Capsulorrhexis Trajectories for Automated Surgical Training Feedback

Anonymous CVPR submission

Paper ID Rhexis

Abstract

Surgical training can vary widely in quality and volume for residents and fellows at different training programs and there is a significant need for better feedback so that trainees can maximize the educational value of every surgical case. Machine learning-based computer vision methods have been shown to be useful in providing automated quantitative assessments of surgical skill from laparoscopic and open surgical videos.

Inspired by these examples, we finetuned a Keypoint R-CNN FPN model with a ResNeXt101 backbone pre-trained on the COCO dataset to generate surgical instrument trajectories for the capsulorrhexis step of cataract surgery, achieving a mean average precision of 94.65%.

1. Introduction

1.1. Background

The quality and volume of surgical training can vary widely between different training institutions, and can be significantly affected by unexpected circumstances such as the recent (and still ongoing) COVID-19 pandemic that saw operating rooms across the world close except for emergent cases [8]. As such, there is a great need for trainees to maximize the educational value of every surgical case they perform. In recent years, internet platforms such as YouTube have enabled surgeons to share cases and discuss new techniques, and allowed trainees to receive valuable feedback from more experienced surgeons on their own cases; however, these initiatives are limited by the availability of experienced surgeons with the time and willingness to review trainee-submitted cases outside of working hours and provide feedback. To address this limitation, we propose a machine learning platform that provides automated feedback for trainee-submitted surgical videos, specifically cataract surgery.

For the purposes of this project, we focus solely on the capsulorrhexis step of cataract surgery: in this step, the surgeon uses a forceps to peel a 5.5-6mm circle in the 15um thick anterior lens capsule (akin to peeling a circle in the skin of a grape) through a 2mm main incision in a space that is roughly 4mm deep and 7mm x 7mm horizontally. The capsulorrhexis is particularly challenging for training cataract surgeons because it is one of two critical steps in which complications most commonly occur, and uniquely stresses the foundational skills of 'floating' and 'pivoting' within the 2mm main incision necessary for maneuvering safely and effectively within the eye.

1.2. Problem Statement

The focus of our CS231N project is to develop a keypoint detection model that tracks the tips of the utrada forceps instrument over the course of the capsulorrhexis in order to generate instrument trajectories. These instrument trajectories will then be fed to downstream models (the focus of our CS229 project) to generate quantitative and automated feedback for training cataract surgeons. Additionally, we employ a pretrained semantic segmentation model to generate additional features of interest such as pupil center and boundaries as well as incision location (which can be more challenging to label with keypoints) to further enrich our feedback [15].

The inputs to our keypoint detection model are JPEG images generated from the frames of MP4 videos of the capsulorrhexis step of cataract surgery, along with ground-truth bounding box and keypoint labels in the COCO JSON format [13]. We then use an R-CNN-FPN model to output bounding box and keypoint predictions for the utrada forceps and utrada forceps tips respectively for each frame of the input video. We then concatenate and plot the predicted keypoints to generate instrument trajectories for each surgical video.

Our baseline method is a Keypoint R-CNN ResNet50-FPN model pre-trained on the COCO dataset [13] in Detectron2's model zoo [21].

2. Related Work

Our project is heavily inspired by work done in the the Stanford MARVL Lab and previous work done in automating the assessment of surgical skill from surgical videos.

Progress in the field has historically coincided with large datasets and challenges hosted at conferences such as MICCAI. Indeed, the M2CAI 2016 Tool Presence Detection Challenge for laparoscopic cholecystectomy [20] and its associated m2cai16-tool dataset [14] led to significant advances in deep-learning-based approaches for instrument detection for laparoscopic surgery [16].

Later datasets enabled finer-grained analysis of surgical skill using spatial detection of surgical tools, such as the m2cai16-tool-location which extended m2cai16-tool with bounding box spatial annotations of tools. Jin et al. fine-tuned all layers of an ImageNet-pretrained Faster R-CNN [17] model with VGG16 backbone and region proposal network for 40,000 iterations with data augmentation via horizontal flipping and learning rate decay every 10,000 iterations for spatial detection of surgical tools, in addition to tool presence detection. They achieved a mean average precision (mAP) of 63.1% for spatial detection of surgical tools and a mean average precision (mAP) of 81.8% on frame-level surgical tool presence detection, significantly outperforming all previous results in the M2CAI 2016 Tool Presence Detection Challenge. Of greater clinical relevance, the resulting spatial annotations of surgical tools were then used to create timelines of tool usage, heat maps of bounding box locations, and tool trajectory maps for the critical clipping phase of laparoscopic cholecystectomy, and generated useful metrics of bimanual dexterity, efficiency, and overall surgical skill for feedback [9].

Significant work has also been done in the realm of open surgery, especially with the use of YouTube and other video sharing sites as data sources. Zhang et al. scraped 188 videos of common open surgical procedures from YouTube including 70 breast, 88 gastrointestinal, and 30 head-and-neck videos for surgical hand detection and tracking using a RetinaNet [12] object detection model with a ResNet-50-FPN backbone and focal classification loss for hand detection, and a modified Simple Online and Realtime Tracking algorithm [1] for hand tracking across frames. They achieved a mean average precision (mAP) of 70.4%, and trajectories of the epicenters of hand bounding boxes highlighted insights that were consistent with independent surgeon review of videos [23].

More recently, Goodman et al. curated the Annotated Videos of Open Surgery, a dataset of 1997 YouTube videos of a diverse range of open surgical procedures from over 50 countries over the last 15 years. 343 videos were annotated with hand-and-instrument bounding boxes, as well as 21 hand joint keypoints and per-second resolution temporal labels of cutting, tying, suturing, or background. Using this dataset, they developed a deep multi-task neural network with a ResNet50 backbone, action recognition branch based on the R(2+1)D architecture [19] with factorized spatial and temporal convolutions trained with cross entropy loss, hand and tool detection branches using RetinaNet with a shared feature pyramid network trained with focal losses, and hand pose estimation using a Deep High-to-Low Resolution Network [18]. Simultaneous optimization of spatial and temporal model branches was achieved with alternating task training on randomly sampled frames and 5 second video clips respectively where gradients only flowed through the targeted heads and shared encoder at each iteration of training. Detections were tracked across frames using the SORT algorithm [1]. The model achieved mean precision of 0.71 and mean recall of 0.73 for action recognition at one second temporal resolution, as well as mean average precisions of 0.89 for hand bounding box detection, 0.46 for tool bounding box detection across all test videos, and an average probability of correct keypoint (PCK) value of 0.38 across all test videos. Unification of multiple vision tasks into a single multi-task architecture resulted in a parameter-efficient model that could generate spatial analytics every 0.08s and temporal action characterizations every 0.33s on a personal workstation with a single RTX 3090 GPU. Finally, hand and hand pose trajectories for 23 open surgeries with 14 surgeons of varying levels of experience (medical students, residents, fellows, and attendings) at Beth Israel Deaconess were found to be promising tools for quantitatively evaluating surgical skill and providing feedback for skill improvement [5].

For cataract surgery, much of the previous work has focused on the temporal classification of the different steps of cataract surgery: most recently. Yeh et al. trained a VGG16 and VGG16-LSTM to classify between 13 steps of cataract surgery, achieving a mean average precision of 0.83 and 0.92 respectively [22]; and Nespolo et al. trained a Faster R-CNN model to classify the step of cataract surgery and spatially detect the pupil for each frame of a cataract surgery video and added optical flow to track the relative acceleration and velocity of surgical instruments to develop a real-time feedback system for cataract surgery, offering audiovisual warnings in the event of potentially risky instrument movements or decentration of the eye from primary position [4].

The CATARACTS Semantic Segmentation Grand Challenge 2020 and its associated CADIS dataset also sparked work towards developing semantic segmentation models of surgical instruments and anatomical landmarks for cataract surgery [6]. The current state of the art trained an ensemble of encoder-decoder models with ImageNet pretrained ResNet encoders and UPerNet, OCRNet, and DeepLabv3+ decoders using the Lovasz-Softmax loss (a differentiable surrogate of the mean IoU metric) instead of cross entropy loss, and with repeat factor sampling to address class imbalance. In ablation studies, the use of the Lovasz-Softmax loss and repeat factor sampling were found to significantly boost performance. Deeper ResNet backbones (ResNet-50,

ResNet-101) and choice of decoder head (UPerNet, OCRNet, and DeepLabv3+) did not seem to significantly change results; however, switching to ResNeXt50 and ResNeXt101 backbones showed promising improvements at the cost of higher GPU memory usage [15].

To our knowledge, ours is the first study to perform keypoint detection in addition to bounding box detection for cataract surgery, while also incorporating features from semantic segmentation for automated surgical feedback. As cataract surgery is performed in a such a constrained space (operating in a space that is roughly 4mm deep and 7mm x 7mm horizontally through a 2mm main incision and 0.8mm sideport incision as mentioned previously), every movement must be purposeful and precise. We believe that being able to track instrument tips and anatomical landmarks over the course of cataract surgery and compare instrument trajectories between experts and trainees can yield valuable feedback for the training cataract surgeon.

3. Methods

Our approach is to fine-tune from Keypoint R-CNN FPN with different backbones (ResNet50-FPN, ResNet101-FPN, ResNeXt101-FPN) pre-trained on the COCO dataset, and modified to predict 2 keypoints (utrada tip left and utrada tip right) instead of 17 used for human pose estimation.

Keypoint R-CNN FPN is a slightly modified Mask R-CNN FPN (itself an extension of Faster R-CNN FPN) to detect instance-specific keypoints by treating each keypoint as a one-hot binary mask with only 1 pixel labeled as foreground.

Faster R-CNN consists of two stages: first, a backbone convolutional neural network extracts feature maps from input images and a region proposal network (RPN) regresses bounding box bounds and object vs. background scores at each location on a regular grid to propose candidate bounding boxes. Then, a second stage extracts features using RoIPool from each candidate bounding box and performs classification and bounding box regression [17].

The region proposal network(RPN) is an 3 x 3 convolutional layer that slides over the feature map output of the backbone convolutional neural network followed by two parallel 1 x 1 convolutional layers for bounding-box regression and classification [17]. At each sliding window location, the bounding-box regression and classification layers predict bounding box offsets and object vs. background classification for K anchor boxes of different sizes and scales. The loss of each training image in the RPN is shown in equation 1 [9].

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda(\frac{1}{N_{reg}})\sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (1)$$

In the above equation, i indicates a particular anchor box position. p_i is the probability that the current anchor box position contains an object and t_i is the coordinates of the predicted bounding box. p_i^* is the actual ground truth label of weather the anchor box contains an object, while t_i^* are the coordinates of a ground truth bounding box that corresponds to a ground truth positive anchor box. The loss is composed of two terms, the classification loss term for object detection L_{cls} and a regression loss term for bounding box regression L_{reg} [9].

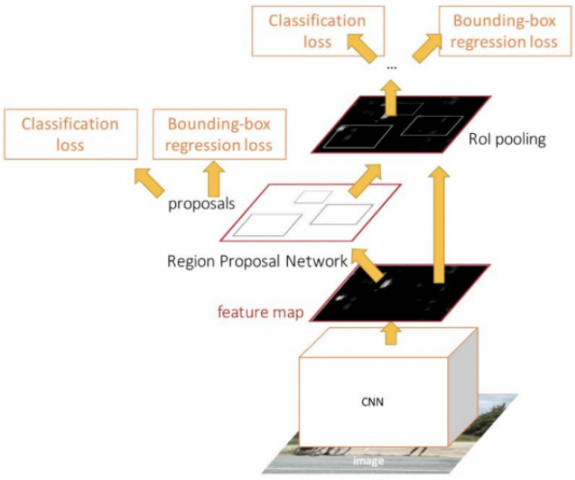


Figure 1. Faster R-CNN Architecture — Image from CS231n L9-Slide 92 [10]

RoIPool (or region of interest pooling) uses max pooling to convert the features inside a region of interest (e.g. size h x w) into a feature map of fixed spatial dimensions (H x W) to be passed as input for classification and bounding box regression. RoIPool accomplishes this by dividing the (h x w) region of interest into a grid of (H x W) bins with size approximately (h/H x w/W), and then max-pooling each bin to achieve a feature map of size (H x W). An example illustration of the RoIPool layer is shown in Figure 2. Note that a drawback of utilizing this method is that proposed input regions may have different sizes, but the resulting features of the regions will always have the same size [17].

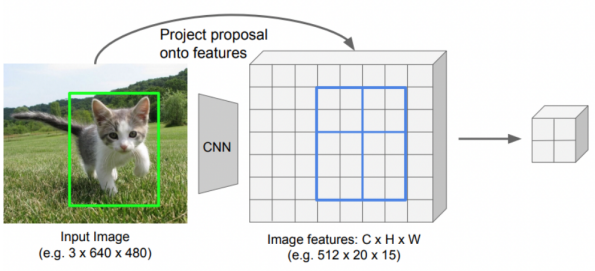


Figure 2. RoI Pool — Image from CS231n L9-Slide 68 [10]

Mask R-CNN extends Faster R-CNN by adding to the existing classification and bounding box regression branches a third branch for predicting binary segmentation masks on each Region of Interest [7]. This slight difference in the architecture of Mask R-CNN from Faster R-CNN (Figure 1) can be seen in Figure 3.

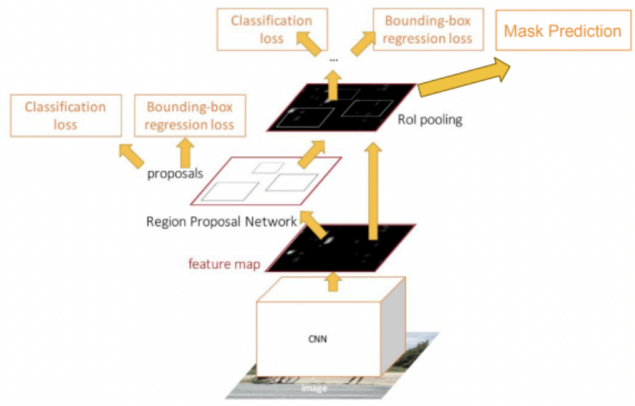


Figure 3. Mask R-CNN — Image from CS231n L9-Slide 93 [10]

This mask branch, a fully convolutional network, outputs K (m x m) resolution binary masks (one for each of the K classes). To this, a per-pixel sigmoid is applied with average binary cross-entropy loss based on the ground-truth class (i.e. for a region of interest associated with ground-truth class k, the mask loss is defined only on the kth mask and other mask outputs do not contribute to the loss). The equation for the cross entropy loss of a single training example i is shown in Equation 2.

$$L_i = -\log\left(\frac{\exp f_{y_i}}{\sum_j \exp f_{y_j}}\right) \tag{2}$$

Where f_{y_i} is the output score of the model for the ith class.

For Keypoint R-CNN, cross-entropy loss is minimized over an m^2 sized softmax output which encourages that a single point is detected.

Because the RoIPool layer used to extract feature maps from each RoI for classification and bounding box regression quantizes (i.e. 'snaps to grid') the RoI boundaries and spatial bins, resulting in misalignments between the RoI and extracted features, the mask branch instead uses a quantization-free RoIAlign layer which uses bilinear interpolation to compute exact (non-quantized) values of input features at 4 regularly sampled locations in each RoI bin before max-pooling within each bin to achieve the target feature map size while preserving pixel-to-pixel alignment and exact spatial locations. An illustration of the RoI align layer is shown in 4.

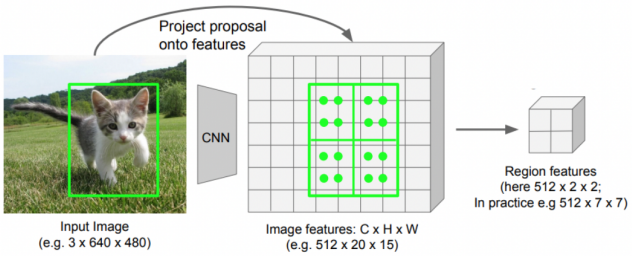


Figure 4. RoI Align — Image from CS231n L9-Slide 73 [10]

More recently, both Faster R-CNN and Mask R-CNN have demonstrated excellent gains in accuracy and speed by incorporating a feature pyramid network (FPN) which constructs a feature pyramid to enhance recognition, spatial detection, and segmentation of objects at different scales [11].

Feature pyramids were a standard component for detecting objects at different scales in traditional computer vision techniques, but were avoided in deep learning models due to the computational and memory cost. The feature pyramid network exploits the inherently pyramidal hierarchy of convolutional neural networks (with feature maps of different sizes) to construct feature pyramids at marginal extra cost for significant improvements in feature extraction. The feature pyramid network is a fully convolutional network that inputs a single scale image of arbitrary size and outputs proportionally sized feature maps at multiple levels using a bottom-up pathway and a top-down pathway linked by lateral connections. The bottom-up pathway is the feed-forward computation of the backbone convolutional neural network which generates a hierarchy of feature maps of decreasing spatial resolution (classically by a factor of 2 at each stage) but increasing semantic value. For ResNets, these correspond to the feature activation outputs at the last residual block for each feature map size. The top-down pathway takes the top-most feature map and sequentially upsamples (by the same factor that the bottom-up pathway downsampled) down the pyramid. Lateral connections merge feature maps of the same spatial dimensions from

the bottom-up (higher resolution but semantically weaker) and top-down (lower resolution but semantically stronger) pathways by element-wise addition. Then, 3 x 3 convolutions are passed over each merged map to reduce the aliasing effect of upsampling and generate the final set of feature maps.

In Faster R-CNN and Mask R-CNN, the feature pyramid network enhances the region proposal network by adding the same 3 x 3 convolutional layer followed by 2 parallel 1 x 1 convolutional layers to each level of the feature pyramid to increase robustness to scale variance. Because each level of the pyramid represents feature maps of a different size, only anchor boxes of a single scale are assigned to each level. The feature pyramid network furthermore allows the region proposal network to pass feature maps from the most appropriate level of the pyramid to the second stage classification, detection, and segmentation branches based on the size of the region of interest, for instance, choosing a finer-resolution feature map for a smaller region of interest.

We chose to implement our model using the Detectron2 [21] library due to the availability of many pretrained models. We utilized Detectron2's default optimizer and dataset configuration code, and implemented our own custom configuration object and trainer. We also implemented a python script that we integrated with wandb.ai to track model performance across various experiments.

4. Dataset and Features

Two ophthalmologists curated a dataset of roughly 200 surgical videos performed by novice, intermediate, and expert surgeons from different institutions as well as from YouTube, recorded from the surgical microscope and capturing the surgeon's point of view while operating. The videos varied in resolution from 712x478 to 1920x1080 depending on the data source. The surgical videos were manually trimmed to the capsulorrhexis step of cataract surgery using QuickTime, and a Python script `mp4tojpeg.py` using the openCV library converted the mp4 video files into directories containing each video frame as a JPEG image. [2]

Next, utilizing another custom python script, `generate_training_data.py`, 1000 video frames from 40 different videos were randomly selected to form "Dataset-1000" (initially, 20 random frames were selected from each of 20 randomly selected videos to form "Dataset-400", but as we progressed, it became clear that more data was required). These 1000 images were split into training (12 videos, 800 images), validation (4 videos, 100 images), and test (4 videos, 100 images) datasets and uploaded to CVAT.org for keypoint and bounding box annotations by our two ophthalmologists [3]. An example input image is shown in Figure 5.

The completed annotations for training, validation, and test datasets were exported in CVAT 1.1 XML format and converted into COCO JSON format with datamaru, CVAT's associated command-line utility. Because datumaru saves a separate COCO JSON for each annotation task (bounding box, keypoint), a Python script `merge_coco_jsons.py` merged the separate JSON files into one complete annotation file for each dataset. Images and annotations were then loaded into Detectron2 using Detectron2's `register_coco_instances()` function.

To improve our model's performance, we implemented a custom trainer object to perform data augmentation through horizontal and vertical flips, random crops and resizes, and random changes to brightness, contrast, and saturation.

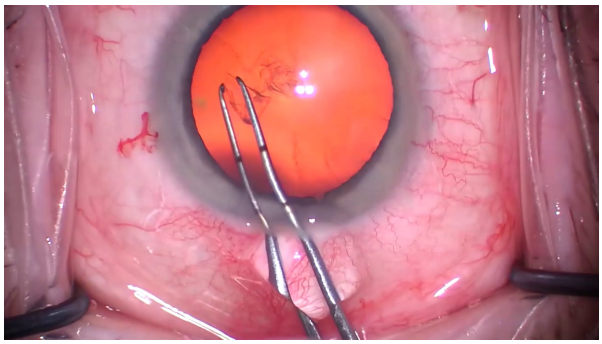


Figure 5. Example input image: Capsulorrhexis with Utrada forceps

5. Experiments and Results

Over the course of our project, we experimented with different backbone model architectures, learning rates, and data augmentations based on our review of the literature and the constraints of our small dataset.

5.1. Evaluation Metrics

The primary objective metric for our experiments was mean average precision (mAP) of keypoint predictions on our validation set, modeling our evaluation metrics after that of the keypoint detection task in the COCO challenge. [13]

5.2. Data augmentation

Given the small size of our dataset, we experimented with different forms of data augmentation that are commonly encountered in cataract surgery during training, namely horizontal and vertical flips (which could reflect left vs. right-handedness of the surgeon or changes in surgical camera orientation) as well as random changes in brightness, contrast, saturation as would be expected from surgeries filmed using different cameras in different operating rooms. We also included random crops to reflect the differences in field of view between different cameras and surgical microscopes.

With our initial dataset of 400 frames and a learning rate of 0.000025, we compared ResNet50, ResNet101, and ResNeXt101 with and without data augmentation to assess the impact of model depth, model architecture, and data augmentation on model performance. As seen in Figure 6, data augmentation improved model performance across the board, and as a result, we included data augmentation in all experiments going forward.

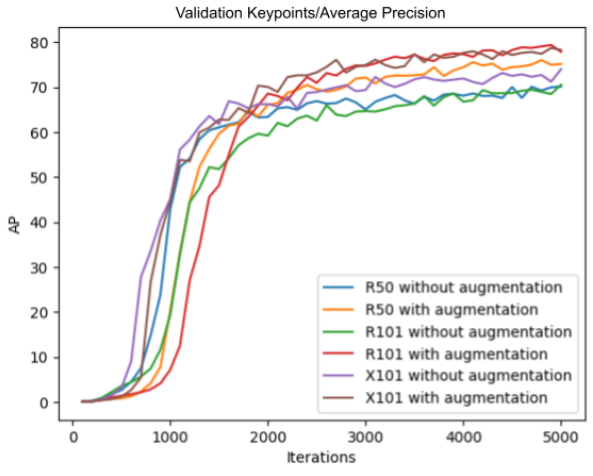


Figure 6. Data augmentation improves model performance

5.3. Increasing data set size

We also continued to add to our dataset whenever model performance plateaued. Increasing our original dataset from 400 frames to 1000 frames resulted in a significant boost in performance, as our results show in Table 1.

Table 1. Best Test Set Performance Metrics Across all models for different dataset sizes.

Number of Datapoints	Best Bounding Box AP	Best Keypoints AP
400	70.83	77.41
1000	79.21	80.96

5.4. Relaxing bounding box

Later, visual inspection of model predictions revealed that bounding box predictions often cut off instrument tips such that keypoints could not be predicted. Hypothesizing that this could be due to how tightly we drew our ground truth bounding box annotations around the utrada forceps, we wrote a custom python script, `edit_json_files.py`, which allowed us to move the top left corner of our bounding boxes to the left and up by 10 pixels (or to the edge of the image if the edge was within 10 pixels). A before and after image of a labeled bounding box is included in the appendix in Figure 11. Having relaxed our bounding boxes, we observed another significant improvement in performance, as demonstrated in Figure 7 and Table 2.

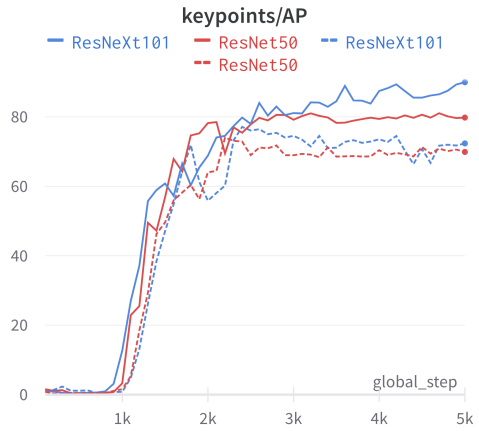


Figure 7. Bounding box relaxation improves model performance (dashed lines: before, solid lines: after). Note: we do not have a ResNet101 AP curve before bounding box relaxation due to initial technical difficulties with wandb.ai

Table 2. Effect of bounding box relaxation on model performance

reBB	Model	Bounding Box AP	Keypoints AP
N	ResNeXt101	79.21	80.96
N	ResNet101	75.56	77.96
N	ResNet50	69.22	72.66
Y	ResNeXt101	80.388	94.65
Y	ResNet101	79.06	84.55
Y	ResNet50	76.54	82.39

5.5. Model architecture and depth

The original Mask R-CNN paper found that deeper models (i.e. ResNet101 instead of ResNet50), ResNeXt model architectures, and the addition of a feature pyramid network (FPN) resulted in better performance on the COCO dataset. [7] However, the current state of the art for the CaDiS dataset noted that while ResNeXt performed better than ResNet, additional model depth did not seem to improve results. [15] Thus, we test how model depth and the use of ResNet vs. ResNeXt architectures would affect our results.

Initially we did not see much of a difference between different model depths and architectures; however, after adding data augmentation, increasing our dataset size, and relaxing our bounding box annotations, ResNeXt101 began to clearly outperform ResNet50 and ResNet101, sim-

ilar to what was observed in Pissas et al. with the CaDis dataset [15].

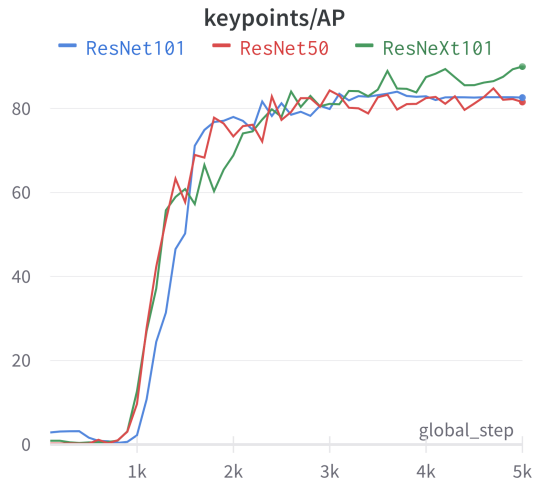


Figure 8. ResNeXt101 outperforms ResNet50 and ResNet101

5.6. Learning rate

Finally, we also experimented with different learning rates, including 0.000025 as suggested by Detectron2, 0.0001, and 0.001 as reported by Jin et al. [9]. Noting steady improvement in training with learning rates of 0.0001 and 0.001 compared to 0.000025, we investigated whether increasing the learning rate further would continue to yield improvements, but ultimately found that training became unstable with a learning rate of 0.005 resulting in performance degradation and crashes due to floating point errors (Figure 9).

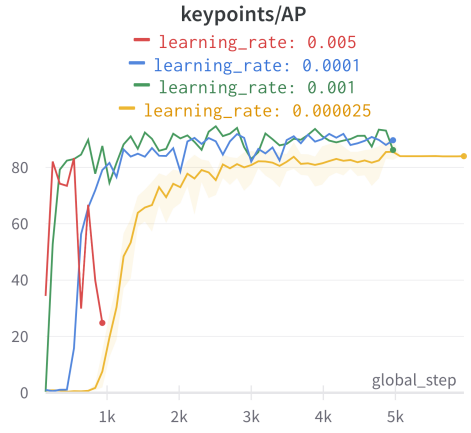


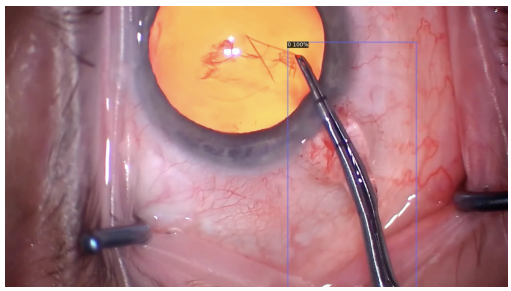
Figure 9. Similar to Jin et al., a learning rate of 0.001 was optimal for training our model.

5.7. Best model

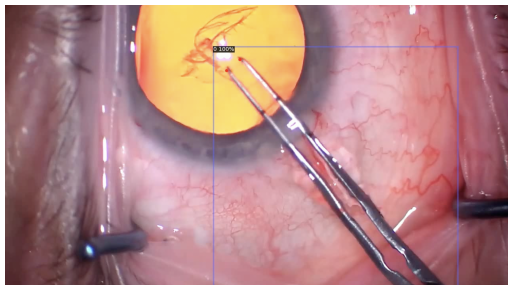
Our best model was a Keypoint R-CNN-FPN with a ResNeXt101 backbone trained with data augmentation and a learning rate of 0.001 on on our 1000-frame dataset with relaxed bounding boxes. This model achieved high mean average precision for both keypoints (94.65) and and bounding boxes (80.388) (3). Qualitative results were similarly excellent, as shown in Figure 10; however, in cases where other surgical instruments are visible, the model will accurately but incorrectly label them as utrada forceps as well. As is shown in Figure 10c, the model correctly detects the utrada forceps (right) but incorrectly detects the 0.12 toothed forceps (left) being used to stabilize the eye during the capsulorrhexis.

Table 3. Best Model Test Set Performance Metrics

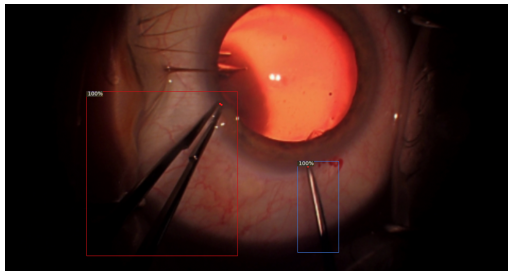
Model	Bounding Box AP	Keypoints AP
ResNeXt101_Best	80.388	94.65



(a) Example output: Closed Utrada forceps



(b) Example output: Open Utrada forceps



(c) Example output: Mislabeled secondary instrument

Figure 10. Example outputs of our best-performing model

5.8. Discussion

In a class that has had us implement cutting edge transformers, GANs, and self-supervised methods, it is ironic that the biggest lesson that we learned over the course of this project was the central importance of data in machine learning. Of everything that we experimented with to improve our model's performance, the most impactful changes were adding data augmentation, increasing the size of our dataset, and (most surprising to our group) relaxing our bounding boxes. There remains much room for further improvement; however, all in all, we are very pleased by our model's performance and excited to begin generating instrument trajectories to provide insightful feedback to cataract surgeons in training.

6. Conclusion and Future Work

Through experimentation with different model architectures, data augmentations, and learning rates coupled with continued expansion and refinement of our dataset, we successfully trained a keypoint detection model to track the tips of the utrada forceps instrument over the course of the capsulorrhexis step of cataract surgery with 94.65% mean average precision for keypoint detection of utrada forceps instrument tips and 80.388% mean average precision for bounding box detection of the utrada forceps. These quantitative metrics are backed up by qualitatively accurate keypoint predictions on test videos as shown in Figure 10.

Our team has learned a great deal over the course of this project and are proud of our model's performance; however, there are many possible avenues for further exploration and improvement:

First, our dataset is still quite small by computer vision standards and limited only to keypoint and bounding box annotations of the utrada forceps: even for just the capsulorrhexis step of cataract surgery, there are several other features of interest such as the pupil center and its boundaries that would be potentially valuable to track in order to provide better feedback to surgeons, not to mention the other instruments and relevant anatomical landmarks involved in other steps of cataract surgery. Creating a larger and richer dataset and perhaps incorporating additional object detection and segmentation heads into our model would greatly improve the utility and robustness of the overall model. For instance, one failure case of our model occurs when other surgical instruments come into view - although our model does an excellent job of identifying the tips of these instruments, it does not know that these are not utrada forceps and should be ignored for this particular step of cataract surgery.

Second, we have not yet successfully leveraged the temporal component of our videos to improve keypoint detection and tracking: we had made initial forays into exploring the use of Lucas-Kanade sparse optical flow and RAFT (recurrent all-pairs field transforms) dense optical flow to track instrument tips across frames; however, we struggled to adjudicate between our Keypoint R-CNN-FPN model and optical flow model when predictions differed. Further exploration into optical flow or the use of recurrent neural networks, transformers, or 3D convolutional neural networks that can model temporal dependencies could also greatly improve performance.

Third, given how much of a challenge that we had (correctly) annotating our dataset, it would be fascinating to explore weakly-supervised or self-supervised methods for object detection, segmentation, and keypoint detection and see how much they can boost performance with a small number of labelled videos and a much larger number of unlabelled videos.

Finally, the fields of keypoint detection and object detection and segmentation are full of innovative architectures ideas: it would be most interesting to try anchor-free single stage object detection models such as CornerNet or CenterNet which directly predict pairs or triplets of keypoints to generate object detection bounding boxes, or transformer-based methods such as DETR. We made preliminary investigations into adapting CornerNet and CenterTrack (an object tracking extension to CenterNet) to our task, including writing (code) to resize our images to the standard input size for CornerNet; however, we were ultimately unable to achieve a working implementation in time.

7. Appendices

7.1. Project Hyperlinks

- Rhexis Trajectory Project GitHub
  https://github.com/davidekuo/rhexis-trajectory
- Google drive link to our Dataset
  https://tinyurl.com/53zh3ezf

7.2. Additional figures

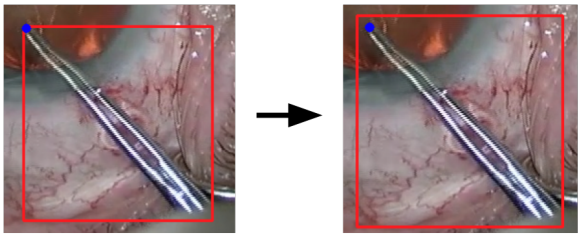


Figure 11. Before and after bounding box relaxation. Since the forceps tip usually is near the top left corner of the bounding box, this manipulation better ensures that forceps tip is included in each bounding box for keypoint detection.

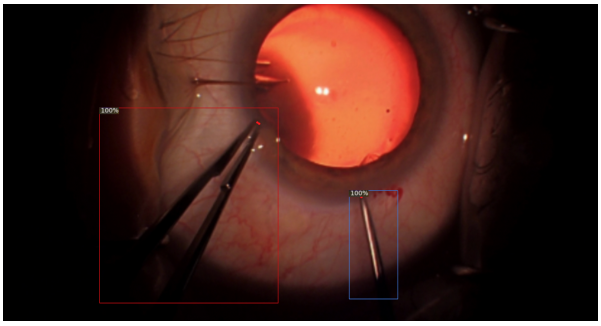


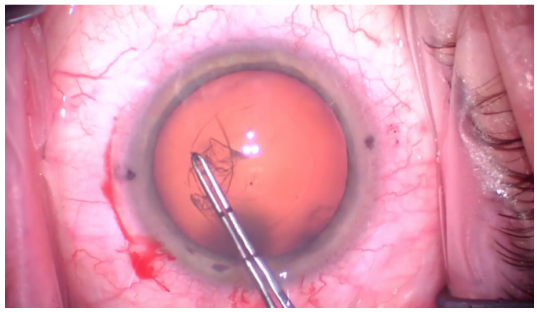
Figure 12. The model correctly detects the utrada forceps (right) but incorrectly detects the 0.12 toothed forceps (left) being used to stabilize the eye during the capsulorrhexis

7.3. Exploratory Experiments

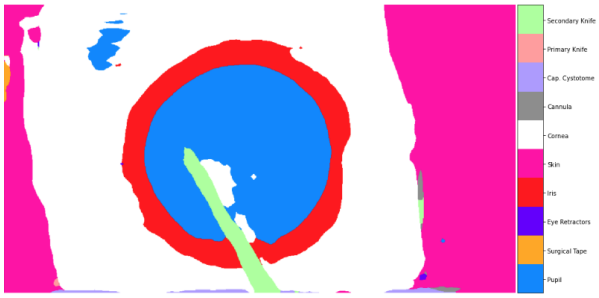
7.3.1 Semantic Segmentation

Our secondary focus after keypoint detection was to implement a semantic segmentation model so we could identify the positions of relevant anatomical landmarks in relation to the utrada tip, (particularly the incision point and the pupil. To implement semantic segmentation, we utilized the publicly available MICCAI2021 Cataract Semantic Segmentation Model [15] from the Robotics and Vision in Medicine Lab at King's College of London. To utilize the segmentation model, the images were read in and resized to a standard 960x540x3, which the MICCAI2021 model utilizes as it's input file size. We additionally had to add a `nn.Softmax2d` layer to the output scores to generate the final predictions. With only these minor modifications to the original codebase, we were able to generate output pixel labels that we stored in PNG format. The final pixel labels were resized back to the same spatial dimensions as their corresponding original input images through majority class nearest neighbors interpolation.

Although we do not have the pixel level labels to perform an objective evaluation of the performance of the semantic segmentation network, through our subjective evaluation of the images, the model appeared to work pretty well. Figure 13 shows an example of the output of the segmentation network on one of our images.



(a) Input image



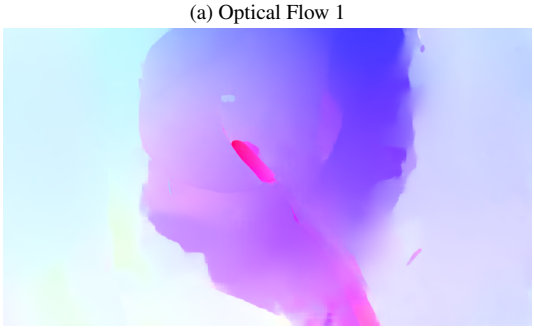
(b) Output pixel level segmentation

Figure 13. Semantic Segmentation Performance Example

7.3.2 Optical Flow

Another avenue we explored was to utilize optical flow techniques which track the movement of pixels over time, specifically Lucas-Kanade sparse optical flow and RAFT (recurrent all-pairs field transforms) dense optical flow, to augment our ability to track the utrada tip between frames. However, we found it difficult to handle situations when optical flow predictions disagreed with our keypoint detection model's predictions. We did however generate many beautiful pictures with RAFT.

(a) Optical Flow 1



(b) Optical Flow 2

Figure 14. Optical Flow from two different frames in an video

8. Contributions and Acknowledgements

GitHub repositiory included in Appendix Section 7.1

8.1. CS 231n Team

- Ben Viggiano: I wrote a lot of our code base from reading in input images, set up model object classes like evaluators, and contributed to the development of our training script functionality. I also implemented the Semantic Segmentation model to generate label PNG images (reference in appendix), and created code to resize our training images if needed for different models we were trying such as CornerNet and CenterNet.
- David Kuo: I curated and annotated our dataset, wrote the data processing pipeline to transform our data into COCO JSON format, helped implement our initial model in Detectron2, helped run hyperparameter tuning experiments, and (unsuccessfully) explored the use of optical flow, CornerNet, and CenterTrack.
- Ben Ehlert: I helped implement our model through Detectron2 and created a custom configuration class that we used to manage data augmentation functionality. I also helped label data and run hyperparameter sweeps.

8.2. Other acknowledgements

- Riya Sinha, a graduate student that was part of the team for CS229 portion of our project, contributed by labeling data and running models from our code base so we could test more hyperparameters.
- Susan Qi MD, MS, an ophthalmology resident contributed by labeling data and providing videos.
- Emmett Goodman, a postdoctoral fellow in the MARVL Lab, gave us guidance in the early stage of the project.
- Zhuoyi Huang, our TA mentor,a Masters Student in Computer Science

References

[1] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. *CoRR*, abs/1602.00763, 2016. 2

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 5

[3] Intel Corporation. Computer vision annotation tool. https://github.com/openvinotoolkit/cvat, 2018. 5

[4] Rogerio Garcia Nespolo, Darvin Yi, Emily Cole, Nita Valikodath, Cristian Luciano, and Yannek I. Leiderman. Evaluation of artificial intelligence–based intraoperative guidance tools for phacoemulsification cataract surgery. *JAMA Ophthalmology*, 140(2):170, 2022. 2

[5] Emmett D. Goodman, Krishna K. Patel, Yilun Zhang, William Locke, Chris J. Kennedy, Rohan Mehrotra, Stephen Ren, Melody Y. Guan, Maren Downing, Hao Wei Chen, Jevin Z. Clark, Gabriel A. Brat, and Serena Yeung. A real-time spatiotemporal AI model analyzes skill in open surgical videos. *CoRR*, abs/2112.07219, 2021. 2

[6] Maria Grammatikopoulou, Evangello Flouty, Abdolrahim Kadkhodamohammadi, Gwenol'e Quellec, Andre Chow, Jean Nehme, Imanol Luengo, and Danail Stoyanov. Cadis: Cataract dataset for image segmentation, 2019. 2

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. 4, 6

[8] R. Hussain, B. Singh, and N. Shah. Impact of covid-19 on ophthalmic specialist training in the united kingdom—the trainees' perspective. *Eye*, 34:2157–2160, 2020. 1

[9] 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. *CoRR*, abs/1802.08774, 2018. 2, 3, 7

[10] Fei Fei Li, Jiajun Wu, and Ruohan Gao. Cs231n: Deep learning for computer vision. 3, 4

[11] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. 4

[12] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. 2

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. 1, 5

[14] R. Stauder D. Ostler, M. Kranzfelder, S. Koller, H. FeuBner, and N. Navab. The tum lapchole dataset for the m2cai 2016 workflow challenge. *CoRR*, abs/1610.09278, 2016. 2

[15] Theodoros Pissas, Claudio S. Ravasio, Lyndon Da Cruz, and Christos Bergeles. Effective semantic segmentation in cataract surgery: What matters most? *CoRR*, abs/2108.06119, 2021. 1, 3, 6, 7, 9

[16] Ashwin Raju, Sheng Wang, and Junzhou Huang. M2cai surgical tool detection challenge report. *University of Texas at Arlington, Tech*, 2016. 2

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. 2, 3

[18] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. *CoRR*, abs/1902.09212, 2019. 2

[19] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. *CoRR*, abs/1711.11248, 2017. 2

[20] A.P. Twinanda, D. Mutter S. Shehata, J. Marescaux, M. de Mathelin, and N. Padoy. Endonet: A deep architecture for recognition tasks on laparoscopic videos. *IEEE Transactions on Medical Imaging (TMI)*, 2017. 2

[21] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 1, 5

[22] Hsu-Hang Yeh, Anjal M. Jain, Olivia Fox, and Sophia Y. Wang. PhacoTrainer: A Multicenter Study of Deep Learning for Activity Recognition in Cataract Surgical Videos. *Translational Vision Science and Technology*, 10(13):23–23, 11 2021. 2

[23] Michael Zhang, Xiaotian Cheng, Daniel Copeland, Arjun D. Desai, Melody Y. Guan, Gabriel A. Brat, and Serena Yeung. Using computer vision to automate hand detection and tracking of surgeon movements in videos of open surgery. *CoRR*, abs/2012.06948, 2020. 2