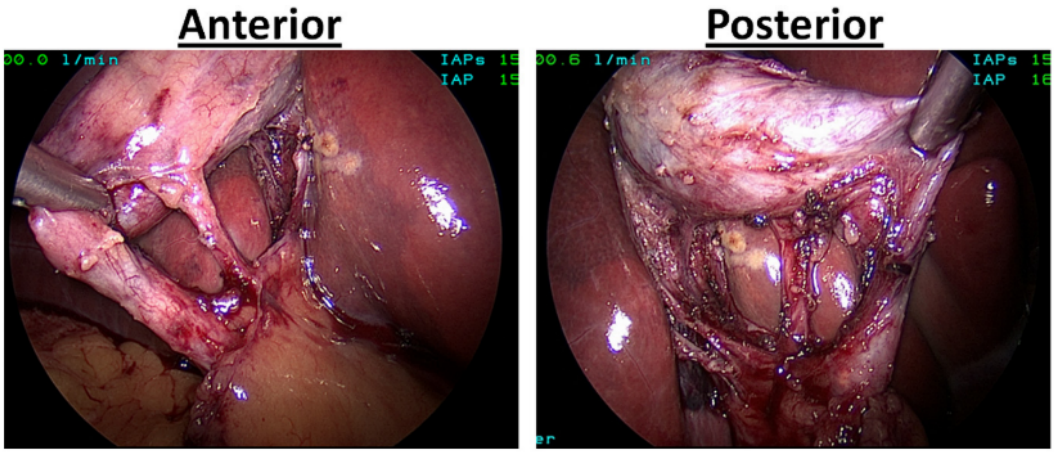


Figure 5.3: The doublet view of the critical view of safety [82]

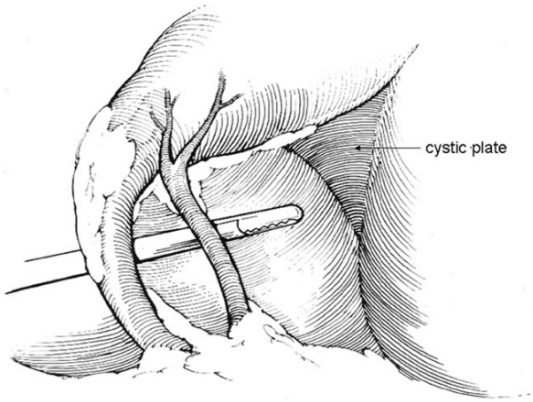


Figure 5.4: Cystic plate [63]

The detailed criteria for scoring is as follows:

Two structures connected to the gallbladder:

- 2 points: Two structures can immediately and clearly be seen connecting to the gallbladder.
- 1 point: Two structures can be seen connecting to the gallbladder, but there is some overlap of duct and artery or a technical feature, such as poor lighting or lack of color contrast, that interferes with the clarity of determination. The photograph requires further study to make a definitive assessment.

- 0 point: Due to overlap or technical issues, 2 separated cystic structures cannot be seen.

Cystic plate clearance:

- 2 points: Cystic plate is immediately clearly visible to approximately its bottom one third.
- 1 point: Cystic plate is visible but overlapped by other structures so that it is not optimally seen or an insufficient amount of the plate is shown. The photograph requires further study to make a definitive assessment.
- 0 point: Cystic plate not visible due to positioning, light, obstruction of view by instruments, or coverage with clot.

Hepatocystic triangle clearance:

- 2 points: Hepatocystic triangle is cleared of tissue so that visibility of cystic structures and plate are completely unimpeded, but also assure that the viewer can ascertain that no other structures are in the triangle.
- 1 point: Somewhat less than the whole triangle can be clearly seen, or technical issues reduce the ability to see optimally. The photograph requires further study to make a definitive assessment.
- 0 point: Tissue in the triangle obscures view of cystic structures, cystic plate and does not allow the conclusion that there are no other structures in the triangle. Or technical issues prevent determination of how well cleared the triangle is.

Based on this scoring system, each video or image is evaluated on different aspects of the achievement of the CVS. The resulting set of rated videos will then be used as a valuable source of education for training future surgeons.

5.2 Approach

In this section, we investigate the effectiveness of using deep learning and CNNs in particular, in identifying the critical view of safety in a laparoscopic cholecystectomy video. A CNN-based deep learning system can be applied to automatically and objectively assess surgery videos for quality improvement, video-based coaching, and the technical and cognitive milestones assessments of surgical residents. The objective is to detect the presence of CVS in each frame of a cholecystectomy video.

We used the publicly available cholec80 dataset[104], which contains 80 videos of laparoscopic cholecystectomy surgery. Since the videos in this dataset do not have labels for the CVS, we manually annotated the dataset based on the achievement of the CVS. We used the first and third criteria explained in the previous section, which are based on the clear view of the two structures and the Hepatocytic triangle, as detecting the cystic plate is not easy for a non-expert. Therefore, we used a binary score instead of the 0-6 scoring system.

All the frames from the video (25 frames/second) were extracted and manually labeled as either 1 (CVS is present or ”positive”) or 0 (no CVS is present or ”negative”). The training set contained 50 videos; ten videos were used as the validation set for tuning the hyper-parameters of the CNN and the final 20 videos for testing. All the videos in training, validation, and test sets included instances, where CVS was not achieved.

We trained a CNN based on the Inception-V3 architecture [99]. Since the CVS is present shortly (usually 3-5 seconds) before the clipping and cutting of the Calot triangle (table 4.1), the training set suffers from a high imbalance issue. To tackle the imbalance issue, both over-sampling and under-sampling of the available frames were implemented using all of the frames in the CVS region and one frame per second outside that region. Despite this attempted solution to the imbalance issue, there was very significant number of frames that did not contain CVS s and a very low percent of unique frames that did.

Thus, we applied the class re-weighting method after re-sampling. The loss function that was used for training is the binary cross-entropy function. To further improve the results of the CNN, the predictions were smoothed using a median filter with the size of 20.

5.3 Results

Since this is the first time CVS detection has been attempted, there is no previous work to compare with. Also, as the labeling was performed by non-experts, there might be minor inconsistencies in the criteria that was used for identifying the CVS in all the videos of the training and the test sets.

In order to evaluate the performance of the CNN, we used the accuracy, precision and recall criteria. The reason for choosing precision and recall is the high imbalance of the validation and test sets, as mentioned above in our approach. Our accuracy using this limited dataset is 94.6%. In other words, 94.6% of the frames were correctly predicted. The precision is 70%, which shows the ratio of the correctly predicted CVS frames to the total number of frames that is predicted as the CVS. The recall of 65.5% shows the percentage of the correctly predicted frames as the CVS over all frames where the CVS is present.

Figure 5.5 shows an example of true positive, which is the achievement of the CVS and, a true negative sample, which shows the absence of the CVS. The true negative example is selected from the frames that belong to the short stopping time (intra-operative time out) before the clipping and cutting starts (where the CVS needs to be checked).

For the 20 test videos, the results were obtained in real-time, which is a major improvement over manual inspection by trained examiners or experts.

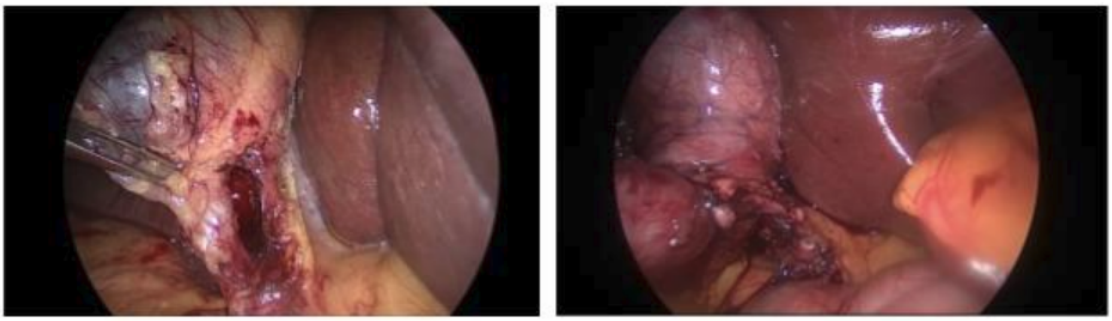


Figure 5.5: (left) True positive results showing the establishment of CVS, (right) True negative results showing the absence of CVS

5.4 Discussion and Conclusions

We proposed using deep learning to accomplish objective assessment of the critical view of safety in laparoscopic cholecystectomy videos. The real-time performance of the proposed CNN shows the suitability of the deep learning model in online intra-operative applications. The limited results prove the potential of such methods in the assessment of the safety in a cholecystectomy procedure and can be used for training and educational purposes.

The detection of the CVS was accomplished by classifying all the frames of the video. Therefore, the model could be used for localizing the critical view of safety in a video as well. However, due to the very short duration of the occurrence of the CVS, the extreme imbalance has a detrimental effect on the performance of the model. In order to remedy this issue, the deep learning solution for identifying the achievement of the CVS needs to be trained and evaluated using short video clips that contain only the essential frames for detecting the CVS.

Due to the lack of a large set of videos for the assessment of the CVS and the required expertise, creating a well-annotated dataset to be used in a deep learning system is an expensive and slow process. Though the results of the proposed model indicates the

suitability of such systems in assessing the CVS even while using probably inaccurate criteria for labeling the data, further research needs to be performed on larger and labelled datasets before applying the findings of this chapter to the relevant clinical applications.

CHAPTER 6

CONCLUSION AND FUTURE DIRECTIONS

Recent advances in technology have facilitated the utilization of the rich content of laparoscopic videos, in order to potentially reduce human error and improve the outcomes of Laparoscopic surgery. In this dissertation, novel methods are proposed and investigated for the automated understanding of the recorded videos or streaming videos from a laparoscopic procedure. Laparoscopic cholecystectomy was chosen as a typical example and as the most common procedure, to validate the proposed methods. The following summarizes the main contributions of the dissertation:

- Monitoring the presence of surgical tools in online and offline modes, using the contextual information, i.e the pattern in the co-occurrence and the ordering of tools' usage.
- Identifying the surgical phases in order to analyze the surgical steps by considering the short-term and long-term temporal correlations in the videos.
- Segmenting the videos by extracting the boundaries of the surgical phases in an offline mode, using an attention-based model.
- Detecting and localizing the critical view of safety in cholecystectomy procedure

CNN was used as a powerful tool for learning the visual features in at least some parts of all the proposed models. Similarly, RNNs were adapted for capturing the coherence in the videos. Though the choice of the architectures has a considerable effect on the results (especially for CNN, which is the dominant component in most cases), our designs do not depend on a specific architecture, and relatively simpler architectures were chosen to highlight the improvements caused by the main novelties in each chapter.

Despite the initial similarities between all the problems that have been discussed in this dissertation, the requirements for achieving reasonable performance vary significantly. For instance, while the detection of the presence of surgical tools is a multilabel classification problem, surgical phase detection is multiclass, and detecting the CVS is binary classification. Furthermore, the impacts of spatial and temporal features are not the same in all the sub-problems of analyzing a surgical video. For example, the short-term consideration of the temporal dependencies are necessary for dealing with the motion blur or occlusion and was effectively captured by the proposed RCNN architecture, whereas in phase or CVS detection, a sliding window method which can capture a longer coherence was more suitable, as the transitions between the phases occur only a few times (five or six times for phase detection and only twice for CVS detection). Another common challenge in all the proposed methods was the class imbalance issue. The choices of the compensation methods were made based on the nature and the severity of the imbalance.

We obtained results which are better than the current for all the sub-problems that were investigated in this dissertation. To the best of our knowledge, this was the first attempt at solving the problems of finding the phase boundaries in laparoscopic videos and detecting the CVS in cholecystectomy procedure.

The proposed solution can be applied to any surgical procedure to potentially enhance the outcome as well as the training and the education of future surgeons. The models can potentially be used to design an integrated surgical video analysis system (SVAS). The encouraging results show the potential of SVAS to be used in various clinical applications.

The following section presents some suggestions for the directions of future research, based on the findings in this dissertation.

6.1 Future Directions

- ***Multi-task learning for SVAS***: The analysis of surgical videos often involves performing multiple tasks to capture useful information from the data. In this dissertation, we investigated tool and phase detections as the two essential components in surgical video understanding. Given the high correlation between the pattern in the tool usage and the phase, an integrated system can be designed by simultaneous training of a deep model to perform multiple tasks. An example of such a model is proposed in [104]. The critical ideas for designing a multi-task learning-based approach are, determining the parts of the network that are shared between the tasks and the strategy for training. We suggest sharing the spatiotemporal feature extraction with an RCNN and using the multi-task training approach described in chapter 3. Other considerations include the imbalance compensation and the post-processing steps.
- ***Tool classification based on their functionality***: The proposed model for detecting the tool usage in chapter 3 mostly relies on the appearances of the tools. Though the temporal dependencies were considered using the RCNN architecture, the unique functionalities of the tools were not explicitly taken into account, as the tools remain the same in all of the videos of the cholecystectomy dataset. In other words, due to the similarity of the tools in the training and validation sets, the spatial features were automatically given more priority in making the prediction on the presence of the tools. In order to check the effectiveness of a tool detection when a single tool like a Grasper can have multiple types, a new method needs to be designed to classify the video frames based on the functionality of the tools. A possible solution could be using the optical flow and trajectory along with the CNN, as described in section 2.2.2.

- ***Semi-supervised tool detection***: The fully supervised learning approach used in this dissertation requires collecting a large dataset that is annotated by the experts. Also, CNNs are known to have limitations in learning the affine transformations. In other words, training a CNN needs all the variants in sizes and orientations of the tools by either using an extensive data augmentation or introducing more data, especially for detecting the presence of the tools that have definite shapes. To tackle this issue Capsule networks have been recently proposed [77]. We believe if we use a sem-supervised learning approach based on Capsules, we might obtain high accuracy while using only a few labeled videos.
- ***Keyframes extraction***: The proposed method in chapter 4 for detecting the phase borders could potentially be applied to any application that requires extracting the keyframes. Examples include detecting the boundaries of any events of interest. The keyframe extraction can also be employed to summarize the videos for faster manual analysis or storage considerations.

Bibliography

[1] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, and Google Research. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.

[2] Sajjad Abdoli, Patrick Cardinal, and Alessandro Lameiras Koerich. End-to-End Environmental Sound Classification using a 1D Convolutional Neural Network. 4 2019.

[3] Tamer Abdulbaki Alshirbaji, Nour Aldeen Jalal, and Knut Möller. Surgical Tool Classification in Laparoscopic Videos Using Convolutional Neural Network. *Current Directions in Biomedical Engineering*, 4(1):407–410, 9 2018.

[4] Hassan Al Hajj, Mathieu Lamard, Pierre-Henri Conze, Batrice Cochener, and Gwenol Quellec. Monitoring tool usage in surgery videos using boosted convolutional and recurrent neural networks. *Medical Image Analysis*, 47:203–218, 7 2018.

[5] Hassan Al Hajj, Mathieu Lamard, Pierre-Henri Conze, Soumali Roychowdhury, Xiaowei Hu, Gabija Maršalkaitė, Odysseas Zisimopoulos, Muneer Ahmad Dedmari,

Fenqiang Zhao, Jonas Prellberg, Manish Sahu, Adrian Galdran, Teresa Araújo, Duc My Vo, Chandan Panda, Navdeep Dahiya, Satoshi Kondo, Zhengbing Bian, Arash Vahdat, Jonas Bialopetravičius, Evangello Flouty, Chenhui Qiu, Sabrina Dill, Anirban Mukhopadhyay, Pedro Costa, Guilherme Aresta, Senthil Ramamurthy, Sang-Woong Lee, Aurlio Campilho, Stefan Zachow, Shunren Xia, Sailesh Conjeti, Danail Stoyanov, Jogundas Armaitis, Pheng-Ann Heng, William G. Macready, Batrice Cochener, and Gwenol Quellec. CATARACTS: Challenge on automatic tool annotation for cataRACT surgery. *Medical Image Analysis*, 52:24–41, 2 2019.

[6] M. Allan, S. Ourselin, D. J. Hawkes, J. D. Kelly, and D. Stoyanov. 3-D Pose Estimation of Articulated Instruments in Robotic Minimally Invasive Surgery. *IEEE Transactions on Medical Imaging*, 37(5):1204–1213, 5 2018.

[7] M. Allan, S. Ourselin, S. Thompson, D. J. Hawkes, J. Kelly, and D. Stoyanov. Toward Detection and Localization of Instruments in Minimally Invasive Surgery. *IEEE Transactions on Biomedical Engineering*, 60(4):1050–1058, 4 2013.

[8] Sana Amanat, Muhammad Idrees, Muhammad Usman Ghani Khan, Zahoor Rehman, Hangbae Chang, Irfan Mehmood, and Sung Wook Baik. Video Retrieval System for Meniscal Surgery to Improve Health Care Services. *Journal of Sensors*, 2018:1–10, 6 2018.

[9] Maria Antico, Fumio Sasazawa, Liao Wu, Anjali Jaiprakash, Jonathan Roberts, Ross Crawford, Ajay K. Pandey, and Davide Fontanarosa. Ultrasound guidance in minimally invasive robotic procedures. *Medical Image Analysis*, 54:149–167, 5 2019.

[10] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. 7 2016.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. 9 2014.

[12] Garth H Ballantyne. The pitfalls of laparoscopic surgery: challenges for robotics and telerobotic surgery. *Surgical laparoscopy, endoscopy & percutaneous techniques*, 12(1):1–5, 2 2002.

[13] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 1171–1179, 2015.

[14] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 3 1994.

[15] David Bouget, Max Allan, Danail Stoyanov, and Pierre Jannin. Vision-based and marker-less surgical tool detection and tracking: a review of the literature. *Medical Image Analysis*, 35:633–654, 1 2017.

[16] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *CoRR*, abs/1710.0:1–23, 2017.

[17] K. Tim Buddingh, Rinse K. Weersma, Rolf A.J. Savenije, Gooitzen M. van Dam, and Vincent B. Nieuwenhuijs. Lower Rate of Major Bile Duct Injury and Increased Intraoperative Management of Common Bile Duct Stones after Implementation of Routine Intraoperative Cholangiography. *Journal of the American College of Surgeons*, 213(2):267–274, 8 2011.

[18] C Olah. Understanding LSTM Networks – colah's blog.

[19] Haoye Cai, Chunyan Bai, Yu-Wing Tai, and Chi-Keung Tang. Deep Video Generation, Prediction and Completion of Human Action Sequences. pages 374–390. Springer, Cham, 9 2018.

[20] Gunnar Carlsson, Tigran Ishkhanov, Vin de Silva, and Afra Zomorodian. On the Local Behavior of Spaces of Natural Images. *International Journal of Computer Vision*, 76(1):1–12, 1 2008.

[21] Francisco Charte, Antonio J. Rivera, Mara J. del Jesus, and Francisco Herrera. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163:3–16, 9 2015.

[22] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports*, 8(1):6085, 12 2018.

[23] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 17241734, 6 2014.

[24] Shanley B. Deal, Dimitrios Stefanidis, Dana Telem, Robert D. Fanelli, Marian McDonald, Michael Ujiki, L. Michael Brunt, and Adnan A. Alseidi. Evaluation of crowd-sourced assessment of the critical view of safety in laparoscopic cholecystectomy. *Surgical Endoscopy*, 31(12):5094–5100, 12 2017.

[25] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-Term Recurrent Con-

volutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691, 4 2017.

[26] Timothy Dozat. Incorporating Nesterov Momentum into Adam. In *ICLR 2016 workshop*, 2 2016.

[27] Xiaofei Du, Maximilian Allan, Alessio Dore, Sebastien Ourselin, David Hawkes, John D. Kelly, and Danail Stoyanov. Combined 2D and 3D tracking of surgical instruments for minimally invasive and robotic-assisted surgery. *International Journal of Computer Assisted Radiology and Surgery*, 11(6):1109–1119, 6 2016.

[28] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[29] Robert Elfring, Matas de la Fuente, and Klaus Radermacher. Assessment of optical localizer accuracy for computer aided surgery systems. *Computer Aided Surgery*, 15(1-3):1–12, 2 2010.

[30] Germain Forestier, Florent Lalys, Laurent Riffaud, D. Louis Collins, Jurgen Meixensberger, Shafik N. Wassef, Thomas Neumuth, Benoit Goulet, and Pierre Jannin. Multi-site study of surgical practice in neurosurgery based on surgical process models. *Journal of Biomedical Informatics*, 46(5):822–829, 10 2013.

[31] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 9 2018.

[32] Eva Gibaja and Sebastin Ventura. A Tutorial on Multilabel Learning. *ACM Computing Surveys*, 47(3):1–38, 4 2015.

[33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*.

[34] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[35] Kazuyuki Hara, Daisuke Saitoh, and Hayaru Shouno. Analysis of Dropout Learning Regarded as Ensemble Learning. pages 72–79. Springer, Cham, 2016.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 6 2016.

[37] Sepp Hochreiter and Jrgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[38] Xiaowei Hu, Lequan Yu, Hao Chen, Jing Qin, and Pheng-Ann Heng. AGNet: Attention-Guided Network for Surgical Tool Presence Detection. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 186–194. Springer, Cham, 2017.

[39] D H HUBEL and T N WIESEL. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–91, 10 1959.

[40] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *the 32nd International Conference on Machine Learning, PMLR*, pages 448–456, 6 2015.

[41] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 1 2013.

[42] Amy Jin, Serena Yeung, Jeffrey Jopling, Jonathan Krause, Dan Azagury, Arnold Milstein, and Li Fei-Fei. Tool Detection and Operative Skill Assessment in Surgical Videos Using Region-Based Convolutional Neural Networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 691–699. IEEE, 3 2018.

[43] Yueming Jin, Qi Dou, Hao Chen, Lequan Yu, and Pheng-Ann Heng. EndoRCN : Recurrent Convolutional Networks for Recognition of Surgical Workflow in Cholecystectomy Procedure Video. pages 2–5.

[44] Yueming Jin, Qi Dou, Hao Chen, Lequan Yu, Jing Qin, Chi-Wing Fu, and Pheng-Ann Heng. SV-RCNet: Workflow Recognition From Surgical Videos Using Recurrent Convolutional Network. *IEEE Transactions on Medical Imaging*, 37(5):1114–1126, 5 2018.

[45] Jakub Kaczynski and Joanna Hilton. A gallbladder with the "hidden cystic duct": A brief overview of various surgical techniques of the Calot's triangle dissection. *Interventional Medicine & Applied Science*, 7(1):42–45, 2015.

[46] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient Neural Audio Synthesis. 2 2018.

[47] Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. Object Detection in Videos with Tubelet Proposal Networks. In

2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 889–897. IEEE, 7 2017.

[48] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Fei Fei Li. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[49] Frederik Keus, Jeroen de Jong, H G Gooszen, and C JHM Laarhoven. Laparoscopic versus open cholecystectomy for patients with symptomatic cholecystolithiasis. *Cochrane Database of Systematic Reviews*, (4):CD006231, 10 2006.

[50] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 12 2014.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances In Neural Information Processing Systems*, pages 1097–1105, 2012.

[52] Florent Lalys and Pierre Jannin. Surgical process modelling: a review. *International Journal of Computer Assisted Radiology and Surgery*, 9(3):495–511, 5 2014.

[53] Alex M. Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. Professor Forcing: A New Algorithm for Training Recurrent Networks. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 4601–4609, 2016.

[54] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[55] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015.

[56] Huayu Li, Martin Renqiang Min, Yong Ge, and Asim Kadav. A Context-aware Attention Network for Interactive Question Answering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, pages 927–935, New York, New York, USA, 2017. ACM Press.

[57] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 12 2017.

[58] G S Litynski. Erich Mühe and the rejection of laparoscopic cholecystectomy (1985): a surgeon ahead of his time. *JSLS : Journal of the Society of Laparoendoscopic Surgeons*, 2(4):341–6, 1998.

[59] Constantinos Loukas. Video content analysis of surgical procedures. *Surgical Endoscopy*, 32(2):553–568, 2 2018.

[60] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. 8 2015.

[61] Kaustuv Mishra, Rachana Sathish, and Debdoot Sheet. Learning Latent Temporal Connectionism of Deep Residual Visual Abstractions for Identifying Surgical Tools in Laparoscopy Procedures. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2233–2240. IEEE, 7 2017.

[62] Paulo Mota, Nuno Carvalho, Emanuel Carvalho-Dias, Manuel João Costa, Jorge Correia-Pinto, and Estevo Lima. Video-Based Surgical Learning: Improving Trainee

Education and Preparation for Surgery. *Journal of Surgical Education*, 75(3):828–835, 5 2018.

[63] Sanjay Nagral. Anatomy relevant to cholecystectomy. Technical Report 2, 2005.

[64] Babak Namazi, Ganesh Sankaranarayanan, and Venkat Devarajan. Automatic Detection of Surgical Phases in Laparoscopic Videos. In *Proceedings on the International Conference in Artificial Intelligence (ICAI)*, pages 124–130, 2018.

[65] Chinedu Innocent Nwoye, Didier Mutter, Jacques Marescaux, and Nicolas Padoy. Weakly supervised convolutional LSTM approach for tool tracking in laparoscopic videos. *International Journal of Computer Assisted Radiology and Surgery*, 14(6):1059–1067, 6 2019.

[66] Nicolas Padoy, Tobias Blum, Seyed-Ahmad Ahmadi, Hubertus Feussner, and Marie-Odile Berger. Statistical modeling and recognition of surgical workflow. *Medical Image Analysis*, 16(3):632–641, 4 2012.

[67] Anibal Pedraza, Jaime Gallego, Samuel Lopez, Lucia Gonzalez, Arvydas Laurinavicius, and Gloria Bueno. Glomerulus Classification with Convolutional Neural Networks. pages 839–849. Springer, Cham, 2017.

[68] Gabriel Pereyra, George Tucker, Jan Chorowski, ukasz Kaiser, and Geoffrey Hinton. Regularizing Neural Networks by Penalizing Confident Output Distributions. 1 2017.

[69] Jonas Prellberg and Oliver Kramer. Multi-label Classification of Surgical Tools with Convolutional Neural Networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 7 2018.

[70] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schluter, Shuo-Yiin Chang, and Tara Sainath. Deep Learning for Audio Signal Processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, 5 2019.

[71] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1 1999.

[72] Jesse Read, Bernhard Pfahringer, Geoff Holmes, Eibe Frank, Carla J Brodley Read, B Pfahringer, G Holmes, E Frank, and J Read. Classifier chains for multi-label classification. *Mach Learn*, 85:333–359, 2011.

[73] Austin Reiter, Peter K. Allen, and Tao Zhao. Feature Classification for Tracking Articulated Surgical Tools. pages 592–600. Springer, Berlin, Heidelberg, 2012.

[74] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[75] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 10 1986.

[76] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 12 2015.

[77] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic Routing Between Capsules. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 3856–3866, 2017.

[78] Manish Sahu, Anirban Mukhopadhyay, Angelika Szengel, and Stefan Zachow. Tool and Phase recognition using contextual CNN features. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 186–194, 2016.

[79] Manish Sahu, Anirban Mukhopadhyay, Angelika Szengel, and Stefan Zachow. Addressing multi-label imbalance problem of surgical tool detection using CNN. *International Journal of Computer Assisted Radiology and Surgery*, 12(6):1013–1020, 6 2017.

[80] Tim Salimans and Durk P. Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages 901–909, 2016.

[81] Sampurna Roy. Normal Anatomy of the Gallbladder.

[82] Dominic E. Sanford and Steven M. Strasberg. A Simple Effective Method for Generation of a Permanent Record of the Critical View of Safety during Laparoscopic Cholecystectomy by Intraoperative Doublet Photography. *Journal of the American College of Surgeons*, 218(2):170–178, 2 2014.

[83] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How Does Batch Normalization Help Optimization? In *Advances in Neural Information Processing Systems 31 (NIPS 2018)*, pages 2483–2493, 2018.

[84] Duygu Sarikaya, Jason J. Corso, and Khurshid A. Guru. Detection and Localization of Robotic Tools in Robot-Assisted Surgery Videos Using Deep Neural Networks for Region Proposal and Detection. *IEEE Transactions on Medical Imaging*, 36(7):1542–1549, 7 2017.

[85] Christopher P. Scally, Oliver A. Varban, Arthur M. Carlin, John D. Birkmeyer, and Justin B. Dimick. Video Ratings of Surgical Skill and Late Outcomes of Bariatric Surgery. *JAMA Surgery*, 151(6):e160428, 6 2016.

[86] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[87] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626. IEEE, 10 2017.

[88] Daniel M. Shabanzadeh and Lars T. Sørensen. Laparoscopic Surgery Compared With Open Surgery Decreases Surgical Site Infection in Obese Patients. *Annals of Surgery*, 256(6):934–945, 12 2012.

[89] Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. Neural Abstractive Text Summarization with Sequence-to-Sequence Models. 12 2018.

[90] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1049–1058. IEEE, 6 2016.

[91] Karen Simonyan and Andrew Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos, 2014.

[92] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. 9 2014.

[93] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Technical report, 2014.

[94] S M Strasberg, M Hertl, and N J Soper. An analysis of the problem of biliary injury during laparoscopic cholecystectomy. *Journal of the American College of Surgeons*, 180(1):101–25, 1 1995.

[95] Steven M. Strasberg and L. Michael Brunt. Rationale and Use of the Critical View of Safety in Laparoscopic Cholecystectomy. *Journal of the American College of Surgeons*, 211(1):132–138, 7 2010.

[96] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112, 2014.

[97] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2 2017.

[98] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 1–9, 2015.

[99] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826. IEEE, 6 2016.

[100] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497. IEEE, 12 2015.

[101] Abhishek Trehan, Ashton Barnett-Vanes, Matthew J Carty, Peter McCulloch, and Mahiben Maruthappu. The impact of feedback of intraoperative technical performance in surgery: a systematic review. *BMJ open*, 5(6):e006759, 6 2015.

[102] Zhigang Tu, Wei Xie, Dejun Zhang, Ronald Poppe, Remco C. Veltkamp, Baoxin Li, and Junsong Yuan. A survey of variational and CNN-based optical flow techniques. *Signal Processing: Image Communication*, 72:9–24, 3 2019.

[103] Andru P. Twinanda, Didier Mutter, Jacques Marescaux, Michel de Mathelin, and Nicolas Padoy. Single- and Multi-Task Architectures for Tool Presence Detection Challenge at M2CAI 2016. 10 2016.

[104] Andru P. Twinanda, Sherif Shehata, Didier Mutter, Jacques Marescaux, Michel de Mathelin, and Nicolas Padoy. EndoNet: A Deep Architecture for Recognition Tasks on Laparoscopic Videos. *IEEE Transactions on Medical Imaging*, 36(1):86–97, 1 2017.

[105] Andru Putra Twinanda, Nicolas Padoy, Mrs Jocelyne Troccaz, and Gregory Hager. *Vision-based Approaches for Surgical Activity Recognition Using Laparoscopic and RBGD Videos*. PhD thesis, 2017.

[106] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. 7 2016.

[107] Armine Vardazaryan, Didier Mutter, Jacques Marescaux, and Nicolas Padoy. Weakly-Supervised Learning for Tool Localization in Laparoscopic Videos. 6 2018.

[108] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008, 2017.

[109] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer Networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 2692–2700, 2015.

[110] Sheng Wang, Ashwin Raju, and Junzhou Huang. Deep learning based multi-label classification for surgical tool presence detection in laparoscopic videos. In *Proceedings - International Symposium on Biomedical Imaging*, pages 620–623, 2017.

[111] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[112] Daniel Wesierski and Anna Jezierska. Instrument detection and pose estimation with rigid part mixtures model in video-assisted surgeries. *Medical Image Analysis*, 46:244–265, 5 2018.

[113] Ronald J. Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, 6 1989.

[114] Yuxin Wu and Kaiming He. Group Normalization. In *The European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[115] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *32nd International Conference*

on Machine Learning, ICML, pages 2048–2057. International Machine Learning Society (IMLS), 2015.

[116] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 8 2018.

[117] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. 12 2012.

[118] Aneeq Zia, Daniel Castro, and Irfan Essa. Fine-tuning Deep Architectures for Surgical Tool Detection. In *Workshop and Challenges on Modeling and Monitoring of Computer Assisted Interventions (M2CAI)*, 2016.

[119] Aneeq Zia, Yachna Sharma, Vinay Bettadapura, Eric L. Sarin, Thomas Ploetz, Mark A. Clements, and Irfan Essa. Automated video-based assessment of surgical skills for training and evaluation in medical schools. *International Journal of Computer Assisted Radiology and Surgery*, 11(9):1623–1636, 9 2016.

[120] Odysseas Zisimopoulos, Evangello Flouty, Mark Stacey, Sam Muscroft, Petros Giataganas, Jean Nehme, Andre Chow, and Danail Stoyanov. Can surgical simulation be used to train detection and classification of neural networks? *Healthcare technology letters*, 4(5):216–222, 10 2017.