CS230

Content Recognition in Surgical Videos

Senthil Vel Gunasekaran sguna@stanford.edu
Madhusudan Hegde mrhegde@stanford.edu
Nithin Akkati nakkati@stanford.edu

Abstract

There is a huge interest in automatic surgical work-flow detection and challenges have not been addressed properly. Advances in Computer Vision and Deep Neural Network are very promising to address the challenges and we explored optimization of CNN+LSTM model to detect work-flow from surgical videos. We also developed tool detection model to evaluate the impact of tool recognition in surgical phase detections. We achieved a validation accuracy of 88% by CNN+LSTM optimization and by applying prior knowledge of surgical phases to improve posterior probability of detection.

1 Introduction

Each year millions of in-patient operations are performed in the United States, however, studies [4] have shown that lack of surgical flow recognition results in high levels of stress causing risks and delays, as well high surgical expenses [5]. So, context-aware system that can monitor different phases of a surgery has become a key component of modern operating room. Such intelligent system can also be used for surgeon training and skill evaluation. Most surgeries in the US are recorded and Computer Vision can be used to analyze surgical videos. The surgical phase detection is an activity detection problem where visual features can be combined with temporal information to discriminate surgical phases. However, it is challenging because visual features are not unique and there is large deviation in duration of each phases. Also, transition between the phases is not well defined and several visual artifacts are introduced during surgery. We built a CNN+LSTM model to address this challenge and developed tool detection model that can be used to further improve the work-flow detection. The complete code base for this project is also available for reference[15][16].

2 Related work

Phase Detection with Sensors: Sensor based models mentioned in [7], [8] [9] required manual annotation of signals, cumbersome to use and failed to generalize surgical work flows.

Endonet: This CNN model relied on video data to predict the phases [3] [5] but models based only visual discrimination and without temporal information perform poorly.

SV-RCNet[1]: Used end-to-end training on Resnet50+LSTM model and achieved an accuracy of 89% on cholec80 data. They also implemented prior knowledge inference to improve phase predictions.

Tool detection: Used Fast-RCNN for spatial detection of surgical tools and VGG16 for classification to achieve state of the art performance of 81%.
Our surgical flow detection work is inspired by SV-RCNet and we used different CNN model and LSTM architecture to achieve similar performance. The tool detection model is inspired by [13] and we achieved similar performance by using Resnet50.

3 Dataset and Features

3.1 Dataset Statistics:

We used Cholec80 dataset with 80 videos of cholecystectomy procedure. We split our dataset into 27 videos for training, 14 each for validation and testing – with split ratio of 50/25/25. There are 7 phases of

,

surgical activity with transitions (prior knowledge) as shown below. We used data augmentation to handle heavily unbalanced data set.

Tool detection: We use total of 1250 images with bounding boxes on tools as our train set, 250 different images for train and test set each, we have 7 tools as different classes to predict. We use similar data augmentation as below to fix unbalanced data set.

ID	Surgical Phase	Duration(min)
P0	Preparation	2.2
P1	Calot Dissection	10.4
P2	Clipping and Cutting	2.8
P3	Gallbladder Dissection	7.5
P4	Gallbladder Packaging	1.8
P5	Cleaning Coagulation	1.9
P6	Gallbladder Retraction	4.5

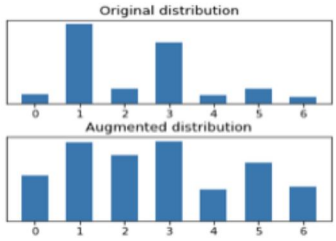

Figure 1: Dataset Statistics

3.2 Data Preprocessing:

The video was converted to image sequence at 5 FPS to capture subtilities of tool movements. The images are grouped (time step) into sequences of 5, 15 and 25 images, corresponding to 1sec, 3sec and 5 seconds and we found that surgical activity is better represented by time step of 24 (5 sec) in our experiments. The original resolution of 1920 x 1080 was an overkill for spatial representation and images were downsized to 224 x 224 RGB format to retain the color information required for visual discrimination. Due to human factors, the images corresponding to same tool activity showed rotations in different videos as seen in the images below. So, the data set was further augmented by random horizontal flips

Figure 2: Data Augumentation

4 Methods

4.1 Model Implementation:

We used the functional API[11] of Keras with Tensorflow backend as it was easy to put together multiple models. We used AWS instance with Tesla V100 GPUs without multi-GPU optimization as it impacted convergence of our results.Given the large feature size (8, 15, 224, 224, 3), we implemented generator functions to feed the optimizer. Several callback functions were implemented to capture activation maps and training statistics.

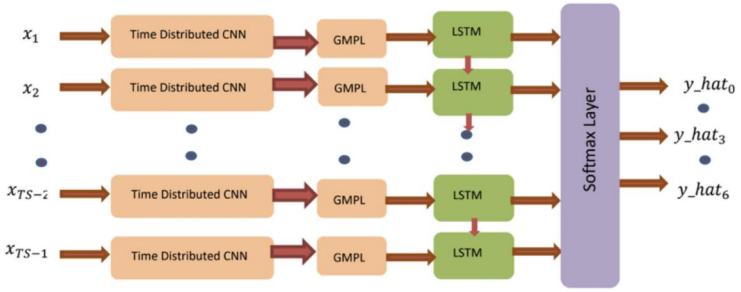

Figure 3: Model Architecture

4.2 Model:

The CNN model is only capable of transforming single image at a time and need to repeat this operation across multiple images to allow the LSTM to build up internal state and update weights using back propagation across a sequence of the internal vector representations of input images.This was achieved by wrapping the entire CNN input model in a TimeDistributed layer in Keras implementation. The softmax layer at the CNN output is replaced by Global Avg Pool Layer (GMPL) and fed into LSTM. The TimeDistributed CNN and LSTM blocks in Figure 3 are different instances of models with same weights.

4.3 Tool model:

The model takes image input and predicts tools presence, localization and probability. Fast RCNN[12][13][14] has two networks, region proposal network(RPN) for generating region proposals and a CNN network to detect objects using these proposals. We tried two models - VGG and ResNet50, and ResNet gave best results. The model was trained separately and used in test phase to evaluate tool importance.

Training Method	Advantages	Disadvantages
Train CNN and LSTM Separately	• Hyper-parameters can be chosen independently as separate loss functions are used • Computationally less expensive	• Sub-optimal due to significant intra-class variance and limited interclass variance of visual features
End to End CNN + LSTM training	• The LSTM error is backpropagated to CNN and CNN training is helped by sequence discrimination of LSTM	• Restriction on hyper parameter selection as both the models share same optimizer/loss function in Keras
Stateful LSTM	• Captures correlations across time steps • Useful when small batch_size is used as Keras resets the LSTM internal states every batch_size	• Care must be taken to maintain temporal information in the input data (no shuffling). Otherwise will degrade the performance

4.4 Loss Formulation:

We used multi-class cross entropy for loss function as given below. The loss function is computed for number of classes, C=7, timestep of Ts = 25 and mini batch_size B = 8. The N is number of mini batches with N = int(num_images/(Ts*B)). We used L2 regularization for LSTM layer with = 0.01. The W captures all the weights inside LSTM cells. The CWi is the class_weights.

$$Loss(x) = -1/N \sum_{n=1}^{N} 1/B \sum_{m=1}^{B} 1/Ts \sum_{t=1}^{Ts} \sum_{i=1}^{C} CW_i * y_i * log\left(y_{hat_i}\right), \quad Loss(x) = Loss(x) + \lambda * W^2.$$

5 Experiments/Results/Discussion

Due to limited computational resource, The improvements were made incrementally after analyzing the results and exploring options that would take the performance to next level. The high dimensional image data was normalized as we observed early training saturation with base model.We did not have to check for vanishing/exploding gradients throughout our experiments as training accuracy was always good.

5.1 Hyperparameter Tuning:

The hyperparameter optimization is a big challenge for multi-model training especially when transfer learning is applied to only one of the models and we found that VGG16 influenced our learning rate. We tried learning rate of 0.001, 0.0001, 0.00005, 0.00001 and overfitting was mitigated with learning rate of 0.00001, rate decay of 0.004. The variance issue was not fully resolved as we saw a difference of over 10% between training and validation accuracy. In combined CNN+LSTM model, the number of samples in each batch gets multiplied by the time step value and over-fitting problem was seen with batch size larger than 16. The batch size selection was a trade-off between CNN and LSTM performances and we chose batch size of 8 (8x25 samples per batch). The SGD is becoming popular these days because of better generalization power but we did not see any improvement with SGD. Finally, we chose Nadam (Adam with Nesterov) optimizer [10] to avoid local minima. The optimizer was trained with class_weights (computed by sklearn utility) to address unbalanced data and it bumped up the validation accuracy.All our hyperparameter tuning was based to training set and early stopping was based on validation set.

5.2 Model Optimization:

We started with one hidden layer of size 4096 (number of LSTM cells) and achieved training accuracy of 95% while validation accuracy was below 72%. The validation accuracy was improved to 75% by adding

	VGG Only (a)			VGG LSTM2 (b)			VGG LSTM Class Weights (c)			Final LSTM Model (d)		
	Precision	recall	f1-score	Precision	recall	f1-score	Precision	recall	f1-score	Precision	recall	f1-score
Preparation	0.53	0.8	0.64	0.41	0.32	0.36	0.59	0.68	0.63	0.76	0.66	0.71
CalotTriangleDissection	0.92	0.36	0.51	0.87	0.92	0.89	0.92	0.94	0.93	0.91	0.95	0.93
ClippingCutting	0.17	0.85	0.28	0.87	0.73	0.79	0.79	0.89	0.84	0.91	0.78	0.84
GallbladderDissection	0.82	0.37	0.51	0.84	0.92	0.88	0.88	0.91	0.89	0.92	0.88	0.9
GallbladderPackaging	0.27	0.83	0.41	0.93	0.63	0.75	0.66	0.78	0.71	0.9	0.79	0.84
CleaningCoagulation	0.32	0.68	0.43	0.84	0.69	0.76	0.76	0.56	0.64	0.76	0.88	0.81
GallbladderRetraction	0.78	0.33	0.46	0.5	0.38	0.43	0.65	0.49	0.56	0.7	0.59	0.64
Overall	0.54	0.6	0.46	0.75	0.66	0.83	0.75	0.75	0.85	0.88	0.88	0.88

Dropout layers at the output of CNN LSTM and L2 kernel regularizer to the LSTM layer. We did not try regularization on VGG model as CNN already used small size kernels. We tried of hidden layer size of 2048, 1024, 512 256 and size 2048 was compromise between too few (underfitting) and too many (overfitting) neurons in the hidden layer. We experimented with second hidden layer (stacked LSTM) as additional layers help to capture complex structures with arbitrary decision boundaries. We did not get time to optimize stacked LSTM and it was found that stacked LSTM performed better than single layer LSTM with identical hyperparameters. The surgical flow has a Markov structure with certain transition probabilities between the phases and accuracy of predictions can be further improved by exploiting this structure. The stateless LSTM does not capture the transition between the phases as the internal states get reset during mini-batch training. We implemented stateful LSTM where model is trained with entire video file without shuffling. We did not have time to complete the optimization of stateful LSTM. Instead we implemented prior knowledge application (PKA) on top of LSTM layer to further improve the accuracy. This is a basic PKA algorithm that exploits knowledge of initial phase of surgical flow (Preparation), allowed transition between the phases and minimum duration of each phases to remove false detections. The visual inspection of images from different camera sources showed some amount of covariance shift (i.e color difference) and we tried to add batch normalization at the output of LSTM. However, batch normalization degraded the performance and did not get time to analyze this issue.

5.3 Error Analysis :

We relied on loss function, confusion matrix and activation maps from CNN to analyze the performance of our models. The global average pooling layer at the output of CNN enabled us to generate activation maps. The validation loss for three models is shown below and end to end model has best performance.The activation map helped us to understand which phases had false detection at the CNN output and we used this information to augment the training set data and to tune class_weights. In the example below, the tool used in Preparation was not seen in the activation map and it was detected as Calot Dissection phase. Finally, the confusion matrix was used to compare performance of multi-label classification. The below

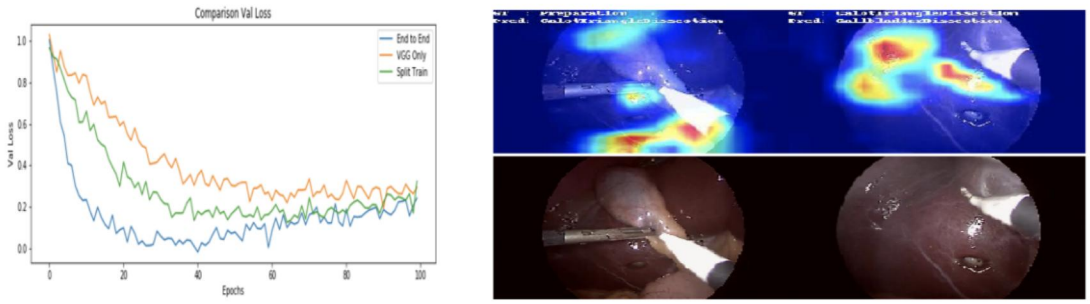

Figure 4: Loss plots

matrix shows comparison of 4 models. The same number of evaluation videos are used for generating the matrix and numbers in the matrix gets scaled by time steps in LSTM implementation. The number of time steps is 1 for VGG only and 25 for the final CNN+LSTM model.

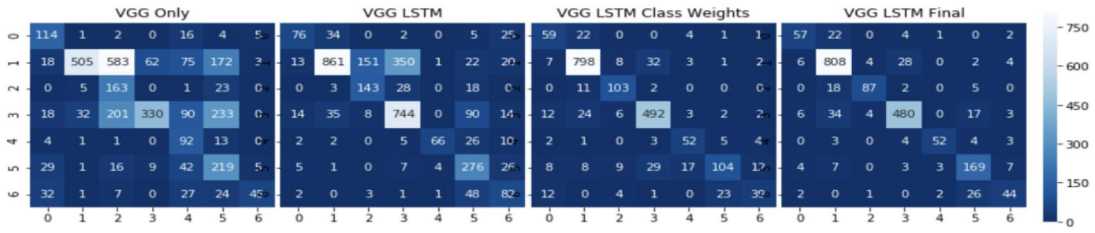

Figure 5: Confusion matrix

5.4 Tool detection results:

We achieved 80% test accuracy with Fast-RCNN, Figure 5 shows tools that are harder to detect due to presence of gas and light at frame level and mispredicted gas with a bag vs when there is no gas and tool was correctly classified.

By using this model with CNN+LSTM, we see improvement in F1 scores of multiple phases confirming tool information to be important, but currently the weights are hand tuned using expert knowledge of presence of tools in each phase.

	Baseline CNN+LSTM				Baseline +Tool Detection		
	Count	Precision	Recall	F1	Precision	Recall	F1
Preparation	128	0.90	0.59	0.72	0.66	0.88	0.75
Calot Training Dissection	500	0.92	0.82	0.87	0.94	0.82	0.87
Clipping Cutting	68	0.54	0.44	0.48	0.62	0.44	0.52
Gallbladder Dissection	370	0.77	0.93	0.85	0.79	0.93	0.85
Gallbladder packaging	80	0.62	0.86	0.72	0.83	0.67	0.74
Cleaning Coagulation	58	0.87	0.96	0.91	0.89	0.93	0.91
Gallbladder Retraction	25	0.40	0.40	0.40	0.33	0.30	0.3
Overall				0.81			0.82

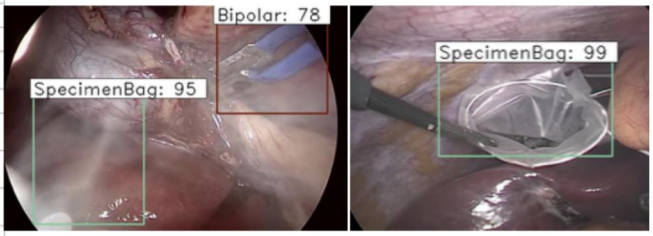

Figure 6: Tool Results

6 Conclusion/Future Work

We could successfully train CNN+LSTM model to achieve high accuracy of 88%. We summarize our findings below.

- End to end training of CNN+LSTM model with data augmentation and hyperparameter tuning enabled to achieve this accuracy.
- The performance of both stateful and stateless LSTM are similar for larger batch sizes and stateful LSTM with un-shuffled input data performs better with weights from stateless LSTM.
- We have got on par state of the art performance of 80% accuracy with the current tool detection model. We are using human knowledge to come up with the weights to help with the confidence of phase detection. We have shown tool knowledge being helpful in phase detection

We experimented with lot of things in this project and could not incorporate everything in our final model. The summary of future work is given below.

- Even with high accuracy, we saw overfitting problem in our model and it was avoided by early stopping. We would like to understand/optimize stacked LSTM and stateful LSTM models to solve overfitting problem.
- We would like to do end to end training of tool model with CNN+LSTM, for networks to learn the relationship and weights during training instead of hand engineering.
- Consider a ML platform that allows separate hyperparameter tuning for each model in a multi-model architecture

7 Contributions

All members contributed equally to the project with, Madhu writing code for CNN+LSTM models, Nithin working on tool detection code implementation and Senthil working on visualization and evaluation module devolopment.
We would like to thank Aarti and Hossein for their guidence on this project.

References

[1] Y. Jin et al.,*"SV-RCNet: Workflow Recognition from Surgical Videos Using Recurrent Convolutional Network,"* in IEEE Transactions on Medical Imaging, vol. 37, no. 5, pp. 11141126, May 2018.
[2] O. Dergachyova, D. Bouget, A. Huaulmé, X. Morandi, and P. Jannin, "Automatic data-driven real-time segmentation and recognition of surgical workflow," Int. J. Comput. Assist. Radiol. Surgery, vol. 11, no. 6,
[3] A. P. Twinanda, S. Shehata, D. Mutter, J. Marescaux, M. de Mathelin and N. Padoy, "EndoNet: A Deep Architecture for Recognition Tasks on Laparoscopic Videos," in IEEE Transactions on Medical Imaging, vol. 36, no. 1, pp. 86-97, Jan. 2017.
[4] Sonal Arora, Louise Hull, Nick Sevdalis, Tanya Tierney, Debra Nestel, Maria Woloshynowych, Ara Darzi et Roger Kneebone. Factors compromising safety in surgery: stressful events in the operating room. The American Journal of Surgery, vol. 199, no. 1, pages 60–65, 2010. Cited page 4.
[5] Alex Macario. What does one minute of operating room time cost? Journal of clinical anesthesia, vol. 22, no. 4, pages 233–236, 2010. Cited page 4.
[6] A. Jin et al., "Tool Detection and Operative Skill Assessment in Surgical Videos Using Region-Based Convolutional Neural Networks," 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, 2018, pp. 691-699.
[7] N. Padoy, T. Blum, S.-A. Ahmadi, H. Feussner, M.-O. Berger, and N. Navab, "Statistical modeling and recognition of surgical workflow," Med. Image Anal., vol. 16, no. 3, pp. 632–641, 2012.
[8] J. E. Bardram, A. Doryab, R. M. Jensen, P. M. Lange, K. L. Nielsen, and S. T. Petersen, "Phase recognition during surgical procedures using embedded and body-worn sensors," in Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom), Mar. 2011, pp. 45–53.
[9] M. S. Holden et al., "Feasibility of real-time workflow segmentation for tracked needle interventions," IEEE Trans. Biomed. Eng., vol. 61, no. 6, pp. 1720–1728, Jun. 2014.
[10] Incorporating Nesterov Momentum into Adam by Timothy Dozat. Stanford CS229 2015 Project
[11] Pretrained weights from Imagenet for *VGG16, Resnet-50*
[12] Baseline used for tool detection code *:https://github.com/rbgirshick/fast-rcnn*.
[13] Ross Girshick (2015) *Fast R-CNN* Microsoft Research.
[14] Amy Jin, Serena Yeung, Jeffrey Jopling, Jonathan Krause, Dan Azagury, Arnold Milstein, and Li Fei-Fei(2017). *Tool Detection and Operative Skill Assessment in Surgical Videos Using Region-Based Convolutional Neural Networks* Stanford University.
[15] Master branch for CNN+LSTM. *"https://github.com/madhuhegde/C230_CNN_LSTM.git"*.
[16] Tool Detection branch for CNN+LSTM. *"https://github.com/madhuhegde/C230_CNN_LSTM.git"*.