Hernia Surgery Phase Prediction

E6691.2022Spring.SURG.report.an3078.bmh2168.wab2138
Columbia University

Abstract

*This report summarizes the findings of an attempt to detect phases of hernia surgeries through videos. The work has been based on the results of previous papers **[1] and [2]** as well as some of our own intuitions.*

1. Introduction

A small dataset of hernia surgeries has been provided by expert annotators. The latter consists of videos containing one frame per second. A number of different deep learning models and processing techniques have been implemented and tested on this dataset with varying success. Models that have been developed for other surgery phase-detection problems did not work as well as expected. More traditional CNN networks, coupled with some post-processing, achieved better results and faster training time.

2. Summary of previous works

2.1 CNNs with Time Smoothing and HMM

After a first pass of the data through a CNN, one of the ideas of Rémi Cadène et al.'s paper was to take the output probabilities of each class, and perform temporal smoothing on them to get more accurate predictions [1]. We can do that either by averaging the probabilities in a given time window (e.g. 10 frames), or running our output through a Hidden Markov Model. The idea would be that the real temporal distribution of the labels in the video follow a Markov distribution, but that we are only able to observe a 'fogged' signal of it: our predictions.

2.2 CNN-LSTMs to Predict Surgery Duration

In their paper, Aksamentov I, Twinanda AP, Mutter D, et al describe the methodology and results from using a combination of both CNNs and LSTMs in order to predict remaining surgery duration (RSD) for Cholecystectomy surgery videos [2]. Specifically, they detail two different approaches, PhaseInferred-LSTM and Time-LSTM.

PhaseInferred-LSTM involved using a CNN (Resnet) to extract visual features from each of the frames in the surgery videos; these feature maps were then fed into an LSTM which would predict the current phase of the surgery being performed; finally, their model used the mean/median RSD of the predicted phase as the predicted RSD. Time-LSTM involved a similar approach of using a CNN as a feature extractor in combination within an LSTM. The difference was that instead of predicting phases and using summary statistics like mean/median from the phase, Time-LSTM trained the LSTM portion of the model to directly predict and model RSD. Essentially, instead of predicting phases, the LSTM was now predicting RSD directly.

Ultimately, Aksamentov I, Twinanda AP, Mutter D, et al reported both PhaseInferred-LSTM and Time-LSTM were able to produce highly accurate RSD predictions, with Time-LSTM outperforming PhaseInferred-LSTM.

3. Methodology

Based on the promising and exceedingly relevant results outlined in the previous section, we decided on exploring three similar approaches:

1. Using pure CNNs (as a baseline of sorts)
2. Using some combination of CNNs and sequence error correction methods
3. Using some combination of CNNs and sequence-based networks like RNNs or LSTMs.

4. Implementation

What follows are the specifications for each of the three different approaches we took for this assignment, as well as some of the reasoning behind them. Then, we review our training implementation as well as the reasons behind some of our choices with respect to training. It is worth noting that these approaches were all implemented in PyTorch.

4.1 CNNs

For our purely CNN-based approach, we decided to utilize transfer learning and fine-tune a wide array of existing CNN models. In our choices, we were aiming to use a combination of models that ranged in complexity and efficiency. Specifically, we trained ResNet-18, ResNet-50, and EfficientNet in this pursuit. The reason that we chose the ResNet and EfficientNet models is that while Resnet is still widely used, EfficientNet has shown better performance with a smaller number of parameters which can help with overfitting.

4.2 CNNs + Error Correction

One of our key observations from [1], was that there was quite a lot of mileage that could be gained from performing post-processing to "clean up" the prediction vectors from a given model. Some of the methodology from [2] also backs this up, where smoothing was performed on the PhaseInferred-LSTM predictions before evaluation. Inspired by these methods, we decided to train an LSTM to learn how to correct errors in the hernia phase data. To generate enough training data with only 50 training videos, we randomly added errors in the label sequences. This means that even if a sequence of labels is fed twice it won't have the same errors.

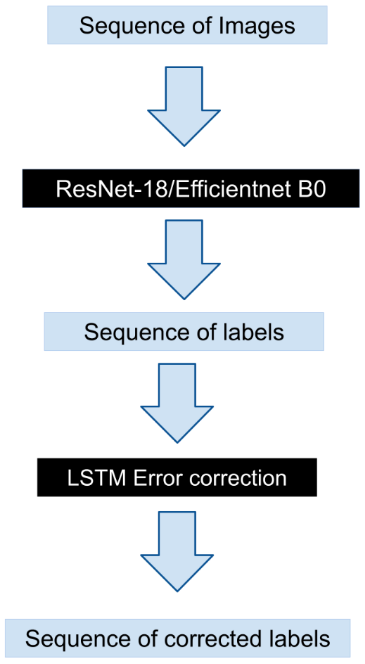

Fig. 1 *Prediction steps for a given video using a CNN model and an LSTM error correcting model*

4.3 CNNs + LSTMs

Following the logic of [2], we knew that using a CNN in conjunction with an LSTM can be moderately useful in predicting and modeling the different phases of surgery. Although the metrics in [2] were not predicting phases, the results were correlated with phases and thus gave us reason to try this methodology as well.

In implementing this approach we opted to finetune a ResNet-18 model on the surgery images before using it together with an LSTM. In line with [2], we used ResNet-18 as a feature extractor which then fed into an LSTM layer, followed by two fully connected layers with 128 and 14 (number of phases) each.

4.4 Training

We noticed that overfitting occurred only after 1 epoch during training. This can be explained by the fact that for a given video, a lot of frames from the same phase are going to be similar. This is why the higher capacity models performed worse, such as the ResNet-50 model. To increase the accuracy of models, a combination of different techniques was used.

The use of data augmentation with horizontal flips, random crops, random rotations, and sharpness were implemented.

Additionally, weight decay allowed to increase the accuracy by about 1 percent, and delay overfitting.

On all networks, training was achieved with an SGD optimizer with a momentum of 0.9 and learning rate decay. The learning rate was set to 0.001 and was halved after 2 epochs for a total of 4 epochs.

For the CNN-LSTM models, training was done with sequences of length 500 randomly sampled from the training video data. This length allows sequences to have at least 2 phases while keeping the memory footprint low. This fixed-length approach also allows using batch training as long as there is enough memory available

To have plenty of data available for the LSTM error correction model, we took sequences of labels from the training dataset and added random errors in the sequence to get anywhere from 10 to 30 percent of errors. The LSTM model then tries to learn how to correct random errors. The idea is that the model tries to correct random errors that appear in a long sequence of a phase.

5. Results

5.1. CNNs

	ResNet-18	**EfficientNet (b0)**
Inference speed	1.09 ms/frame (1085 fps)	1.98ms/frame (1975 fps)
Accuracy	78.29 %	78.9 %
Micro F1 score	0.78	0.79
Macro F1 score	0.50	0.49
Weights size	43 Mb	15.6 Mb
Number of parameters	11 million	5.3 million

Fig. 2 *Comparison of models*

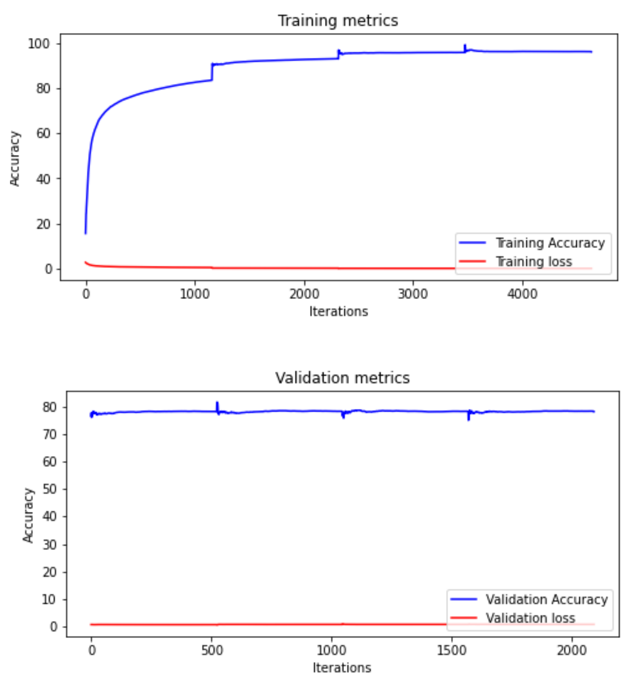

Fig. 3 *Training metrics for the ResNet-18 model*

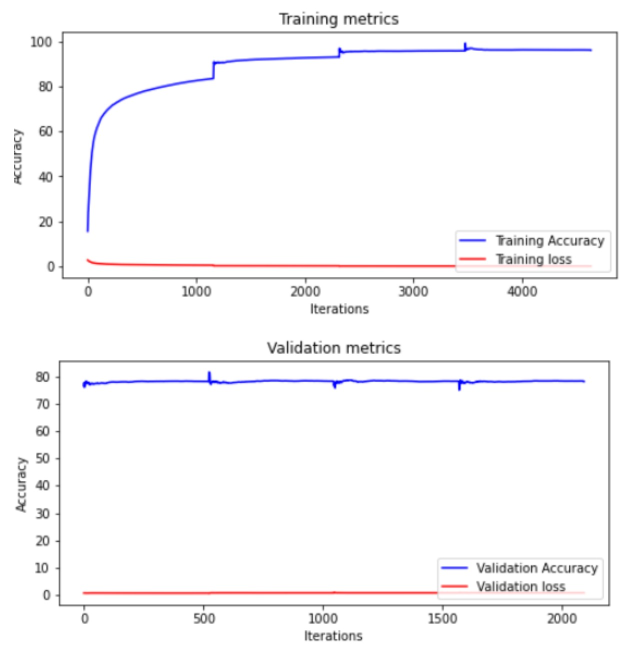

Fig. 4 *Training metrics for the EfficeintNetb0 model*

5.2. CNNs + Error Correction

The use of a simple LSTM network to detect errors in the sequences returned by the CNN networks allowed to increase the accuracy of the test set by 1 percent from 77% to 78%. A more advanced LSTM model using the features of the second last layer of the ResNet-18 layer was tested, but we could not get past the accuracy of a simple CNN setup.

5.3. CNNs + LSTM

Again, this network could not get past the accuracy of a baseline CNN approach. More data could allow it to make it perform better. Even freezing the trained CNN layers did not yield satisfactory results, with accuracies under 70%.

6. Insights Gained and Future Work

Our most surprising result was that the CNN + LSTM model was outperformed by the pure CNN based models. Based on our initial research and intuition, we believed that incorporating RNNs and LSTMs would only improve our results since our data involved highly sequential data. However, this was not the case. We believe this might have occurred for a variety of reasons, but we attribute it mostly to the data having minimal changes from frame to frame.

To get a better F1 score we could have tried to sample the training data differently as some classes were underrepresented. Additionally, having a look at the order of the phases, and the logical steps of a hernia surgery, it would be interesting to implement some heuristic algorithms to detect mislabeling and correct the errors.

7. Conclusion

In this report we have outlined the three approaches we implemented in trying to predict hernia phases based on video data. Namely, we attempted a pure CNN-based approach, a CNN + Error Correction approach, and a CNN + LSTM based approach. Interestingly, we found that CNN + Error Correction model gave us the best results, much to our surprise. However, this approach leaves little room for improvement. In the future we might want to explore more sequence-based approaches, as we intuitively believe models more closely resembling the structure of the data would lead to much better results in the long run.

8. References

[1] Rémi Cadène, Thomas Robert, Nicolas Thome, Matthieu Cord. “M2CAI Workflow Challenge: Convolutional Neural Networks with Time Smoothing and Hidden Markov Model for Video Frames Classification” Sorbonne Universites, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606 (2016).

[2] Aksamentov I, Twinanda AP, Mutter D, et al. “Deep Neural Networks Predict Remaining Surgery Duration from cholecystectomy videos.” 2017 International Conference on Medical Image Computing and Computer-Assisted Intervention: Springer; (2017): 586–593.

[3] Anh-Vu Nguyen, Brian Hernandez, Wael Boukhobza (2022). Hernia Surgery Phase Prediction [code] https://github.com/ecbme6040/e6691-2022spring-assign2-SURG-an3078-bmh2168-wab2138