Smart Surgical Tools Checker

Vinamr Athavle and Tang Kok Zuea

Abstract According to Singapore Health Services, the largest healthcare group in Singapore, they perform approximately 479 surgeries daily (SingHealth (2018) About us. https://www.singhealth.com.sg/PatientCare/Overseas-Referral/En/AboutUs/Introduction/Pages/Home.aspx. Retrieved, 12 June 2018). Multiple requirements and steps are also involved in instrument processing, including preparation, cleaning, and packaging (Even Cuny and Fiona M. Collins (May 2010). Instrument Processing, Work Flow and Sterility Assurance. Retrieved June 12, 2018). These factors increase the stress that instrument processing department staff face and the propensity of human errors, as most hospitals check surgical tools with their own eyes. The Smart Surgical Tools Checker (SSTC) is an intelligent scanner that will identify the tool on the platform and tally it with a reference toolset. If there is a wrong tool placed, or if there are missing tools, the software, using image processing algorithms and artificial intelligence (AI), will warn the user about the error. This reduces the occurrence of incomplete surgical sets and missing tools in the inventory.

Keywords Artificial intelligence · Machine learning · Image processing · Surgical tools · VGG16 neural network · Surgical tool checker

1 Introduction

The World Health Organization (WHO) released the WHO Safe Surgery Checklist which has played a major role in preventing errors [1, 2] caused by the usage of wrong tools during operations [3]. However, there are many disadvantages of the checklist, such as the duplication of existing checklists which leads to 'checklist fatigue' [4]. Another product is the Ultra High-Frequency Radio Frequency Identification (RFID),

V. Athavle (✉)
NUS High School of Mathematics and Science, 20 Clementi Avenue 1, Singapore 129957, Republic of Singapore
e-mail: h1610153@nushigh.edu.sg

T. Kok Zuea
Faculty of Engineering, National University of Singapore, 9 Engineering Drive 1, #07-26 EA, Singapore 117575, Republic of Singapore

H. Guo et al. (eds.), *IRC-SET 2020*,
https://doi.org/10.1007/978-981-15-9472-4_17

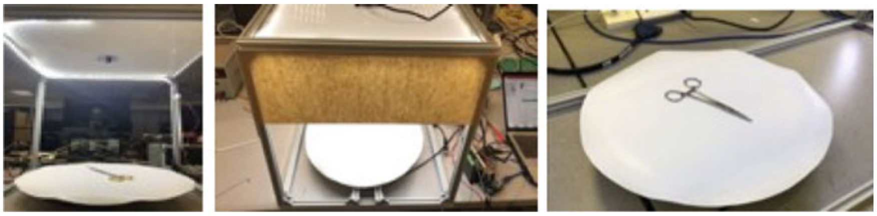

Fig. 1 Images of hardware set-up of surgical tool checker

which keeps track of surgical tools. While it reduces the time wasted and is cost-efficient [5], the signal produced may interfere with sensitive medical equipment.

Previous prototypes from NUS students [6] have been rejected. These systems involved a push-in-pull-out closed system tray, which was hard to sterilize. The SSTC built upon these limitations and thus the system had been made semi-open for ease in sterilization, while preventing ambient lighting from affecting the images captured of the surgical tools.

The SSTC will benefit hospital staff and patients by reducing human errors affecting the surgery and packaging staff can work in less stressful environments and more efficiently.

2 Materials and Methods

2.1 Phase 1—Building of Prototype

The setup used an acrylic sheet ceiling to attach an 8-megapixel camera, and the underside was lined with LED lights along the aluminium rods. To support the structure, aluminium rods were attached to the sheet. A circular rotating disc was placed at the bottom of the structure (with a stepper motor) and covered with a non-slip medical cloth (Fig. 1).

2.2 Phase 2—Coding (Image Processing and Machine Learning)

Turntable: After connecting the stepper motor to the Arduino Uno Microcontroller, we used the Arduino Genuino IDE [7] software to code a program allowing for the turning of the mechanism.

Image Capturing: Using OpenCV [8] and NumPy libraries in Python, we connected the camera to the computer and obtained one picture. A high-resolution red–green–blue (RGB) image was captured by altering external factors such as lighting around the tool. The image capturing was then adjusted such that 1 picture was taken per 1-degree rotation of the turntable. Image processing tools and variation in lighting intensity, opaqueness etc., were used to create 2 or more copies of the same image allowing for a larger dataset for training the model.

Neural Network: We used a VGG16 neural network model to construct a deep learning model. To test the accuracy of the network, we trained it with pictures from Kaggle [9]. We then trained with real surgical tools and improved the accuracy such as by improving the picture resolution.

2.3 Obtaining Data

Experiments were done using a macOS High Sierra with a 2.9 GHz Intel Core i5 processor 64 bit installed with Python version 3.6, without GPU acceleration. We applied the VGG16 neural network and Keras software (with TensorFlow backend) to train the computer using data (pictures of tools) gathered from Phase 2. The accuracy of the model was tested before more functions were applied to improve the accuracy of the computer with more data of tools.

2.4 Getting Accuracy of Model

Given N training samples, x represents the annotated parts of the model while y represents the labels given to the images. After training, the VGG16 model can approximate a model F by mapping out the relationship between the input vectors x and output vectors y. During the forward propagation phase, when a training sample (x_i, y_i) is taken in by the neural network, x_i is fed-forward from the input layer to the output layer. Finally, we get the output o_i. This process can be formulated as, where L is the number of layers in the sequential model, w_j is the weight vector of the jth layer F_j. We define F_j as the convolutional layer which performs operations. After a series of operations by the convolutional layers, estimating the weight vectors w_1, w_2, …, w_L can be solved with the following optimization problem, where is usually defined as cross-entropy loss function.

3 Results and Discussion

We implemented a method of saving the bottleneck features of the image taken by using a data flow generator to convert the images into a NumPy array and a VGG16 Neural Network without the fully-connected top layer to train with the images initially. The second part of the program involved a bottleneck model with two dense layers, and a dropout of 50% to achieve a greater generalization across all convolutional layers for them to be independent of each other and generate a higher validation accuracy. The *sigmoid* activation was used on the last dense layer to generate the predictions of the images we put into the bottleneck model. The model was compiled with the *RMSProp* optimizer and the *binary cross entropy* loss function. Table 1 shows the results of our training and validation at the 100th epoch of two of the runs in our experiments.

From our results in Table 1, the training accuracy was higher than the validation accuracy in the 1st and 2nd runs, indicating that we had overfitted the neural network. This caused the model to learn the specific details of the pictures and random fluctuations in the training data as new concepts such that the performance was negatively affected. This affected the validation as these concepts did not apply to new data and affected the model's ability to generalize. While we achieved a 90% accuracy in both runs, our results improved after replacing the images from Kaggle with surgical tools (Table 2).

This could be due to the effect of generalization, where in the case of the training images, all training data varied a lot from each other in terms of shape, colour and features; whereas for the images of the tools we had taken with the SSTC, it was limited to a small set of images where all the images were similar. As such, the trained model would be able to pick up the salient differences in the images and learn how to differentiate between the tools, thus giving the high training and validation accuracy [10, 11]. However, the accuracy started to decrease significantly when more tools were added for training (Tables 3 and 4).

To improve the accuracy of our model, we adapted multiple functions and physical changes to our program and set-up respectively. We worked with different optimizers to compile the Keras model and different activations for the last and dense layers of the model to obtain a better differentiation curve where the mapping of the given data sets can be separated clearly. We worked on more aggressive dropout and batch normalization to generalize the model as much as possible to prevent complex co-adaptations on training data. Physical changes we made include the usage of better lighting and better internal reflection of the internal setup such that a clearer image is used by the model. We also took more pictures to enlarge our limited dataset, and train the model to pick up changes in the images.

Using our original data and bottleneck model (Table 1) as a benchmark, we made alterations to the code. The original bottleneck model utilized a flattening layer, followed by a dense layer of output size 256, data dropout of 50%, and finally a dense layer of output size 1 with a *sigmoid* function. In the first test, two tools, the crab-claw tool (CCT) and golden scissors (GS) were used. In the second, the CCT, GS and length-one scissors (L1S), and for the third, the CCT, GS and pincer tool (PT). As seen from the results of the first three tests, we can conclude that as the number of images and number of classes put into the model increased, the accuracy of the model decreased.

In test 4, the dropout layer was removed. The accuracy was lower as overfitting had occurred and the neurons in test 3 were able to be co-dependent of each other during training and curb the individual power of each neuron. In test 5, a few more

Table 1 Results of training and validation with VGG16 neural network using pictures from Kaggle

	Training images	Saving time/s	Validation images	Saving time/s	Training time/s	Training loss/%	Training accuracy/%	Validation loss/%	Validation accuracy/%
1st run	10,000	1859.83	5000	925.605	531.106	8.51	0.9750	69.8	88.92
2nd run	10,000	1805.74	5000	916.176	537.016	3.79	0.9910	65.9	90.80

Table 2 Results of training and validation with VGG16 neural network using two tools taken by the tool checker

Training images	Validation images	Training loss/%	Training accuracy/%	Validation loss/%	Validation accuracy/%
10,800	3600	208.56e−07	100.00	109.60e−07	100.00

Table 3 Results of training and validation with VGG16 neural network using four tools (crab-claw tool, golden scissors, length-one scissors and pincer tool) taken by the tool checker

Training images	Validation images	Training loss/%	Training accuracy/%	Validation loss/%	Validation accuracy/%
37,800	12,600	199.02	25.00	196.59	25.00

Table 4 Comparison of results of training and validation with VGG16 neural network using two, three and four tools taken by the Tool Checker at the last (50th) epoch

Number of tools	Training loss/%	Training accuracy/%	Validation loss/%	Validation accuracy/%
2	208.56e−07	100.00	109.60e−07	100.00
3	327.83e−09	33.33	397.36e−08	33.33
4	199.02	25.00	196.59	25.00

dense and dropout layers were added, but the model was slightly less viable. In tests 6 and 7, different amounts of dropout (75% and 25% respectively) caused underfitting of the model, resulting in lower accuracies recorded. In test 8, the size of the output dense layer was modified from 8 to 32 to check if it had an effect on the predictions that were obtained from the final dense layer. However, the accuracy was lower, indicating that too many parameters were present for the prediction layer to have a conclusive accurate classification.

From tests 10 to 13, we tested to see if the size of the output dense layer had an effect on the model again, but using the original model (from test 1). As the size increased from 8 to 32, the accuracy increased from 43 to 63%. However, as the size increased further from 32 to 64 and 512, the accuracy dropped from 63 to 59 and 33%.

From test 14 to 16, we worked with various activation functions of the final prediction layer. With softmax, the accuracy was 33%. With tanh, the accuracy dropped from 59% in epoch 1–37% in epoch 50. With the linear function, the accuracy was at 0.00% indicating that the model was unable to differentiate the tools from each other. As such, the sigmoid function was still the most viable activation function.

Through these results, we concluded that for this specific dataset of tools, the dense layer with an output size of 32 and data dropout of 50% with the sigmoid activation function would be the most efficient and reliable model.

Increasing epoch number and decreasing batch size proved to yield a lower accuracy as seen from test 17, and limits in computational power increase the training time required significantly. Increasing dense and dropout layers with the current model as in test 18 decreased model accuracy (like in test 3 and 4) to 33.33%, and decreasing batch size in test 19 and an intermediate dense layer was added in test 20 with a SoftMax activation function provided negligible change to the accuracy from test 18. Increasing the number of epochs to 1000 (like in test 17) lead to higher accuracy as more time was provided for training, so accuracy increased to 62%.

A method we finally utilized was a data augmenter. We conducted vigorous data augmentation of the images by changing the images' zoom range, flipping the image, altering the RGB channels etc. This yielded an accuracy of about 75%, which was a significant improvement from the previous sets of results, as the more varied set of images present prevented the model from picking up random fluctuations.

Another method was the use of multi-label instead of multi-class classification. This means each image can be grouped together under the same class so the model can identify similar aspects in tools for faster classification as well as higher accuracy due to the neurons learning to focus only on specific parts of the image. We also made the VGG16 smaller by decreasing the convolutional layers in each convolution. More vigorous batch normalization was used to provide better generalization as proven from the tests. This smaller VGG16 also allowed us to save training time, allowing for more efficient, practical usage of the model. With all 3 methods implemented, the results improved significantly (Table 5).

In our machine learning program, a support vector machine (SVM) [12] was being built. Using an SVM classifier enabled classification of data into two or more classes. When training, the SVM builds a model before mapping the decision boundary for each class, and specifies the hyperplane that separates the classes. Increasing the hyperplane margin improves the classification accuracy. As such, SVM can be used to effectively perform non-linear classification. The model needs to know what input shape it should expect. Thus, the first layer in a sequential model needs to receive information about its input shape. This input image is a placeholder tensor that contains generated images and will be put into the network for training. A convolutional neural network (CNN) is comprised of one or more convolutional layers, followed by one or more fully connected layers as in a standard multilayer neural network.

First, we know that deep learning needs a large amount of labelled training data, and second, the surgical tools are very similar to each other and in order to differentiate them, a high-resolution image would be preferred such that the hyperplane margin can increase in distance. Ambient lighting as well as reflectivity of the turntable play an essential role in ensuring the correct identification of the tools into their classes.

Table 5 Results of training and validation with VGG16 neural network using seven tools at the last (75th) epoch with data augmentation and multi-label classification

Number of tools	Training loss/%	Training accuracy/%	Validation loss/%	Validation accuracy/%
7	0.44	99.84	27.25	92.05

As such, we need to find the optimum point at which both lighting and reflectivity can be balanced to ensure a high-quality image produced. Our final model was also able to balance the resolution of the image by altering the RGB channels of the image such that it would be optimized for input into the neural network model. Our dataset this time consisted of 7 tools, length-0 scissors (L0S), L1S, length-2 scissors (L2S), GS, PT, ST and CCT.

4 Conclusion and Future Work

The SSTC is able to identify tools accurately with a low-cost production. By minimizing human errors, it will greatly reduce the time wasted that will affect the surgery being carried out. Unlike similar projects, the SSTC will enable the streamlining of hospital procedures in a more efficient manner. Some improvements for the future include creating a larger platform such that more tools can fit onto it such that the speed of processing and registering the tools will be increased. Some factors that we did not work on would be the use of a different table below the setup, as that could affect the image taken by the camera. We could have adapted a Faster R-CNN model which would be able to bypass the problem of having to select a huge number of regions for identification. Finally, we also hope to work on a Graphical User Interface (GUI) such that the SSTC can be more user-friendly and catered to the hospital staff, rather than be hindered by complicated lines of codes which may be irrelevant to them.

Acknowledgements We would like to thank Dr. Tang Kok Zuea for his constant guidance throughout this project. This project would not have been successful without his patient understanding and guidance.

We would also like to acknowledge Ms. Winni Valentina and Mr. Jonathan, graduates from NUS, for providing us with deep insight into the project, and guiding us on how to sieve out ideas for a better prototype.

Next, we would like to thank Mr. Lim Teck Choow, the teacher-in-charge for all SMP teams in NUS High. Without his dedication and constant reminders of the many deadlines, we would not have been able to achieve success in this project. A special thanks to our lab officers, Mr. Tan Chee Siong, who provided us with space in the lab for us to do our research and the necessary resources required to conduct our experiments, as well as Mr. Richard Yip and Mr. Justin Yip for the help and advice they have given us on the design of our prototype.

Lastly, we would like to acknowledge our family members for their support while we worked on this project. Their support has given us the motivation to continue to work hard and strive for the best.

References

1. SingHealth (2018) About us. https://www.singhealth.com.sg/PatientCare/Overseas-Referral/En/AboutUs/Introduction/Pages/Home.aspx. Last retrieved, 12 June 2018.

2. Even Cuny and Fiona M. Collins (May 2010). Instrument Processing, Work Flow and Sterility Assurance. Retrieved June 12, 2018.
3. World Health Organisation (2018) WHO surgical safety checklist and implementation manual. https://www.who.int/patientsafety/safesurgery/ss_checklist/en/. Last retrieved, 13 June 2018.
4. Dr. Natasha Woodman (5 February 2016) 325 WHOSurgical Safety Checklist. Retrieved June 12, 2018.
5. Xerafy (2018) Usage of RFID Surgical Instruments Tracking Established to Increase Patient Safety. https://www.xerafy.com/blog/usage-of-rfid-surgical-instruments-tracking-established-to-increase-patient-safety/. Last retrieved, 12 June 2018.
6. SmartSurgicalTool Checker(2017),SohWJ,NUSSchoolOf Engineering. Retrieved June 12, 2018.
7. Arduino (n.d.). https://www.arduino.cc/. Last retrieved, 12 June 2018.
8. OpenCV library. (n.d.). https://opencv.org/. Last retrieved, 13 June 2018.
9. Kaggle: Your Home for Data Science.(n.d.). https://www.kaggle.com/. Last retrieved, 12 June 2018.
10. Kim et al. Performance analysis of CNN frameworks forGPUs. in Proceedings of 2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2017, pp 10–13. Last retrieved, 16 June 2018
11. Li, Hailiang, et al. An improved deep learning approachfor detection of thyroid papillary cancer in ultrasound images. *Scientific Reports*, 26 Apr. 2018. Last retrieved, 12 June 2018
12. Lin, Y. et al. Large-scale image classification: Fast feature extraction and svm training. in Computer Vision andPattern Recognition, 1689–1696 (2011).