Not End-to-End: Explore Multi-Stage Architecture for Online Surgical Phase Recognition

Fangqiu Yi*, Yanfeng Yang*, and Tingting Jiang(✉)

National Engineering Research Center of Visual Technology, School of Computer Science, Peking University, Beijing, China
{chinayi,yangyanfeng,ttjiang}@pku.edu.cn

Abstract. Surgical phase recognition is of particular interest to computer assisted surgery systems, in which the goal is to predict what phase is occurring at each frame for a surgery video. Networks with multi-stage architecture have been widely applied in many computer vision tasks with rich patterns, where a predictor stage first outputs initial predictions and an additional refinement stage operates on the initial predictions to perform further refinement. Existing works show that surgical video contents are well ordered and contain rich temporal patterns, making the multi-stage architecture well suited for the surgical phase recognition task. However, we observe that when simply applying the multi-stage architecture to the surgical phase recognition task, the end-to-end training manner will make the refinement ability fall short of its wishes. To address the problem, we propose a new non end-to-end training strategy and explore different designs of multi-stage architecture for surgical phase recognition task. For the non end-to-end training strategy, the refinement stage is trained separately with proposed two types of disturbed sequences. Meanwhile, we evaluate three different choices of refinement models to show that our analysis and solution are robust to the choices of specific multi-stage models. We conduct experiments on two public benchmarks, the M2CAI16 Workflow Challenge and the Cholec80 dataset. The SOTA comparable results show that the multi-stage architecture holds the great potential to boost the performance of existing single-stage models. Code is available at https://github.com/ChinaYi/NETE.

Keywords: Surgical Phase Recognition · Surgical Workflow Segmentation · Multi-Stage Architecture.

1 Introduction

Surgical phase recognition is of particular interest to computer assisted surgery systems, because it offers solutions to numerous demands of the modern oper-

* contribute equally to this work.

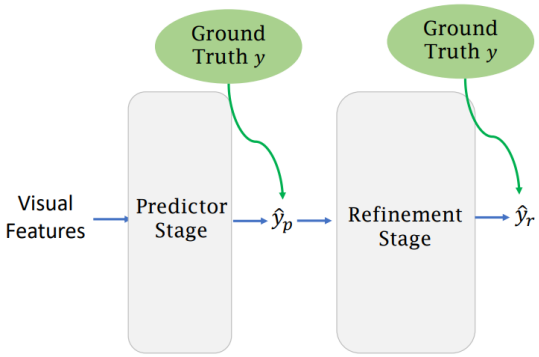

Fig. 1. Pipeline of multi-stage architecture.

ating room, such as monitoring surgical processes [1], scheduling surgeons [2] and enhancing coordination among surgical teams [3]. This paper works on the online surgical phase recognition task, which requires to predict what phase is occurring at each frame without using the information of future frames.

Existing surgical phase recognition models can be divided into two groups. The first group is single-stage models which output prediction results with the input visual features, while the second group is the multi-stage models which additionally stack a refinement stage over the prediction results to perform a further refinement. In our opinion, the multi-stage architecture is the one which is well-suited for the surgical phase recognition task. First of all, networks with multi-stage architecture have been widely applied in many computer vision tasks with rich patterns, such as human pose estimation [4, 5] and action segmentation [6]. Generally speaking, the idea of multi-stage architecture consists of a predictor stage and a refinement stage, as shown in Fig. 1. Sometimes, due to the hard-to-recognize visual features, the initial predictions output by the predictor stage may have errors that violate intrinsic patterns within the data. (*i.e.* The tiny spikes of over-segmentation errors for a continuous action or human pose estimation results that do not conform to the connections of human body joint.) The initial predictions are thus further refined by the refinement stage. Secondly, surgical video contents are well ordered and contain rich temporal patterns. Some works have been motivated by utilizing the rich temporal patterns to refine the predictions. The success in [7, 8] shows that it is possible for the multi-stage architecture to rectify the misclassifications due to the ambiguous visual features in the predictor stage.

However, we observe that the improvement of multi-stage structure in the surgical phase recognition task is not as obvious as other tasks. Experiments in [9] show that the performance improvement of the additional refinement stage is very limited. This interesting phenomenon raises our concerns, *why* the multi-stage architecture does not work as well as we expected?

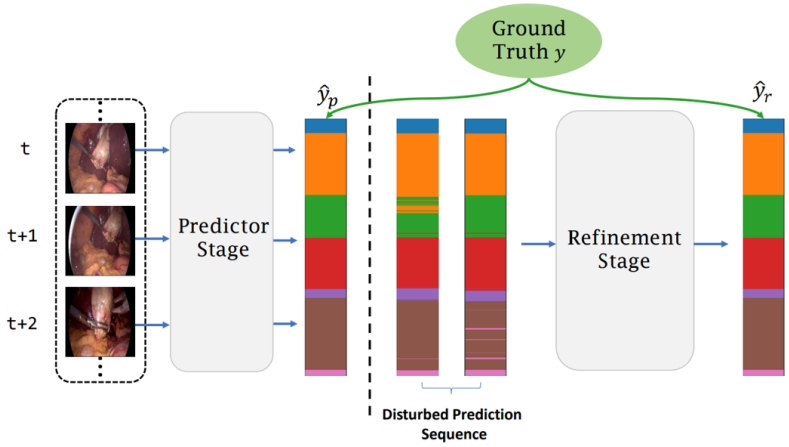

Fig. 2. Our non end-to-end training strategy where two stages are trained separately.

In this paper, we first answer the question of *why* with our analysis, and then further give a solution of *how* to make multi-stage architecture work better for surgical phase recognition task. The reason of *why* is two folds. Firstly, a common issue for the multi-stage architecture is that the refinement stage cannot actually learn how to refine with an end-to-end training manner. As shown in Fig. 1, the supervision signal is applied in both the predictor stage and refinement stage. After several epochs of training, the predictor stage will quickly converge to the ground truth, which means that the initial predictions $\hat{y}_p$ are almost 100% correct during the training process. However, in the inference process with the test data, $\hat{y}_p$ still remain lots of errors due to imperfect predictor. The huge gap between the inputs of the refinement stage during training and inference makes the refinement ability fall short of its wishes. Secondly, in the case of end-to-end training, the limited size of current datasets for surgery phase recognition cannot afford the training of refinement stage which brings additional parameters. This leads to a severe overfitting problem compared to the results of applying multi-stage architecture to other vision tasks.

With the answer of *why*, we propose a non end-to-end training strategy where the predictor stage and the refinement stage are trained separately to solve the above two issues simultaneously. As shown in Fig. 2, the predictor stage is trained with the raw video data. To reduce the gap between the inputs of the refinement stage during training and inference, two types of training sequences for refinement stage are carefully designed to simulate the real output of the predictor during inference, denoted as cross-validate type and mask-hard-frame type respectively. Meanwhile, as the refinement stage is separately trained with the two types of carefully designed training sequences, the over-fitting problem for surgical phase recognition can also be alleviated.

Besides the training strategy, we further explore the designs of multi-stage architecture by evaluating three different temporal models for the refinement

stage, including TCN (offline) [10], causal TCN (online) [10] and GRU (online) [11]. As for the predictor stage, we use the causal TCN in [9] for its high efficiency and good performance. In principle, our solution can be applied to any single-stage predictor model. Extensive experiments are conducted on two public benchmarks, M2CAI16 [12] and Cholec80 [13]. Results show that all three refinement models trained with our strategy successfully boost the performance of the single predictor stage, demonstrating that our analysis and solution are robust to different choices of refinement models. And the SOTA comparable results show that the multi-stage architecture holds the great potential to boost the performance of existing single-stage models.

2 Related Work

Existing surgical phase recognition models can be divided into two groups. The first group is single-stage models which output prediction results with the input visual features. For example, a number of works utilized dynamic time warping [14, 15], conditional random field [16], and variations of Hidden Markov Model (HMM) [17, 18] over extracted visual features. [7] trained an end-to-end RNN model that first used a very deep ResNet to extract visual features for each frame and then applied a LSTM network to model the temporal dependencies of sequential frames. [19] proposed a LSTM-based temporal network structure that leveraged task-specific network representation to collect long-term sufficient statistics that were propagated by a sufficient statistics model. In addition, with the wide application of transformer in computer vision, there are also some single-stage models based on transformer. [20] used a novel attention regularization loss which encouraged the transformer model to focus on high-quality frames during training, and the attention weights could be utilized to identify characteristic high attention frames for each surgical phase, which could further help the surgery summarization. [21] proposed a hybrid embedding aggregation transformer model which used cleverly designed spatial and temporal embeddings by allowing for active queries based on spatial information from temporal embedding sequences.

The second group is the multi-stage models which additionally stack a refinement stage over the prediction results to perform a further refinement. [9] is the first one which brought in multi-stage architecture for surgical phase recognition task. They used a causal TCN [10] to output initial predictions over pre-extracted CNN features, and then appended another causal TCN [10] to refine the predictions. [22] proposed a multi-task multi-stage temporal convolutional network along with a multi-task convolutional neural network training setup to jointly predict the phases and steps and benefit from their complementarity to better evaluate the execution of the procedure.

In addition, some surgical phase recognition models used data augmentation techniques to improve performance, such as cross validation and hard frames detection. Cross validation is the simplest way to make part of training data unseen to the predictor model, so that the unseen part of data could be used to

simulate the real predictions during the inference. Besides, the concept of hard frames was first proposed by [8], which denoted the frames that were not recognizable from their visual appearance. They observed that single-stage models usually made mistakes on hard frames, so they found out all the hard frames in the training videos and mapped them to corresponding phases separately.

3 Methods

The multi-stage architecture stacks a refinement stage over the predictor stage sequentially. We propose a training strategy where these two stages are trained separately and explore designs of multi-stage architecture. We first introduce the predictor stage and its training in Sec. 3.1, then describe the generation process of disturbed prediction sequences in Sec. 3.2, and finally discuss the refinement stage and its training in Sec. 3.3. It is worth noting that, although we train the multi-stage architecture in a non end-to-end manner, the inference process is still end-to-end as the normal multi-stage architecture.

3.1 Predictor Stage

We use causal TCN in [9] to get the initial predictions for its high efficiency and good performance. Instead of the general temporal convolutions which depend on both n past and n future frames, the causal temporal convolutions only rely on the current and previous frames and thus meet the demand of online surgical phase recognition. In principle, the predictor model can be any online model. The input of the causal TCN is the frame-wise extracted features from a pre-trained CNN. Denote the output prediction sequence as $\hat{y}_p \in \mathcal{R}^{C \times T}$. For each frame, the output is a vector of size C denoting the classification probability for each class. For the loss function $\mathcal{L}_p$, we use a combination of cross-entropy classification loss and a smoothing loss [23] by deploying a mean square error over the classification probabilities of every two adjacent frames. The loss function writes as

$$\begin{aligned} \mathcal{L}_p = & \frac{1}{T} \sum_{t=1}^{T} -log(\hat{y}_{p(c,t)}) \\ & + \frac{1}{TC} \sum_{m=1}^{C} \sum_{t=1}^{T-1} \left| \hat{y}_{p(m,t)} - \hat{y}_{p(m,t+1)} \right|^2 . \end{aligned} \tag{1}$$

3.2 Disturbed Prediction Sequence Generation

The input of the refinement stage is the prediction results $\hat{y}_p$ output by the predictor stage. During the inference, $\hat{y}_p$ will remain lots of errors due to the imperfect predictor. In order to achieve better refinement results, we should minimize the distribution gap between the preliminary prediction results in training and inference. Thus, we design two types of disturbed prediction sequences by simulating the imperfect prediction results of the predictor model during the

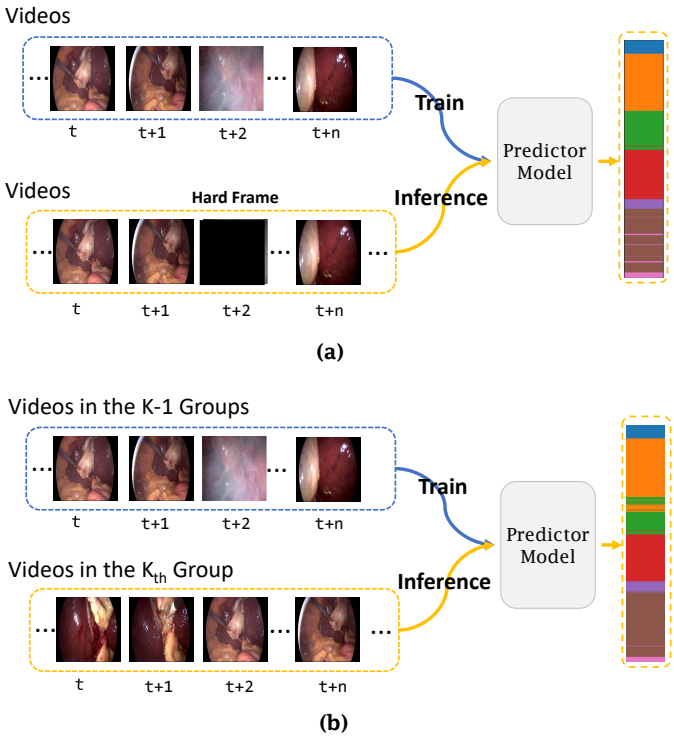

Fig. 3. (a) The generation of mask-hard-frame type. (b) The generation of cross-validate type.

inference. Note that the generation of both two types of disturbed prediction sequences are related to the predictor model, since we cannot directly obtain the prediction sequences from the raw video data.

Mask-Hard-Frame Type. The concept of hard frames was proposed by [8], which denoted the frames that were not recognizable from their visual appearance. Their work shows that single-stage models usually make mistakes on hard frames. Motivated by this, we seek to add perturbations to the prediction of these hard frames. We first train a predictor model with the normal video data. Then, we follow the rule of [8] to find out hard frames in the training set and add perturbations on those hard frames by using a black mask to cover the whole image. Finally, we pass the perturbed video data to the predictor model to get the disturbed prediction sequence. The workflow is shown in Fig. 3(a).

Cross-Validate Type. Cross validation is the simplest way to make part of training data unseen to the predictor model, so that the unseen part of data could be used to simulate the real predictions of the predictor model during

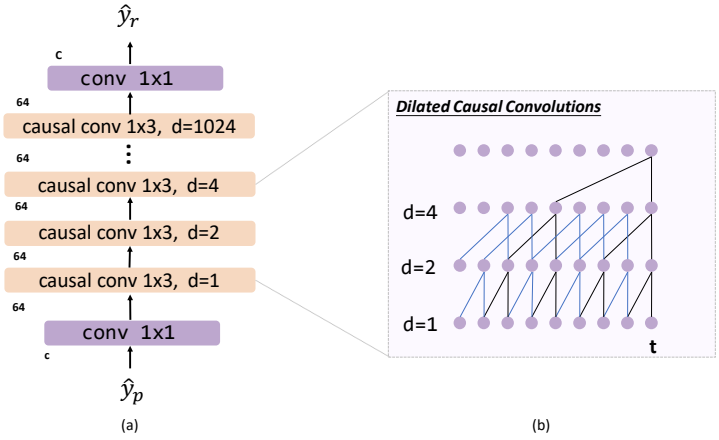

Fig. 4. (a) An overview of causal TCN, which can be performed online. The dilation factor is doubled at each causal layer, *i.e.* 1, 2, 4, ..., 1024. (b) The architecture of the causal convolutions with different dilation factors.

the inference. Specifically, we randomly partition the training videos into K groups of equal size. Each time, a single group is retained for validation, and the remaining $K-1$ groups are used to train a predictor model. And then, we use the trained predictor model to obtain the predictions of the retained video to get the prediction sequence. The workflow is shown in Fig. 3(b). We set $K = 10$ in our experiments.

3.3 Refinement Stage

For a training set with N training videos, both two methods can generate N perturbed prediction sequences to train the refinement model. Both two types of disturbed sequences are used to train the refinement stage in following experiments except for special specification. For the loss function of the refinement stage, we use the cross-entropy loss. For the choice of specific refinement model, we evaluate three common temporal models, including TCN [10], causal TCN [10] and GRU [11]. We chose these three models as the refinement models because these are the three most common temporal models. In addition, stacking several predictors sequentially has shown significant improvements in many tasks like human pose estimation and action segmentation. The stacked architecture is composed of several models sequentially such that each model operates directly on the output of the previous one. So we also stack single GRU stages to form a stacked GRU for the refinement stage.

Causal TCN. The overview of causal TCN is illustrated in Fig. 4(a). The first layer of causal TCN is a 1×1 convolutional layer, that adjusts the dimension

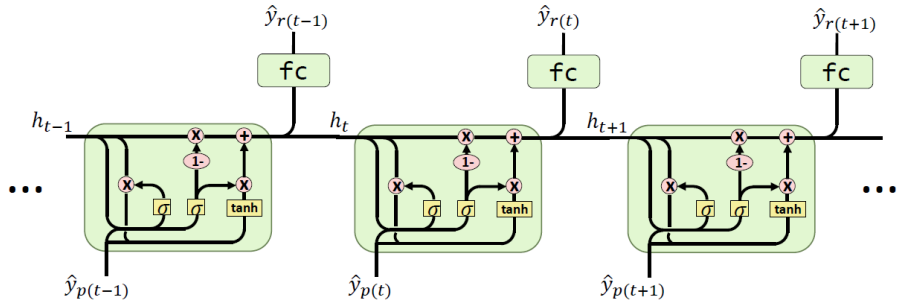

Fig. 5. The overview of single GRU network. The GRU unit employs three gates to modulate the interactions between the GRU cells and the environment. The dimension of the hidden state is set to 128.

of the input features from 2048-d to 64-d. Then, this layer is followed by several layers of dilated causal convolutions, as shown in Fig. 4(b). The causal convolution is different from common temporal convolution, for each time step t and filter length d, it convolves from X_{t-d} to X_t, which meets the demands of online surgical phase recognition. Note that we do not use temporal pooling layer, because it might results in a loss of fine-grained information that is necessary for phase recognition. Instead, we use a dilation factor that is doubled at each causal layer, *i.e.* 1, 2, 4, ..., 1024, to enlarge the temporal receptive field. Finally, we apply a 1×1 convolution over the output of the last dilated causal convolution layer to get the probability vector dimension.

Single GRU. Gated Recurrent Unit (GRU) is a popular variant of Recurrent Neural Network (RNN). The architecture of GRU is shown in Fig. 5. The GRU unit employs three gates, *i.e.*, a reset gate r_t, an update gate z_t and a new gate n_t, to modulate the interactions between the GRU cells and the environment. The dimension of the hidden state is set to 128. At timestep t, given input p_t (probability vector of frame X_t belonging to each phase), hidden state h_{t-1}, the GRU unit updates with following equations:

$$\begin{aligned} r_t &= \sigma(W_{ir}p_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \ , \\ z_t &= \sigma(W_{iz}p_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \ , \\ n_t &= tanh(W_{in}p_t + b_{in} + r_t * (W_{hn}h_{(t-1)} + b_{hn})) \ , \\ h_t &= (1 - z_t) * n_t + z_t * h_{(t-1)} \ . \end{aligned} \tag{2}$$

In above equations, σ is the sigmoid function, and $*$ is the Hadamard product. To get the refined predictions, we additionally apply a fully connected layer over the hidden state.

Stacked GRU. The refinement model takes an initial prediction as input, and then outputs a refined prediction. It is natural to come up with the idea that if

we stack multiple refinement models sequentially, each model will successively operate on the refined predictions of the previous one. The effect of such composition is an incremental refinement of the predictions from the previous model. Therefore, in the architecture of stacked GRU, the input to the latter GRU is the refined result from the previous GRU, not the hidden state. The set of operations at each GRU in stacked GRU can be formally described as follows:

$$\begin{aligned} \mathcal{P}^0 &= \hat{p}_{1:T} \ , \\ \mathcal{P}^s &= GRU(\mathcal{P}^{s-1}) \ . \end{aligned} \tag{3}$$

In above equations, $\mathcal{P}^s$ is the refined predictions at sth GRU and GRU is the single-stage GRU discussed before. As for loss function, different from single-stage refinement models, we use the cross entropy loss on the refined probability sequence of each GRU.

3.4 Training Details

We employ the ResNet50 [24] as the visual feature extractor to extract off-the-shelf video features. Specifically, the ResNet50 is trained frame-wise without temporal information through cross-entropy loss. The dimension of the input ResNet50 features is 2048-d. With the pre-extracted video features, we train the predictor stage for 100 epochs with initial learning rate 1e-4 and Adam optimizer. For the refinement stage, we train the model for 40 epochs with two proposed disturbed sequences.

4 Experiment

4.1 Dataset

M2CAI16 Workflow Challenge Dataset. M2CAI16 dataset [12] contains 41 laparoscopic videos that are acquired at 25 fps of cholecystectomy procedures, and 27 of them are used for training and 14 videos are used for testing. These videos are segmented into 8 phases by experienced surgeons.

Cholec80 Dataset. Cholec80 dataset [13] contains 80 videos of cholecystectomy surgeries performed by 13 surgeons. The dataset is divided into training set (40 videos) and testing set (40 videos). These videos are segmented into 7 phases and are captured at 25 fps.

4.2 Metrics

To quantitatively analyze the performance of our method, we use three metrics [7] including the jaccard index ($JACC$), recall (Rec) and accuracy (Acc). Among them, Acc quantitatively evaluates the amount of correctly classified phases in the whole video, while Rec and $JACC$ evaluate the results for each individual phase.

Table 1. Comparison of multi-stage architectures with three different choices of refinement models under the end-to-end training strategy and ours on Cholec80 dataset. Predictor denotes single-stage predictor without the refinement stage.

Method	Acc	JACC	Rec
Predictor	88.8±6.3	73.2±9.8	84.9±7.2
End-to-End+GRU	87.1±7.8	69.7±12.6	83.2±9.4
End-to-End+causal TCN	87.7±6.3	77.7±11.2	84.3±6.3
End-to-End+TCN	89.8±6.6	75.8±8.4	87.4±7.5
Ours+GRU	90.8±7.0	75.5±11.1	85.6±10.0
Ours+causal TCN	91.0±5.2	74.2±11.8	84.1±9.6
Ours+TCN	**92.8±5.0**	**78.7±9.4**	**87.5±8.3**

4.3 End-to-End VS. Not End-to-End

In this section, we evaluate the performance of multi-stage architectures with the end-to-end training strategy and our non end-to-end training strategy on the Cholec80 dataset. Results are shown in Table 1. We can observe that, with the end-to-end training strategy, the multi-stage architecture only achieves comparable results with the single-stage predictor. When we use causal TCN or GRU as the refinement model, performance is even slightly worse possibly due to the over-fitting problem. Such results are consistent with the results in [9]. Meanwhile, all three refinement models trained with our proposed disturbed sequences largely boost the performance of the predictor, which proves our previous analysis above the multi-stage architecture and shows that our solution is effective and not sensitive to the choices of different refinement models. Although TCN achieves the best performance as the refinement model, however, it does not meet the constraint of online surgical phase recognition because it needs information from future frames. So, we explore GRU as the refinement model for further experiments.

Fig. 6 shows the qualitative results of GRU as the refinement model under the end-to-end training strategy and our non end-to-end training strategy on the Cholec80 dataset. We can observe that the results of single-stage predictor and end-to-end multi-stage model both contain a large number of short surgical phases, which do not meet the temporal continuity and destroy the smoothness of the predicted results. Besides, the result of end-to-end multi-stage model also contains more prediction errors than single-stage predictor. These results clearly highlight the ability of the multi-stage architecture trained with our solution which obtains consistent and smooth predictions.

4.4 Stacked Number of GRUs in Refinement Model

In this section, we use the stacked GRU as the refinement stage model. During training, the stacked GRU is trained as a whole with the two proposed sequences and the loss function is applied on the output of each GRU. Results of the stacked

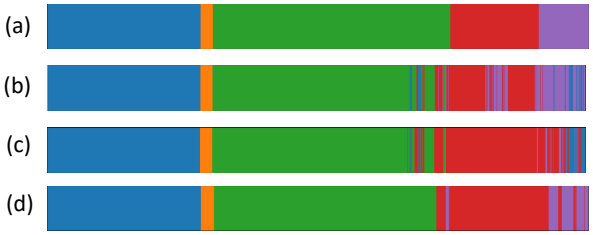

Fig. 6. Qualitative results on the Cholec80 dataset of the multi-stage architectures trained with end-to-end manner and ours. GRU is used for the refinement stage. (a) is the ground truth. (b) is the prediction from the single-stage predictor. (c) is the prediction from the multi-stage architecture trained with end-to-end manner. (d) is the prediction from the multi-stage architecture trained with our solution.

Table 2. Effects of the stacked number of GRUs on the Cholec80 dataset.

Method	Acc	JACC	Rec
Single GRU	90.8±7.0	75.5±11.1	85.6±10.0
Stacked GRU$_{(2\ \text{GRU})}$	91.3±6.1	75.9±11.2	84.4±10.0
Stacked GRU$_{(3\ \text{GRU})}$	**91.5±7.1**	**77.2±11.2**	**86.8±8.5**
Stacked GRU$_{(4\ \text{GRU})}$	90.8±5.8	74.2±15.9	83.1±14.6

GRU with different numbers of GRUs on the Cholec80 dataset are shown in Table 2. We get the best results when stacking 3 GRUs sequentially on the Cholec80 dataset. In order to give an intuitive explanation about how the stacked refinement model works, we also show the qualitative results of a video in different stages for a stacked GRU with 3 GRUs in Fig. 7. Fig. 7(a) shows the predictions at each stage, and Fig. 7(b) shows the probability sequences of the predictions. We can observe that adding more GRUs results in an incremental refinement of the predictions. We also conduct experiments on the M2CAI16 dataset. Results of the stacked GRU with different numbers of GRUs on the M2CAI16 dataset are shown in Table 3. we can observe that the experiments get the best results when stacking 2 GRUs, which is less than that of the Cholec80 dataset. This may due to the fact that the size of M2CAI16 is smaller.

Table 3. Effects of the stacked number of GRUs on the M2CAI16 dataset.

Method	Acc	JACC	Rec
Single GRU	86.2±9.1	72.6±11.6	90.0±11.7
Stacked GRU$_{(2\ \text{GRU})}$	**88.2±8.5**	**75.1±10.6**	**91.4±11.2**
Stacked GRU$_{(3\ \text{GRU})}$	86.9±10.2	72.7±11.1	89.9±9.2
Stacked GRU$_{(4\ \text{GRU})}$	87.0±8.4	72.4±11.0	89.0±12.3

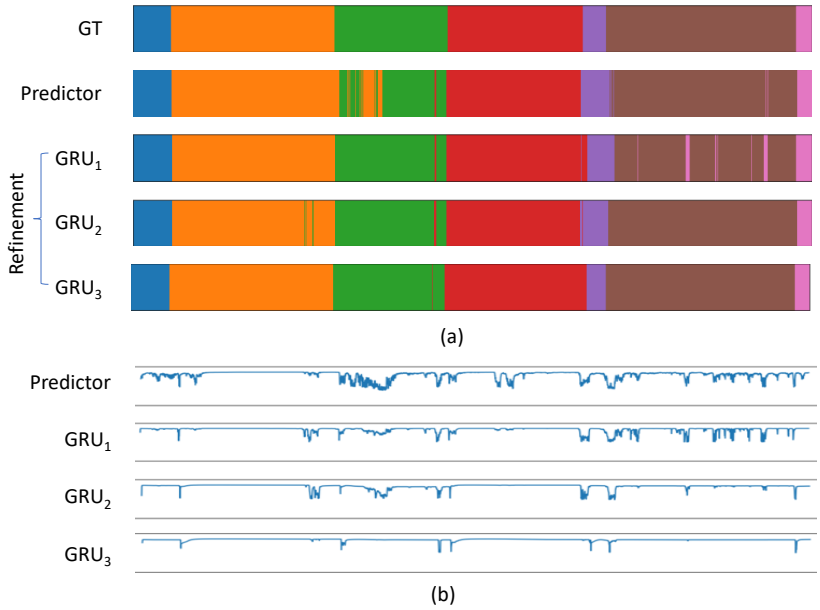

Fig. 7. (a) Qualitative results for the predictions in each GRU of the stacked GRU on the Cholec80 dataset. (b) Qualitative results for the probability sequences of the predictions in each GRU of the stacked GRU.

4.5 Impact of Disturbed Prediction Sequence

The disturbed prediction sequence for training the refinement stage is very important. In this section, we conduct ablative experiments by using different combinations of the disturbed prediction sequences and other augmentation techniques. For the multi-stage architecture, we use the stacked GRU with 3 GRUs as the refinement stage. Besides the mask-hard-frame type and the cross-validate type, we additionally designed the random-mask type and the normal-noise type. Different from mask-hard-frame type, the random-mask type randomly masks frames, no matter how important the masked frames are. The normal-noise type adds Gaussian noise with different standard deviations to the output of the predictor model to simulate the imperfect prediction results of the predictor model during the inference. Table 4 shows the results on the Cholec80 dataset. First, if we only use one type of disturbed sequence, the performances will drop due to the lack of training data. Meanwhile, we can observe that, the refinement stage trained with the random-mask type is not as good as the mask-hard-frame type. This demonstrates the importance of the location to add perturbations. Only when those frames that are not easily classified correctly during inference are masked, the sequence obtained is most similar to the real situation, and that's why the result of *mhf* is better than *rm*. In addition, when using one augmentation method, the way of adding noise performs better than the mask-hard-frame

Table 4. Results of the multi-stage architectures trained with different augmentation techniques. Abbreviations: *cv* for the cross-validate type, *mhf* for the mask-hard-frame type, *rm* for the random-mask type, *nn* for the normal-noise type.

Method	Acc	JACC	Rec
nn($\sigma = 0.3$)	88.9±6.3	72.8±9.4	83.6±9.6
nn($\sigma = 0.5$)	89.3±6.4	74.2±9.1	84.8±8.2
nn($\sigma = 0.7$)	89.1±6.4	73.2±9.7	84.2±9.2
cv	89.6±5.6	70.4±13.5	83.5±13.2
mhf	88.7±9.4	70.6±9.4	81.8±9.9
rm	86.5±7.8	69.7±10.9	82.4±7.0
nn+cv	90.4±5.3	72.9±14.6	85.6±10.6
nn+mhf	89.2±7.2	71.9±10.5	82.7±9.7
nn+rm	87.4±6.8	69.2±11.5	82.6±8.6
cv+mhf	91.5±7.1	**77.2±11.2**	86.8±8.5
nn+cv+mhf	**92.0±5.3**	77.1±11.5	**87.0±7.3**
cv+rm+mhf	91.0±6.8	75.3±13.4	85.4±10.3
nn+cv+rm+mhf	91.7±5.2	76.0±12.6	86.4±9.5

type and the random-mask type. And when combined with the other methods, the normal-noise type improves the performance of other methods and achieves the SOTA performance. These results show that adding Gaussian noise to the output of the predictor model is also an effective perturbation, and it can play a complementary role to the disturbed prediction sequences.

4.6 Comparison with the SOTA Methods

In order to compare our solution with the SOTA methods, we use a stacked GRU with 3 GRUs and a stacked GRU with 2 GRUs as the refinement stage for the Cholec80 dataset and the M2CAI16 dataset, respectively. Table 5 and Table 6 show the results. Compared with the single-stage predictor causal TCN, multi-stage architecture trained with our solution largely boosts its performance. Noting that TeCNO [9] is also a multi-stage network, where causal TCN is used for both predictor stage and refinement stage. We can observe that, when simply applying the multi-stage network on surgical phase recognition task, the end-to-end training makes the refinement ability fall out of its wishes, TeCNO only achieves comparable results with the single-stage predictor causal TCN. However, when applying our strategy, the advantage of multi-stage network is revealed, which proves that our previous analysis above the multi-stage architecture and shows that our solution is effective. Besides, compared with these SOTA methods, our method is comparable to Trans-SVNet [21] on the Cholec80 dataset and much better than the other methods, while on the M2CAI16 dataset, our method outperforms all other methods.

Table 5. Comparison with the SOTA methods on the Cholec80 dataset.

Method	Acc	JACC	Rec
ResNet [8]	78.3±7.7	52.2±15.0	-
PhaseLSTM [25]	80.7±12.9	64.4±10.0	-
PhaseHMM [25]	71.1±20.3	62.4±10.4	-
EndoNet [13]	81.7±4.2	-	79.6±7.9
EndoNet-GTbin [13]	81.9±4.4	-	80.0±6.7
SV-RCNet [7]	85.3±7.3	-	83.5±7.5
OHFM [8]	87.0±6.3	66.7±12.8	-
TeCNO [9]	88.6±2.7	-	85.2±10.6
OperA [20]	85.8±1.0	-	87.7±0.7
Trans-SVNet [21]	90.3±7.1	**79.3±6.6**	**88.8±7.4**
causal TCN	88.8±6.3	73.2±9.8	84.9±7.2
Ours	**92.0±5.3**	77.1±11.5	87.0±7.3

Table 6. Comparison with the SOTA methods on the M2CAI16 dataset.

Method	Acc	JACC	Rec
ResNet [8]	76.3±8.9	56.4±10.4	-
PhaseLSTM [25]	72.5±10.6	54.8±8.9	-
PhaseHMM [25]	79.5±12.1	64.1±10.3	-
SV-RCNet [7]	81.7±8.1	65.4±8.9	81.6±7.2
OHFM [8]	84.8±8.0	68.5±11.1	-
Trans-SVNet [21]	87.2±9.3	74.7±7.7	87.5±5.5
causal TCN	84.1±9.6	69.8±10.7	88.3±9.6
Ours	**88.2±8.5**	**75.1±10.6**	**91.4±11.2**

5 Conclusion and Future Work

In this paper, we observe that the end-to-end training manner makes the refinement ability of the multi-stage architecture fall out of its wishes. In order to solve the problem, we propose a new non end-to-end training strategy and explore different designs of multi-stage architectures for surgical phase recognition task. To minimize the distribution gap between the training and inference, we generate two types of disturbed sequences as the input of the refinement stage. In the future, we will explore other different predictor models, and apply our solution to other computer vision tasks where the multi-stage architecture is widely applied.

Acknowledgements. This work was partially supported by the Natural Science Foundation of China under contract 62088102. We also acknowledge the Clinical Medicine Plus X-Young Scholars Project, and High-Performance Computing Platform of Peking University for providing computational resources.

References

1. Bricon-Souf, N., Newman, C.R.: Context awareness in health care: A review. International Journal of Medical Informatics **76** (2007) 2 – 12
2. Bhatia, B., Oates, T., Xiao, Y., Hu, P.: Real-time identification of operating room state from video. In: Proceedings of the Conference on Innovative Applications of Artificial Intelligence. Volume 2. (2007) 1761–1766
3. Lin, H.C., Shafran, I., Murphy, T.E., Okamura, A.M., Yuh, D.D., Hager, G.D.: Automatic detection and segmentation of robot-assisted surgical motions. In: Medical Image Computing and Computer Assisted Intervention. (2005) 802–810
4. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European Conference on Computer Vision. (2016) 483–499
5. Wei, S.E., Ramakrishna, V., Kanade, T., Sheikh, Y.: Convolutional pose machines. In: IEEE Conference on Computer Vision and Pattern Recognition. (2016) 4724–4732
6. Farha, Y.A., Gall, J.: MS-TCN: Multi-stage temporal convolutional network for action segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition. (2019) 3570–3579
7. Jin, Y., Dou, Q., Chen, H., Yu, L., Qin, J., Fu, C.W., Heng, P.A.: SV-RCNet: Workflow recognition from surgical videos using recurrent convolutional network. IEEE Transactions on Medical Imaging **37** (2018) 1114–1126
8. Yi, F., Jiang, T.: Hard frame detection and online mapping for surgical phase recognition. In: Medical Image Computing and Computer Assisted Intervention. (2019)
9. Czempiel, T., Paschali, M., Keicher, M., Simson, W., Feussner, H., Kim, S.T., Navab, N.: Tecno: Surgical phase recognition with multi-stage temporal convolutional networks. In: Medical Image Computing and Computer Assisted Intervention. Volume 12263. (2020) 343–352
10. Lea, C., Flynn, M.D., Vidal, R., Reiter, A., Hager, G.D.: Temporal convolutional networks for action segmentation and detection. In: IEEE Conference on Computer Vision and Pattern Recognition. (2017) 1003–1012
11. Cho, K., van Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. (2014) 1724–1734
12. Stauder, R., Ostler, D., Kranzfelder, M., Koller, S., Feußner, H., Navab, N.: The TUM lapchole dataset for the M2CAI 2016 workflow challenge. arxiv **abs/1610.09278** (2016)
13. Twinanda, A.P., Shehata, S., Mutter, D., Marescaux, J., de Mathelin, M., Padoy, N.: EndoNet: A deep architecture for recognition tasks on laparoscopic videos. IEEE Transactions on Medical Imaging **36** (2017) 86–97
14. Blum, T., Feußner, H., Navab, N.: Modeling and segmentation of surgical workflow from laparoscopic video. In: Medical Image Computing and Computer Assisted Intervention. (2010) 400–407
15. Padoy, N., Blum, T., Ahmadi, S.A., Feussner, H., Berger, M.O., Navab, N.: Statistical modeling and recognition of surgical workflow. Medical Image Analysis **16** (2012) 632–641
16. Tao, L., Zappella, L., Hager, G.D., Vidal, R.: Surgical gesture segmentation and recognition. In: Medical Image Computing and Computer Assisted Intervention. (2013) 339–346

17. Lalys, F., Riffaud, L., Morandi, X., Jannin, P.: Surgical phases detection from microscope videos by combining SVM and HMM. Lecture Notes in Computer Science **6533** (2011) 54–62
18. Padoy, N., Blum, T., Feussner, H., marie odile, B., Navab, N.: On-line recognition of surgical activity for monitoring in the operating room. In: Proceedings of the Conference on Innovative Applications of Artificial Intelligence. Volume 3. (2008) 1718–1724
19. Ban, Y., Rosman, G., Ward, T., Hashimoto, D., Kondo, T., Iwaki, H., Meireles, O., Rus, D.: Aggregating long-term context for learning laparoscopic and robot-assisted surgical workflows. In: IEEE International Conference on Robotics and Automation. (2021) 14531–14538
20. Czempiel, T., Paschali, M., Ostler, D., Kim, S.T., Busam, B., Navab, N.: Opera: Attention-regularized transformers for surgical phase recognition. In: Medical Image Computing and Computer Assisted Intervention. Volume 12904. (2021) 604–614
21. Gao, X., Jin, Y., Long, Y., Dou, Q., Heng, P.A.: Trans-svnet: Accurate phase recognition from surgical videos via hybrid embedding aggregation transformer. In: Medical Image Computing and Computer Assisted Intervention. Volume 12904. (2021) 593–603
22. Ramesh, S., Dall'Alba, D., Gonzalez, C., Yu, T., Mascagni, P., Mutter, D., Marescaux, J., Fiorini, P., Padoy, N.: Multi-task temporal convolutional networks for joint recognition of surgical phases and steps in gastric bypass procedures. International Journal of Computer Assisted Radiology and Surgery **16** (2021) 1111–1119
23. Farha, Y.A., Gall, J.: MS-TCN: multi-stage temporal convolutional network for action segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition. (2019) 3575–3584
24. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016) 770–778
25. Twinanda, A.P., Mutter, D., Marescaux, J., Mathelin, M.D., Padoy, N.: Single- and multi-task architectures for surgical workflow challenge at M2CAI 2016. arxiv **abs/1610.08844** (2016)