Surgical Data Science - Final Project

Tal Ifargan and Ziv Keidar

March 2023

1 Introduction

In this project we are trying to tackle the challenging task of Action Segmentation, specifically we are trying to segment videos of participants performing multiple suturing tasks on open surgery simulator to 6 different possible specific gestures. Hence, we will refer to this task as Gesture Segmentation from now on. The unique data was collected by Goldbraikh et al. [Goldbraikh et al., 2022a] and contains 100 videos and kinematic sensor data.

This task is unusual in the landscape of computer vision in general and in surgical data science and computer assisted surgery in particular because the data is hard to collect and there aren't many works dealing with the unique challenges that this task brings. Some of the challenges include large amount of occlusions of recorded subject, rapid replacements of surgical tools and in general small amounts of annotated data [1].

2 Data

As mentioned in the Introduction, the data we had included 100 videos of participants performing multiple suturing tasks from two different angles, top and side. In addition the available data also included kinematic sensor data from 6 different sensors. We were also kindly given per-frame features extracted using EfficentNet [Tan and Le, 2019] trained on the task of Gesture Segmentation. In our work we decided to focus on the feature extractor and test how modified training task can improve the performance of the system on the task in mind. Thus, we only used the video data from the side view, as was used in the given per-frame extracted features.

3 Architecture

The architecture we chose to use is comprised of two modules that trained separately, the first module is the feature extractor which is a pre-trained EffiecentNet-B0 [Tan and Le, 2019], the smallest version of EffiecentNet. The EffiecentNet-B0 architecture is pretty simple and detailed in Figure 1.

The pre-trained model was trained on Imagenet [Deng et al., 2009], a very large annotated dataset for a classification task of 1000 classes.

The second module is ASFormer [Yi et al., 2021] which is a strong transformer model that takes in the features extracted from each frame using the first module and through a refinement process outputs the final prediction. The ASFormer has a one encoder and three decoder modules, each with 9 layers, where each of the modules outputs a prediction, which then gets refined in the following steps. The architecture of ASFormer is of an encoder-decoder transformer with dilated convolution instead of feed forward layer, followed by non-linearity (ReLU) and self-attention or cross-attention in encoder and decoder respectively. The architecture is presented in Figure 2.

[1] The code for reproducing our results can be found at our github repository `https://github.com/TalIfargan/ASFormer`

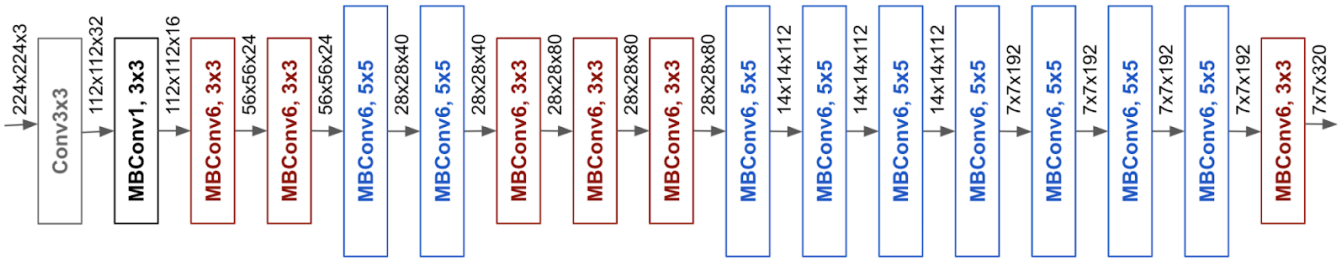

Figure 1: The architecture of EffiecentNet-B0, MBConv blocks are inverted residual blocks, they are blocks that use similar idea to residual blocks but the order of a narrow $->$ wide $->$ narrow structure in contrast to the wide $->$ narrow $->$ wide structure of residual block, hence the name inverted.

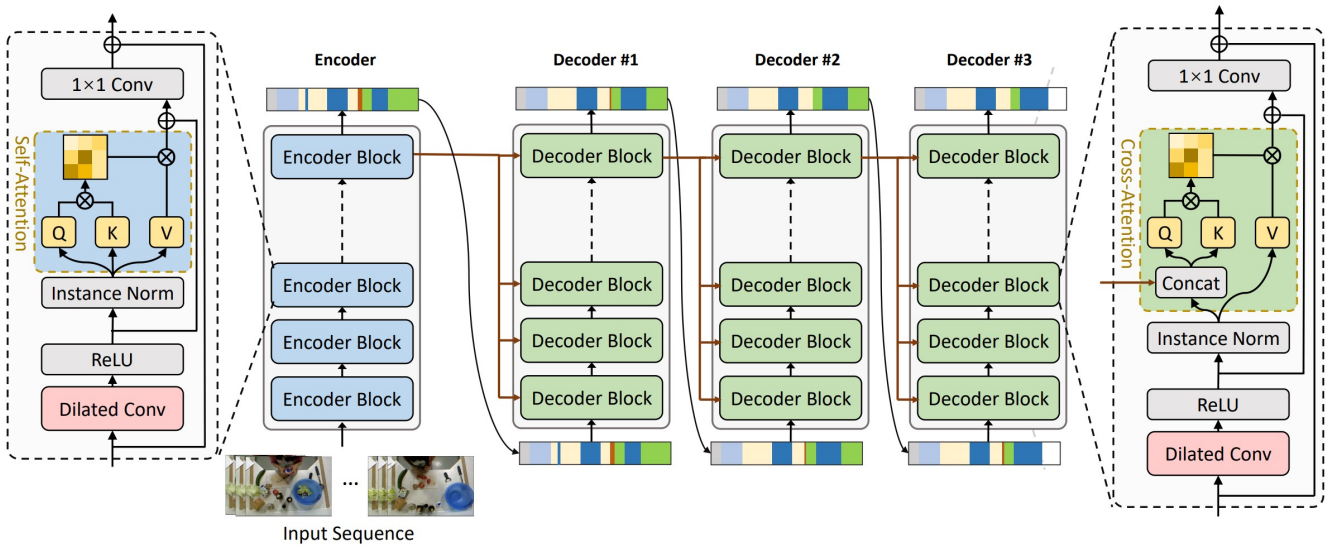

Figure 2: The architecture of ASFormer. The order of feed forward and attention is flipped from the original transformer block and the feed forward is replaced with dilated convolution layer.

As mentioned, each one of the modules was trained separately, the train pipeline detailed in Figure 3.

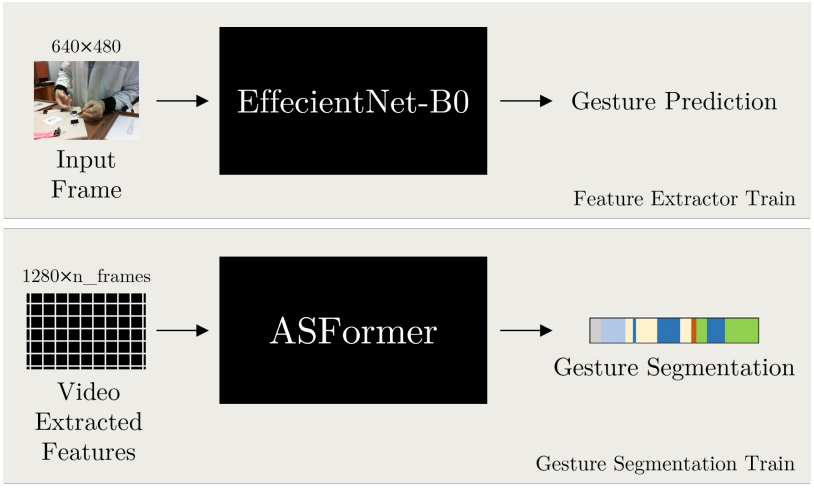

Figure 3: The train pipeline for each of the modules

The complete architecture is detailed in Figure 4.

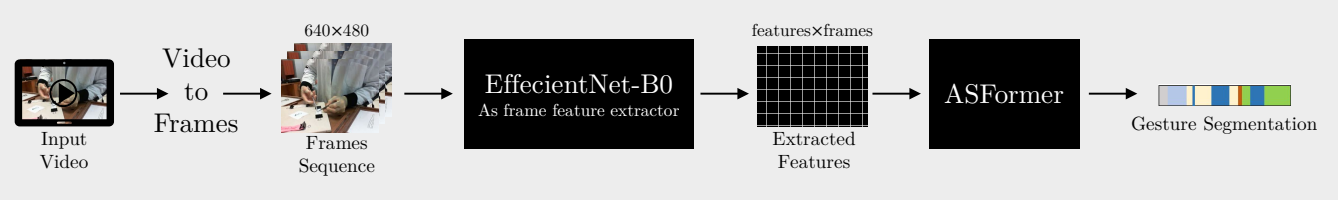

Figure 4: The complete architecture of the model. Input video is split to frame sequence, the frames go as batches into the EffiecentNet-B0 feature extractor, the output features enters ASFormer to output the segmentation result.

3.1 Changes in Architecture

As mentioned, we were given pre-extracted features from the side view videos that were extracted using a pre-trained EffiecentNet. We used this features to train ASFormer to output gesture segmentation and we were able to get satisfactory results.

The most compelling part of the architecture for us to explore was the feature extractor. We tried to come up with an idea that can help boost the results of the model, make intuitive sense and in addition can be generalized to other architectures and tasks.

To try improve the results we had using the base architecture we took an off direction. Instead of making a change in the architecture itself, we borrowed inspiration from the recent interest and development in data-centric machine learning [Miranda, 2021]. A new regime in AI that was introduced by the renown researcher Andrew Ng.

We changed the actual frames' annotations, instead of one label for each gesture, we created three different labels. The labels are one for the beginning of the gesture - first 10 frames of the gesture, the end of the gesture - 10 last frames of the gesture and middle of the gesture - for the rest of the frames. The idea is that the features learnt by the model will incorporate information about transitions between gestures and will thus improve the final segmentation.

The architecture of the model is thus identical to the one was presented.

4 Trade-off

The trade-off we tested is trade-off between number of features (size on disk) to the model performance. The output size of the last layer of EffiecentNet-B0 is 1280. We used the PCA dimension reduction algorithm to test different sizes of feature vectors ranges from 1280 to 64. We expected to see that when we are taking less features for each frame the performance will deteriorate accordingly. We were surprised to see that the opposite was happening. The accuracy results for each dimension are presented in Table 1.

	Accuracy	F1@10	F1@25	F1@50	Edit Score
64	0.876±0.02	0.902±0.025	0.891±0.026	0.828±0.038	0.862±0.036
128	0.877±0.022	0.902±0.028	0.89±0.03	0.829±0.043	0.864±0.035
256	**0.879±0.019**	**0.914±0.019**	**0.903±0.02**	**0.846±0.032**	**0.882±0.026**
512	0.875±0.021	0.905±0.016	0.894±0.019	0.831±0.03	0.869±0.022
1280	0.871±0.019	0.902±0.026	0.888±0.027	0.821±0.038	0.864±0.034

Table 1: Trade-off results

In order to explain ourselves the surprising results we plotted the cumulative explained variance as function of number of features, see Figure 5. As we hypothesized the explained variance for 64

dimensions is already over 98% and this provides possible good explanation for the success of low number of features.

From that we can learn that the optimal amount of features is much lower than the default 1280 and one might train a feature extractor model with much smaller output dimension or preform PCA to improve the model performance while preserving storage space.

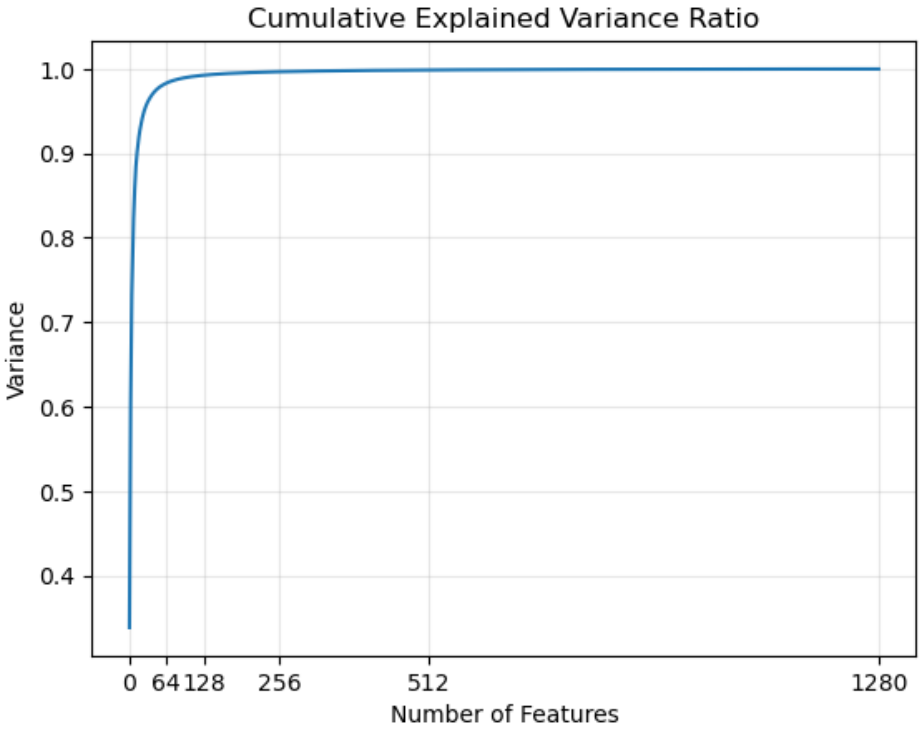

Figure 5: Cumulative explained variance as function of number of features

5 Results

In order to test our method we used the well know metrics of accuracy, F1@k that was suggested by [Lea et al., 2017] and became the main measurement of performance in action segmentation tasks and the good old edit distance metric.

All models were evaluated using k-fold cross-validation (K=5). The data was split in similar way to that described in [Goldbraikh et al., 2022b].

In addition to the results over each of the folds we also present the standard deviation for the reader to evaluate.

The results presented in Table 2

		Accuracy	F1@10	F1@25	F1@50	Edit Score
Fold 0	Baseline	0.853	0.871	0.853	0.779	0.811
	Smoothing	0.848	0.881	0.858	0.788	0.825
	Transition Classes	0.868	0.889	0.881	0.816	0.847
Fold 1	Baseline	0.858	0.874	0.855	0.782	0.826
	Smoothing	0.844	0.884	0.858	0.786	0.837
	Transition Classes	0.874	0.929	0.909	0.852	0.906
Fold 2	Baseline	0.87	0.883	0.866	0.798	0.841
	Smoothing	0.862	0.88	0.864	0.781	0.829
	Transition Classes	0.852	0.879	0.865	0.795	0.844
Fold 3	Baseline	0.856	0.85	0.829	0.749	0.806
	Smoothing	0.855	0.856	0.836	0.755	0.806
	Transition Classes	0.858	0.875	0.855	0.769	0.821
Fold 4	Baseline	0.916	0.916	0.906	0.868	0.888
	Smoothing	0.903	0.911	0.901	0.861	0.888
	Transition Classes	0.906	0.937	0.928	0.873	0.904
Average	Baseline	0.87±0.023	0.879±0.022	0.862±0.025	0.795±0.04	0.834±0.029
	Smoothing	0.862±0.021	0.883±0.017	0.863±0.021	0.794±0.036	0.837±0.027
	Transition Classes	**0.871±0.018**	**0.902±0.026**	**0.888±0.027**	**0.821±0.038**	**0.864±0.034**

Table 2: Results of *Baseline model*, *Smoothing* (additional experiment we preformed and will be described in the following chapter) and *Transition Classes* which is the method we are suggesting.

As we hypothesized, the method we suggested helped improving mainly the F1 and edit distance metrics. What we understand from the results we got is that the idea of forcing the model to learn the beginning and ending of a gesture helped it to create better features for the following transformer model and actually learn something about the transitions between gestures.

Generally we can learn that that thinking ahead of time about the data we have and the task in mind, and try to understand how to manipulate the data and annotations we can get "free" improvement in the model results. In our case we saw the results of the model and lingered about the places where the uncertainty of the model was the highest - which obviously was in the transitions between gestures. That pointed us towards the solution we actually implemented in the end.

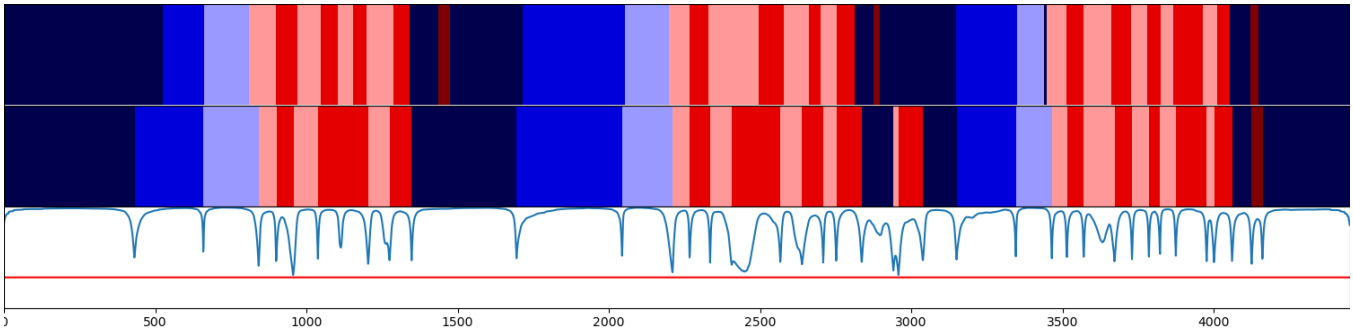

Figure 6: The blue line in the picture represent the uncertainty of the model about the prediction in each frame. We can see that the uncertainty is lowest in the transitions between gestures.

6 Additional Experiment

In addition to the experiment we presented, we also tried another idea. We added a 1D convolution layer on top of each of the outputs of the ASFormer, thinking of it as an extra smoothing step that can improve the results. Actually at the begging we wanted to run exponentially decaying average filter but we understood that back-propagation knows better so we went for the 1D conv layer.

The results on the first try was disappointing as can be seen in Table 3, so that's why we ditched this idea and moved on to improve the feature extractor. We do think that with different hyperparamters and using different parameters for each module instead of the same one for all the modules it might work.

References

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[Goldbraikh et al., 2022a] Goldbraikh, A., D'Angelo, A.-L., Pugh, C. M., and Laufer, S. (2022a). Video-based fully automatic assessment of open surgery suturing skills. *International Journal of Computer Assisted Radiology and Surgery*, 17(3):437–448.

[Goldbraikh et al., 2022b] Goldbraikh, A., Volk, T., Pugh, C. M., and Laufer, S. (2022b). Using open surgery simulation kinematic data for tool and gesture recognition. *International Journal of Computer Assisted Radiology and Surgery*, 17(6):965–979.

[Lea et al., 2017] Lea, C., Flynn, M. D., Vidal, R., Reiter, A., and Hager, G. D. (2017). Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165.

[Miranda, 2021] Miranda, L. J. (2021). Towards data-centric machine learning: a short review. *ljvmiranda921.github.io*.

[Tan and Le, 2019] Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR.

[Yi et al., 2021] Yi, F., Wen, H., and Jiang, T. (2021). Asformer: Transformer for action segmentation. *arXiv preprint arXiv:2110.08568*.