Assignment 2 Report ZOJP

Zihui Ouyang, zo2151 Jianfei Pan, jp4201

***Abstract*—Classifying surgical phases is a time-consuming task, yet using large pre-trained model the task can be made much easier. It was found out that using EfficientNet as the pre-trained model the classification accuracy can reach 93%.**

I. Introduction

Minimally invasive surgery(MIS) is a common and preferred method technique over open surgery since it causes less pain. However, since it only operates in a very finite space, it is therefore difficult to master and require considerable amount of training. Assessment of surgical skills is routinely required, so surgeons need to look at videos. However, videos are often 30 minutes long and there are often hundreds of video to analyze, which makes watching them one by one infeasible. Therefore, in this report, we propose several machine learning methods to automatically classify phases of a surgery that is recorded in the video. The model can be used to improve the efficiency of surgical performance screening, as well as providing educational support to novices.

II. Data Extraction and Processing

Our data set is prepared by professional doctors, which contains more than a hundred surgery videos of varying lengths. Out of those, there are 70 videos with annotations, where the doctor documented the time intervals for the 14 different phases in total for each video in a csv file. All videos have single frames per second, and thus we firstly convert all frames from the videos into images. The raw frames have size of 768 by 480, but we resize them to various sizes for our different models. For example, each frame is resized to 224 by 224 for our transferred Alexnet model. All frames are then normalized and converted to tensors and then fed into our Pytorch models. The names of operations are extracted from the csv file for each frame in the video and compiled as the labels for our video frame classifications. When we processed the data, we found that there were 35 misspelled labels whereas there should be only 14 distinct classes. We resolved this issue by manually inspecting and attributing the misspelled labels to their correct counterparts. After this process, we eventually were able to gather 14 distinct labels in total. Additionally, we found that our dataset is very unbalanced, as in frames for some classes are way more than others. We initially thought we could augment the data via standard image augmentation techniques such as adding noise, random cropping, rotation and flipping and etc. However, due to our limited storage and time, the team could not augment the data extensively, which will be of better interests in future works. Instead, the team tried different sampling techniques to deal with the unbalanced dataset, which will be explained further in late sections.

TABLE I
Summary of baseline and first models

Layer	Size
Convolutional	5×5 kernel
Maxpool	2×2
Convolutional	5×5 kernel
Maxpool	2×2
Fully Connected(FC)(or LSTM)	120
FC	84
FC	14

III. Proposed System Designs

The team had extensive brainstorming and analysis on the problem at hand before building our models. After reviewing several literature, we found that recent year papers frequently adopted various deep neural network models to perform similar tasks, as previous attempts have explored the usage of conventional machine learning techniques such as Hidden Markov Models and etc. Thus, we decided to start from building neural networks to tackle this problem. We considered both temporal models and non-temporal models, which will be discussed in details in this section.

A. Baseline Design

Our baseline has 2 convolutional and 3 fully connected layers, whose layer structure is detailed in Fig.1. The reason we chosen this as our baseline model is because it is very simple and easy to train, and convolutional and fully connected layers are the basic building blocks for our more complex models.

Our first model is a temporal model that has 2 convolutional, 1 LSTM and 2 fully connected layers, whose architecture is detailed in Table I. We added an LSTM layer to take the temporal relations of video frames into consideration, due to the video frames' sequential nature. The numbers beside the fully connected layers indicate the output size.

B. Proposed Design II - Transfer Learning with AlexNet, TAlex

Since our dataset is too large for us train a complicated model from scratch, we think using existing power models to extract key features from the image, which we then use to classify the images, will be a viable approach. The following models all utilize the techniques of transfer learning with different models. Our second model adopts the transfer learning methodology to use AlexNet [2] as the feature extractor for input frames and only the last layer of the AlexNet is trainable.

TABLE II
SUMMARY OF BASELINE AND FIRST MODELS

Layer	Size
(Frozen) Alexnet.features[:-2]	Stacked Conv2d and Maxpool2d
(Train) Convolutional with RELU	3 × 3 kernel
Maxpool	3 × 3 and stride = 2
AdaptiveAvgPool	Output size = 6 × 6
classifier, Dropout	p = 0.5
(Train) Linear with RELU	4096
Dropout	p = 0.5
(Train) Linear with RELU	4096
(Train) Linear with RELU	1000
(Train) Linear	14

TABLE III
SUMMARY OF TRANSFER LEARNING WITH EFFICIENTNET, TEFFI

Layer	Size
(Frozen) efficientnet.features[:-1]	1 × 1
(Train) Conv2d × 11	Multi-head Attention + FC
(Train)BatchNorm2d	2560
SiLU	inplace=True
classifier Dropout	p = 0.5
(Train) Linear with RELU	1024
Dropout	p = 0.5
(Train) Linear with RELU	512
(Train) Linear	14

We added a customized classifier on top on the AlexNet feature extractor, which are trained together. We call this model TAlex, whose architecture is illustrated in Figure 3.

C. Proposed Design III - Transfer Learning with EfficientNet

Our third model adopts the transfer learning methodology to use EfficientNet [4] as the feature extractor for input frames. Similar to TAlex, we only trained the last layer of the EfficientNet. We added a customized classifier on top on the EfficientNet feature extractor, which are trained together. We call this model TEffi, whose architecture is illustrated in Figure 4. The reason of choosing the EfficientNet over other popular models is that it requires much fewer parameters to achieve promising results. Due to the hardware limitation, we favor models that are more efficient to train at the benefit of having fewer parameters.

D. Proposed Design IV - Transfer Learning with Vision Transformer - small model

We also use the pretrained Vision transformer [1] as a feature extractor. For the similar reason, vision transformer is used because it is highly computationally efficient and pretrained vision transformer has proven to reach or even outperform the state-of-art convolutional neural networks. We firstly used the base vision transformer model that has 11 encoder layers as the feature extractor and stack our classifier on top of. We call this model TViT-B, whose architecture is illustrated in Figure . The team also tried training the last multi-head attention block together with our classifier, but the results seemed worse in the first five epochs that examined.

TABLE IV
SUMMARY OF TRANSFER LEARNING WITH VIT-B

Layer	Size
(Frozen) ViT.conv$_{proj}$	32 × 32 kernel
(Frozen) ViT.encoder × 11	Multi-head Attention + FC
classifier(head) Dropout	p = 0.5
(Train) Linear with RELU	512
Dropout	p = 0.5
(Train) Linear with RELU	128
(Train) Linear	14

TABLE V
SUMMARY OF TRANSFER LEARNING WITH VIT-L

Layer	Size
(Frozen) ViT.conv$_{proj}$	32 × 32 kernel
(Frozen) ViT.encoder × 23	Multi-head Attention + FC
classifier(head) Dropout	p = 0.5
(Train) Linear with RELU	512
Dropout	p = 0.5
(Train) Linear with RELU	128
(Train) Linear	14

E. Proposed Design VI - Transfer Learning with Vision Transformer

Our last model is similar to model IV except that we now use a much larger and complex pretrained vision transformer model, which we call TViT-L. This model has 23 encoder layers of stacked multi-head attention blocks and consequently has much more parameters than the base version. The classifier design is the same as our TViT-B model, and thus its architecture is very similar to TViT-B except it has more layers and parameters. The TViT-L architecture is shown in Figure 6.

IV. TRAINING AND OTHER IMPLEMENTATION DETAILS

In our training, we used Adam optimizer with a learning rate of 0.001, and batch size of 32. We used both random sampler and weighted random sampler to load our data. In weighted random sampler, the weights are determined as the inverse of the number of images that each class has. In this way, we hope we can sample images that appear less often to get sampled more frequently. However, we do think a better approach to deal with unbalanced dataset is to augment the scarce data, because we observe that weighted random sampler occasionally performs worse than the vanilla random sampler. We decided to train all of our models for five epochs, because training each model often takes several hours on our machines. We believe training for five epochs can still provide us with insights as we can clearly observe the trends of training accuracy for our models at five epochs and reason about it. The team does believe that training more epochs will make our analysis more compelling, since it takes time to properly fine tune layers of pretrained models for our specific task.

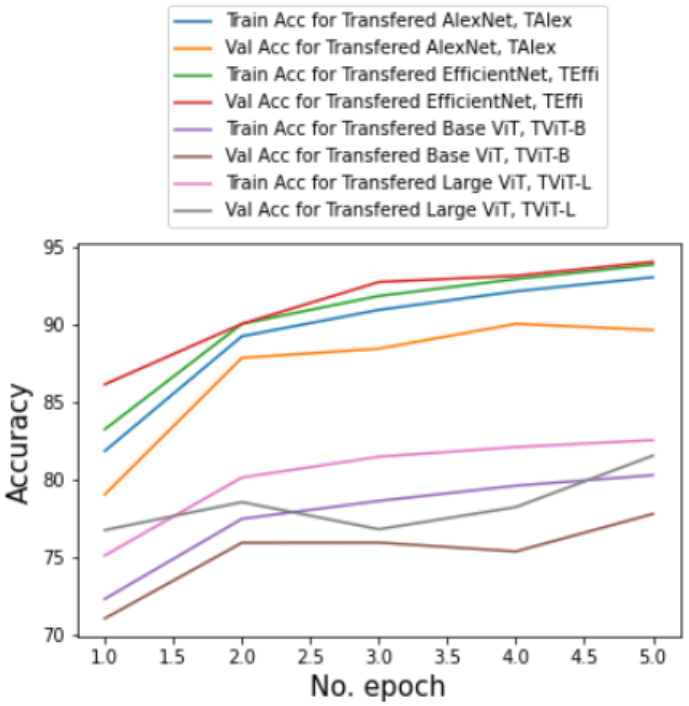

Fig. 1. Accuracy history comparison for all transfer learned models.

V. Results from experimentation

A. *Accuracy / Macro F1 score / Class-wise Precision & Recall / AUC*

Fig. 1. compares the training and validation accuracy for all of our transfer learning models. We observe that different models produce considerably distinct learning curves for our problem. It is initially to our surprise that models using pretrained vision transformer performs very poorly in our problem. Models built on transfer learning with Alexnet and EfficientNet produce much higher learning curves for both training and validation dataset than the transformer based models, which shows that the CNN based feature extractors with inductive bias can capture the key features in our frames more effectively. Additionally, vision transformer splits images into patches and use positional encoding to ensure learning the spatial information in the images, whereas in our problem many classes share very similar backgrounds and thus produce similar patches which could introduce a source of confusion to the transformer model. However, the learning curves are not entirely convincing. Firstly, the CNN based models may have already overfit to the training and validation dataset. Secondly, the number of epochs is still too small to see how different models learn in the long run, and thus more epochs of training is desirable. Thirdly, we fine tuned layers in the pretrained Alexnet and EfficientNet, whereas we did not do in the transformer networks. One reason is because we tried fine tuning the last encoder layer but the results for the five epochs barely reached our baseline model. Second reason is that transformer is known to be very data thirsty and thus may need more data if we were to fine tune certain blocks of the transformer model. Nevertheless, we believe fine tuning transformers could yield promising results for longer training horizon and augmented dataset due to their staggering number of parameters.

TABLE VI
Comparison of F1-score

Model	F1-score
Baseline	0.68
Baseline and weighted sampler	0.77
CNN + LSTM	0.82
TAlex	0.87
TEffi	0.92
TViT-B	0.71
TViT-L	0.75
TEffi with test set	0.76

Table. VI. compares the different F1 score for our different models. TEffi has the highest F1 score of 0.92 among all of our models. This score is as expected, because EfficientNet is known to be the state-of-art image classification model and this model has many clever designs such as ConvNormActivation and SqueezeExcitation layers. Moreover, it has very high model capacity, which allows it to learn complex features efficiently. However, our initial test score of this model on the holdout test dataset was 0.76, which shows that TEffi could have been overfit to both our training and validation dataset, since our F1 score is evaluated on the validation dataset. There are also surprises in the results. TViT-L has higher F1 score than TViT-B, which is as expected since TViT-L is a much larger model and has much higher capacity. However, the fact that using them as pretrained feature extractors barely exceeds our baseline model, and is way worse than Alexnet and EfficientNet feature extractors. We think one way to improve upon it may be to fine tune several encoder blocks of the vision transformer model over augmented dataset together with our classifier, since we know transformer models are known to be data thirsty. Our simple CNN + LSTM model does very well when comparing with other more complex models, which shows that temporality is an important factor to be considered.

Class-wise recall and precision for all of our models are summarized in Table. VII. and Table. VIII. Overall, different models have their strength an weakness in determining different phases, and we think the architecture of the model plays the most important role here. TEffi has the best overall performance for all classes and we will mainly relate our discussion to TEffi in particular. However, we noticed that TEffi performs worse on some classes such as the stationary idle, transitory idle and mesh positioning phase compared to other classes. The team thinks the reason is that these phases can be easily confused with other data if without any temporal information. For example, transitory idle exists in between any two phases but without any movement. This means that frames in this state can easily confuse the TEffi with other states, since TEffi can only predict the class of the frames

TABLE VII
VALIDATION CLASS-WISE PRECISION

Labels/Models	baseline	baseline with weighted sampler	CNN+LSTM	TAlex	TEffi	TViT_B	TViT_L
adhesiolysis	1	0.98	0.83	0.94	0.88	0.67	0.62
blurry	1	0.81	0.88	0.94	0.95	0.49	0.39
catheter insertion	0.8	0.94	0.91	0.87	0.93	0.5	0.63
mesh placement	0.8	0.96	0.93	0.94	0.96	0.88	0.9
mesh positioning	0.81	0.84	0.46	0.94	0.68	0.88	0.9
out of body	0.96	0.86	0.89	0.52	0.97	0.21	0.28
peritoneal closure	0.95	0.93	0.84	0.96	0.98	0.91	0.91
peritoneal scoring	0.99	0.83	0.72	0.88	0.95	0.68	0.75
positioning suture	0.73	0.29	0.87	0.65	0.75	0.45	0.55
preperitoneal dissection	0.68	0.8	0.86	0.9	0.89	0.81	0.89
primary hernia repair	0.96	0.94	0.96	0.95	0.93	0.73	0.81
reduction of hernia	0.87	0.96	0.91	0.94	0.98	0.83	0.8
stationary idle	0.88	0.9	0.95	0.9	0.86	0.77	0.79
transitionary idle	0.62	0.39	0.51	0.85	0.87	0.67	0.75

TABLE VIII
VALIDATION CLASS-WISE RECALL

Labels/Models	baseline	baseline with weighted sampler	CNN+LSTM	TAlex	TEffi	TViT_B	TViT_L
adhesiolysis	0.33	0.81	0.87	0.9	1	0.98	0.98
blurry	0.11	0.94	0.83	0.83	1	0.94	0.94
catheter insertion	0.72	0.69	0.81	0.94	0.95	0.9	0.9
mesh placement	0.69	0.75	0.9	0.96	0.97	0.87	0.9
mesh positioning	0.94	0.32	0.71	0.87	0.88	0.34	0.87
out of body	0.93	0.97	0.97	0.98	0.98	0.99	0.98
peritoneal closure	0.84	0.66	0.97	0.93	0.95	0.81	0.88
peritoneal scoring	0.19	0.89	0.95	0.97	0.96	0.94	0.93
positioning suture	0.6	0.9	0.44	0.85	0.89	0.7	0.7
preperitoneal dissection	0.94	0.8	0.79	0.91	0.98	0.63	0.6
primary hernia repair	0.84	0.96	0.93	0.99	1	0.99	0.99
reduction of hernia	0.93	0.63	0.79	0.94	0.92	0.88	0.94
stationary idle	0.92	0.97	0.98	0.99	0.99	0.99	0.99
transitionary idle	0.57	0.59	0.74	0.54	0.74	0.35	0.39

based on objects in the picture instead of information from this sequential process. In the future, we can add more temporal architecture to the TEffi, so that it can learn stationary idle phase also taking into account its relationship with several frames prior to it. Another interesting results we observe from Table. VIII. is that TEffi has 1 in recall for adhesiolysis, blurry, and primary hernia repair phases, whereas their corresponding precision scores weren't outstanding. This indicates that our model predicts these classes more often than needed, since they almost does not have any false negative predictions. This behavior can be explained from the fact that they all have very distinguishing features such as being blurry, or containing instruments in the scene. Our model overfits to these features due to their dominance in the dataset and predict these classes more often whenever similar features arise.

TABLE IX
COMPARISON OF INFERENCE SPEED

Model	Inference Speed (FPS)
Baseline	141
CNN + LSTM	153
TAlex	141
TEffi	122
TViT-B	24
TViT-L	17

B. Inference speed for each video

Table. IX. compares the inference speeds for our different models. The baseline model gives out the fastest inference time, because it has the simplest structure and smallest number of parameters. Vision transformer provides the slowest inference since it has the largest number of parameters. The general trend seems to be that the inference time scales with

TABLE X
SUMMARY OF MODELS' MEMORY FOOTPRINT

Model	Params	Trainable	Memory	Temporal
Baseline	21.6M	5.4M	60MB	No
CNN + LSTM	86.6M	21.6M	316MB	Yes
TAlex	61M	59M	456MB	No
TEffi	67M	7M	266MB	No
TViT-B	88M	461k	335MB	No
TViT-L	306M	592k	1130MB	No

the size of the model, which means that higher number of parameters often result in relatively slower inference speed. However, CNN + LSTM has more parameters than the TAlex and TEffi, whereas it has faster inference speed than the other two models. The team thinks number of layers also affects the inference speeds, as our CNN + LSTM only has four hidden layers, which is significantly smaller than the number of layers in TAlex and TEffi. Overall, TEffi is our model of choice, since it has relatively fast inference speed and the best model performance.

C. Model's memory footprint

Table X compares the sizes and memory footprints of our models. It demonstrates that models differ vastly from others in the number of total parameters, pretrained parameters, and final size of the model. Transformer models are generally larger and TViT-L using large vision transformer as the feature extractor has a model size of staggering 1.13 GBs, although they are shown to be more computationally effective than CNNs [1]. Transfer learned vision transformers have much smaller trainable parameters, since we only train our classifier while freezing all the pretrained weights in the encoders. In contrast, we also fine tuned a few convolutional layers from the original models in TAlex and TEffi models. In our transfer learned models, a large portion of memory footprint comes from the pretrained architectures. Overall, TEffi is the most memory efficient model of all, because it has comparatively small model size for storage, while having high model capacity and the best performance during our training and inference. TEffi is the recommended model to be investigated further, as it is the most efficient model while yielding the best model performance during inferences.

VI. DISCUSSION - INFERENCES AND PLAUSIBLE EXPLANATIONS FOR THE EXPERIMENTS

In this section, we inspect some results from our inferences on the validation dataset for our best model, which is the TEffi. We also visualize the original images with the Grad-CAM [3] on top of it. The Grad-CAM helps us identify regions that most activate the maximum predicted class, which means that it shows us where the model focuses the most on to decide which operation this image belongs to. Then, we also visualize some convolutional kernels to get a better sense of what our model learns from our data.

A. Visualization of correct and incorrect Inferences with Grad-CAM

Since we have over 14 classes, we do not have enough space to present our analysis on all of our classes and prediction failures among different classes often occur irregularly. Therefore, we focuses our studies on only several classes that are most representative of this diverse issue to facilitate our discussion. We applied a gradient hook on the last convolutional layer during our forward pass of the model. This last convolutional layer is the last layer of pretrained EfficientNet, and also the only trainable layer before we feed the results into our fully connected classifier. The gradient hook record the gradients for the maximum predicted class with respect to the output of this layer. The gradients are averaged along across all dimensions except the channel dimension. Then, activations at output of the last convolutional layer are scaled with the mean gradients. Then, finally, the Grad-CAM heatmap is derived by averaging the activations along the channel direction and applied a RELU to get rid of the negative activations and then normalize once more by the maximum activation in the map. The heatmap is superimposed on the input images and provides us with insights into the performance of our different models. Here, we only perform Grad-CAM analysis on our best model, TEffi.

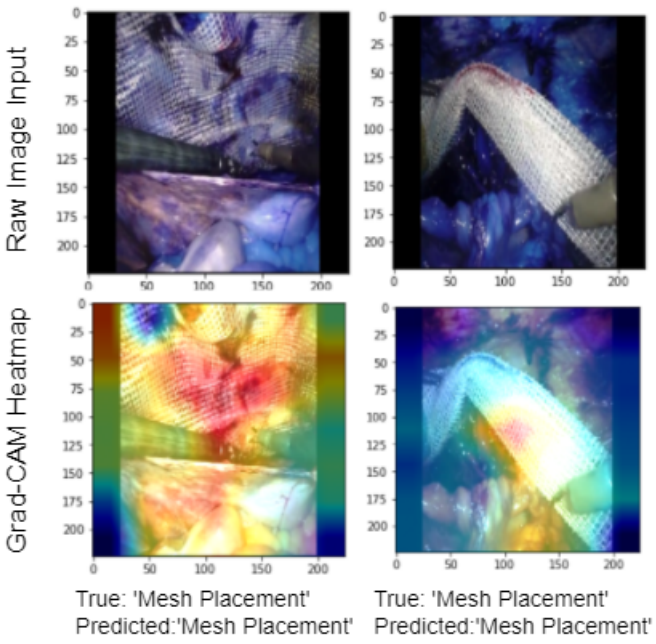

Fig. 2. Inference for the mesh placement class.

Mesh placement and out of body classes are the top two classes that have the highest validation accuracy, nearly 97 percent for each. We inspect the prediction results on the mesh placement class in Figure 2. As the name of the class suggests, this step involving placing the mesh in the correct location, and indeed both two frames have meshes in the predominant spot. The Grad-CAM heatmap provides us more insights into our

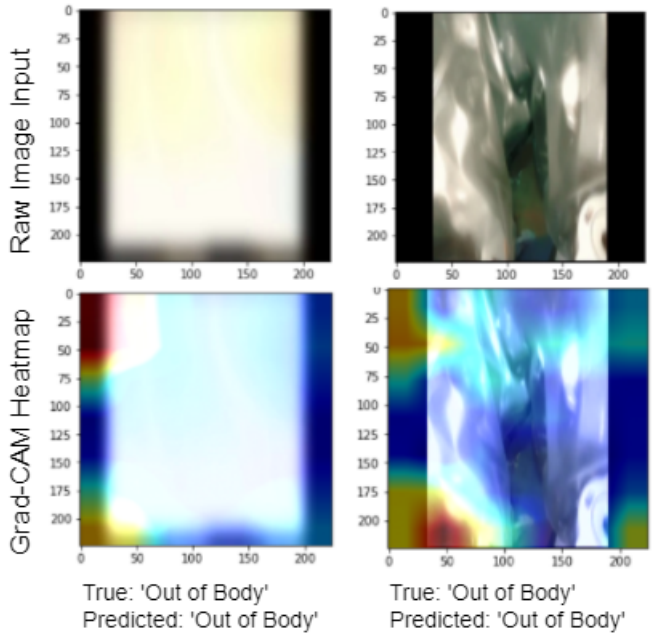

Fig. 3. Inference for the out of body class.

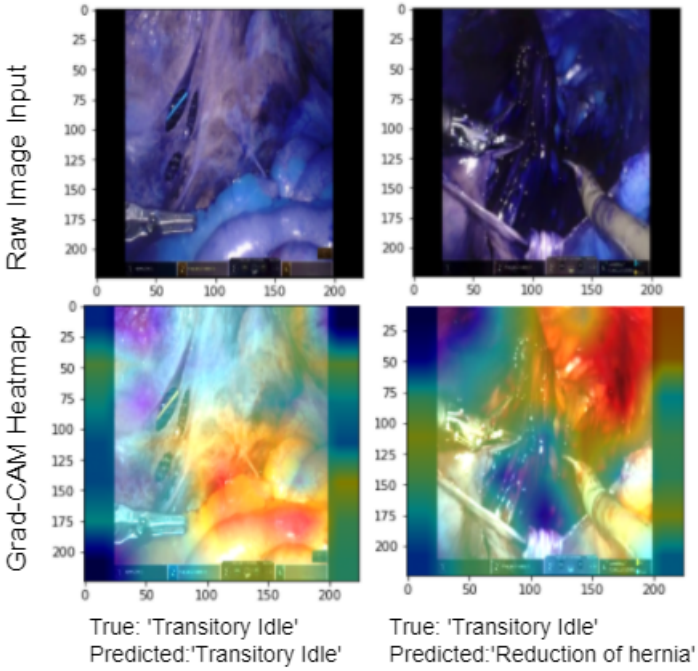

Fig. 4. Inference for the transitory idle class.

model's focal regions. For both images, we see that the mesh region mostly activate our model to predict the class correctly, which shows that our model is able to identify salient features for the mesh placement class and thus have high accuracy for this class. However, the other highly confident prediction for class out of body does not reveal that our model has learnt as much useful information. Figure 3. shows that although this class has very high prediction accuracy, but the model actually focuses on the surrounding black ground, which is the noise that could have been removed in our prior processing. This means that our model is learning the wrong features regarding this class and, if we change the black background to other colors, the model could potentially be misled. Thus, the out of body class is an example that shows our model is not able to capture key features in the images, whereas saturated on the bias in the images, which would influence the model's generalizability.

Then, we look at examples that are predicted incorrectly. Our model has one of the poorest performance on the transitory idle class and Figure 4. show a correct prediction of this class and also an erroneous one. In definition, transitory idle is any idle time between any two clearly defined segments where there is no movement of the instrument. As complex as it sounds, transitory idle exists in any two consecutive phases but with no motion. Our current best model TEffi is not a temporal model and has inherent disadvantages in dealing with phase that requires necessary temporal information. Therefore, Figure 4. shows that our model seems to focus on random portions of the frames when predicting this class. The model predicts the transitory idle phase as the reduction of hernia phase, since confusions arises when instruments and opera-

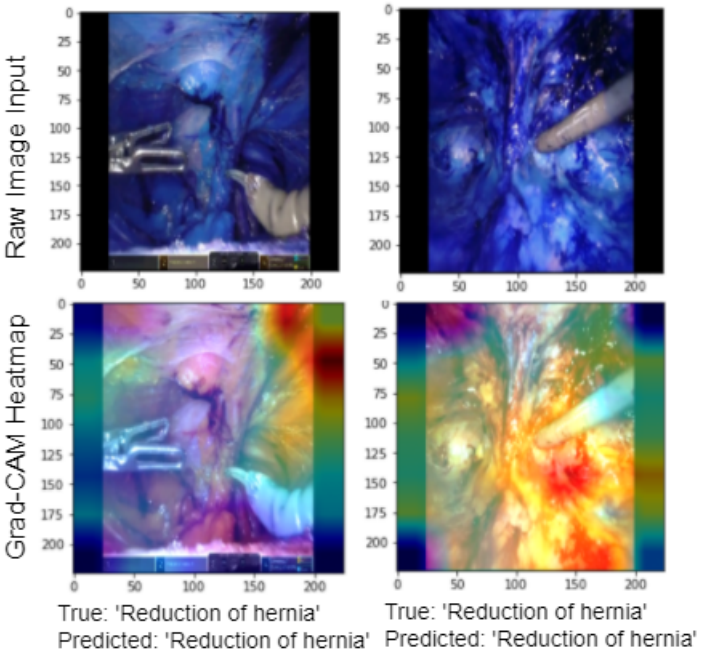

Fig. 5. Inference for the reduction of hernia class.

tions in the frame is very much identical to another phase but the model does not realize that this process is stationary. Thus, we propose we can add in temporal components into our transfer learned model in future practices. To further lead to our conclusion, we also inspected the inference results for the reduction of hernia phase. Image on the left in Figgure 5. shows that our model does not properly but focus on the edge of hernia content, whereas the image on the right shows that our model does focus on the hernia content. Although reduction of hernia class has one of the highest prediction accuracy of 92 percent, it shows that our model still needs extensive training to fully grasp key features in this phase.

B. Visualization of Feature maps after different layers

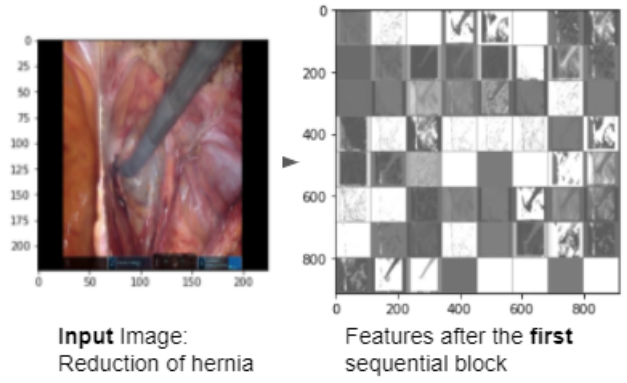

Fig. 6. Feature map after the first block of the EfficientNet.

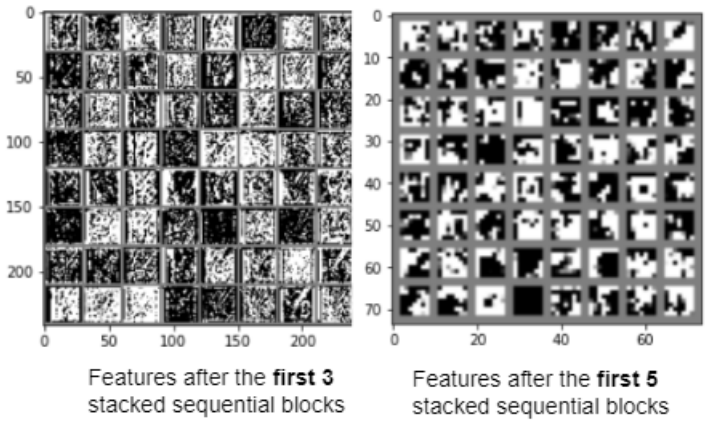

Fig. 7. Feature map after different blocks of the EfficientNet.

In this section, we will analyze features after different building blocks of our transfer learned EfficientNet model to help us infer how the pretrained EfficientNet encode input images before we pass them into our classifier. Figure 6. shows the output from the first sequential building block in the pretrained EfficientNet consisting of a single ConvNormActivation layer. We can see that the input frames are abstracted to patches of features, where at this stage we can still interpret their meanings. For example, some patches show

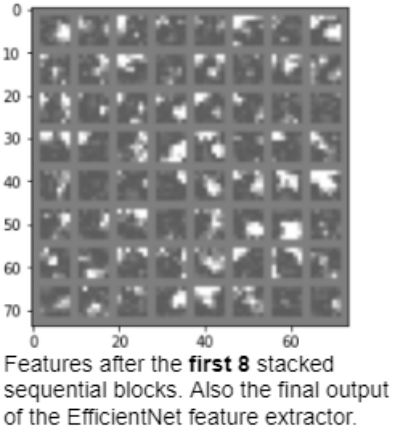

Fig. 8. Feature map after the last block of the EfficientNet.

the silhouette of the grasp, and some patches learning the contrast and fine details like these. However, as we go deeper into the network such as shown in Figure 7., we see that features after each layer becomes more and more abstract and coarse, and the representation of images has become more and more difficult to qualitatively describe. The output of the last layer shown in Figure8 has become incomprehensible and these are the high dimensional abstract features of the original images that we will then flatten and feed into our classifier network. This small experiment has demonstrated that pretrained EfficientNet is very powerful in transforming an image into its high dimensional abstracted representation, where meaningful features can be extracted more efficiently than training a feature extractor from scratch.

VII. Conclusion and Future Works

In this project, we applied various deep neural network models to identify stages in a minimally invasive surgery recording. We have proven the feasibility of using both temporal and non-temporal models for this task, as they both show promising results. In this report, we have tried over six different models and document their effectiveness in our task at hand, and thus we have limited resource to train all our models extensively, but still we do see a general trend that the model performance scales with the number of parameters in the model. However, more powerful models are prone to overfitting on data, and thus the trade-off between model complexity and accuracy is still needed to be investigated in the future. For future works, the team thinks it will be more beneficial to train our models for longer horizon and compare results and, if possible, we can also document and compare their test accuracy. Additionally, we could also augment our data to get a more balanced dataset, ideally at least 2000 images for each class. If we have more powerful hardware, we can investigate the effects of combining temporal models with transfer learned model such as feeding the outputs of transfer learned efficient net into stacked LSTMs. Overall, the team thinks there are many promising aspects that we can continue exploring in future studies.

REFERENCES

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[3] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[4] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.