Technical Report of Challenge on VideoPipe Track on Temporal Defect Localization

Jiawei Dong[1], Shuo Wang[1]
[1] Shanghai Paidao Intelligent Technology Co., Ltd.,
Shanghai 200240, China
Email: {Jiawei.dong, gavin.wang}@ai-prime.ai

***Abstract*—Video anomaly analysis is important for industrial applications in the real world. In particular, the urban pipe system is one of the most important infrastructures in a city. In order to ensure its normal operation, we need to inspect pipe defects smartly. This is a technical report on the temporal defect localization track of the ICPR VideoPipe Challenge. The report focuses on the team's data preparation, task definition, model selection, training process and inference process during the competition. The solution provides feasible adjustment strategies, and we will also provide complete training and inference code for others to reproduce.**

I. Introduction

Closed-Circuit TeleVision (CCTV) is a popular method for pipe defect inspection. Different from short QV videos, CCTV videos are much longer and record more comprehensive content in the very distant pipe. The main task is to discover temporal locations of pipe defects in such untrimmed videos.

Compared with other temporal localization datasets, first, the average length of videos in our dataset is 545 seconds, which is 3-4 times longer than other datasets. Second, unlike regular segment annotations, our dataset is based on moment annotations. Third, multiple defects may appear at the same moment, while in other datasets, multiple actions rarely appear in the same segment.

Compared with other pipeline defect detection datasets, our dataset is mainly based on video, other datasets are basically based on pictures. In dataset size, our dataset is much larger than other datasets.

In view of the above characteristics , we finally define the task as a temporal defect localization task based on the frame-level predictions, and the score obtained by this method was ten points higher than the second place. This report will focus on the descriptions of this method, while explaining the thinking ideas of the solution, and propose feasible strategies to further improve the score.

II. Data Preparing

The first is data preparation, including frame image extraction, data distribution observation, data multi-fold splitting and data enhancement. In terms of training dataset composition, we did not use any extra datasets.

A. Frame Extracting

In the frame extraction method, the size of our dataset reaches 130 Gigabytes. It takes about ten times the space of the video to extract all the frames on our disk, and it also takes up large computing resources. Therefore, we adopted a relatively special frame extracting strategy. In all the experiments on this challenge, we used the Decord library to extract frames. This library can read specific frames directly from video at a very high speed. During experiment. its speed is basically the same as the speed of reading frame images from disk, and it will only slightly increase the CPU usage. Therefore, based on the need to save storage resources, we do not actually store any frame images on our disk.

B. Folder Spliting

Usually we need to divide the data into multiple folders so that all samples can be learned in the model. Since this task is a multi-label classification task, and the data has a very significant long-tailed distribution, in order to divide the dataset into 5 different folders, and try to ensure that the distribution of training and validation samples in the folder basically conforms to the distribution of dataset, we used the iterative stratification from scikit-multilearn library.

C. Data Augmentation

In data augmentation, we mainly use horizontal flipping. We have also experimented with other data augmentation methods such as RandAug, AutoAug, and some common data augmentation methods such as vertical flipping, random cropping, rotation, color shift, etc. In the experiments, these methods will reduce our score, so we do not use these data augmentation methods.

III. Task Definition and Solutions

According to the thinking of the solution, the task can be positioned as the following three methods:temporal defect localization based on the frame-level predictions, temporal defect localization based on the frame-level annotations and temporal defect localization based on the segment-level annotations.

First is temporal defect localization based on the frame-level predictions, the general process is as follows: Use frame label data to generate pseudo-label samples, at the same time select a certain number of defect-free samples to build our training dataset which has 17 categories, then train an image classification network(like ConvNeXt) on the dataset, and finally find applicable post-processing methods to convert frame-level predictions to moment-level predictions.

Second method is temporal defect localization based on the frame-level annotations, which means using a single-frame supervision action localization network such as SF-Net [1] and LACP [2]. Works best in theory, but domain shift might be a problem.

Third method is temporal defect localization based on the segment-level annotations.which means using a segment-based action localization network such as ActionFormer. But we need to convert moment labels to pseudo segment labels, and for result, we also need to convert the segment predictions to moment predictions.

During experiment, the performance of Method 2 and Method 3 are relatively poor, so in this report, we mainly introduce the implementation details of Method 1.

A. Pseudo-label Samples Generation

If only one frame of labeled-moment is used for training, the number of samples may be seriously insufficient, so we need to generate a certain number of pseudo-label samples to train the image classification network. According to the observation, the marked moment can be understood as the moment when this type of defect appears, and 5 seconds after the marked moment in the video can also be regarded as the same label, so the samples with defects can be extracted from these 5 seconds.

After generating pseudo-label samples, we also need to generate a certain number of defect-free samples. However, we cannot directly use the area outside the marked 5s range for defect-free sampling, because it is observed that there are several defects in the 10 seconds before and after the marked moment, so we need to set a safety margin to ensure the correctness of defect-free samples, we set the margin to 10 seconds, as shown in Figure 1, now we can generate some defect-free samples from the blank part of the timeline. In actual training, the ratio of defect-free and defective samples is set to 1:1.

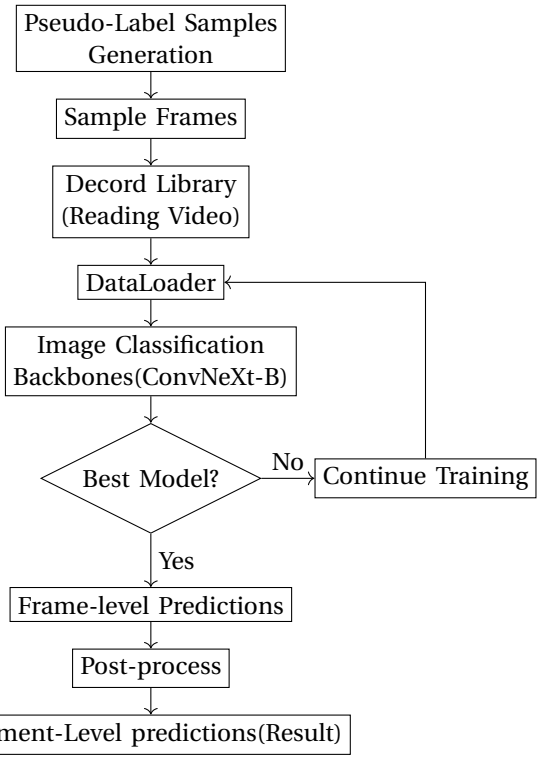

Fig. 2. Flow Chart of Method Based on the Frame-Level Predictions

B. Model

Since we have transformed it into a regular image classification task, we can use some image classification network for training. Here we use ConvNeXt [3], one of the most powerful convolutional neural networks available today. ConvNeXt has moderate parameters numbers, fast convergence speed, and outstanding performance. The score of ConvNeXt on ImageNet exceeds Swin, Vit and other models based on transformers, but it is much easier than transformer in terms of training difficulty. Therefore, ConvNeXt is the best choice for fast training and testing.

C. Training and Validation Pipeline

The overall flow of training and validation is shown in Figure 2. First, we generate some pseudo-label samples to form a dataset with 17 classes. Second, we sample some frames on the dataset. In actual training we use a sampling strategy with a 5 seconds frame-to-frame interval, the desired frames are marked in the form of a file list. Third, we use the Decord library to directly extract the frames marked in the file list from the video, and send them

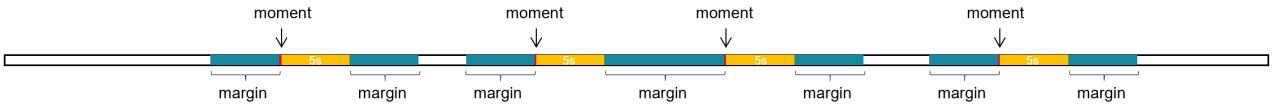

Fig. 1. Pseudo-label Samples Generation Example

to the dataloader for network training. It should be noted that the file list is not shuffled during training, this is mainly due to the trade-off of the video reading efficiency by Decord. If the file list is shuffled, Decord needs to frequently switch the video to be read. According to the experiment, the training time after shuffled is at least 5 times longer than the normal training time. Fourth, train a regular image classification network. Here we use ConvNeXt-Base. After getting the model with the highest validation mAP, use this model to infer on the test dataset to get frame-level predictions. Fifth, through experiments with different post-processing methods, we distill key moments from frame-level predictions as the final results.

D. Postprocess

Post-processing is an important step in this task. Our post-processing mainly includes the following steps.
First, output the prediction result curve and observe the features in the curve(Figure 3). The horizontal axis is the seconds of the video, and the vertical axis is the probability. Second, we found that in a single video, the peak moment of the prediction curve of each category can be understood as the most significant moment of the defect, but we cannot use the peaks from curve directly due to the poor smoothness of the prediction results, here we use the moving average algorithm to smooth our prediction.
Third, the curve features after smoothing are more obvious. We use the peak finding algorithm to find all the peaks in the curve of each category. Here we define that there will be no point with a higher probability than this point within 10 seconds before and after the peak point(Figure 3).
Fourth, by observing the deviation of the peak moment from the ground truth, we found that the moment at which the peak moment is located always lags behind the gt labeling moment, and the lag time is about 3-5 seconds (Figure 3).

In fact, it was mentioned before that the time point marked in ground truth is often the moment when the defect first appeared. But for prediction curve, peak moment should be the mid-point moment of the defect segment in the video. This is the reason for the deviation of the peak moment from the ground truth, which also proves the conjecture we made through the observation of the data from the side. At this point, the post-processing ends, and the result of the post-processing can be output as the final result.

TABLE I
POST-PROCESSING PARAMETERES SETTING

Post-process	Parameters
EMA or MA	MA
MA(EMA) Window Size	5
Peak Finding Interval	10
Peak Score Threshold	0
Moment Offset	-3 to -5s
Using ZC label	No

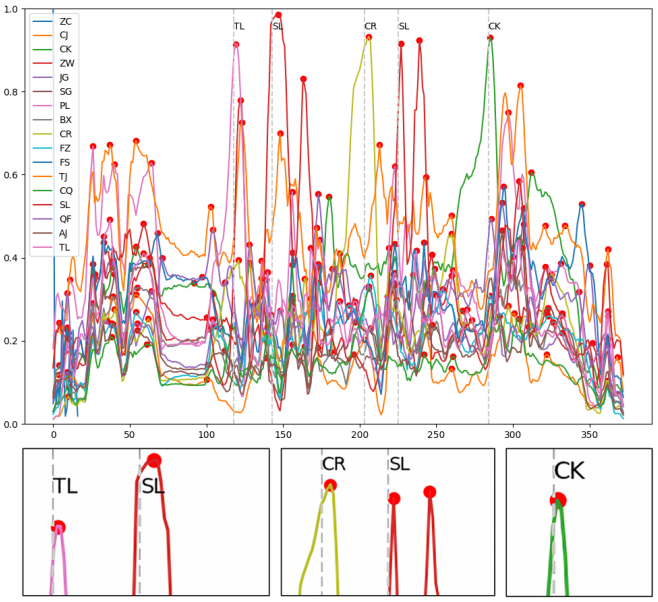

Fig. 3. Predictions Curve with Peaks and GT

E. Result

Based on the post-processed results, we achieved an average mAP of 17.653% on the Leaderboard(Table II), and the scores of method 2 and method 3 are shown in the Table III, which are more than 10% lower than frame-level prediction based method.

TABLE II
RESULT OF FRAME-LEVEL PREDICITON BASED METHOD

Model	Pretrain	mAP@ T=5(%)	mAP@ T=10(%)	mAP@ T=15(%)	Avg. mAP(%)
ConvNeXt Base	IN22kft1k 384	9.646	18.316	24.995	17.653

TABLE III
RESULTS OF OTHER METHODS

Action Localization Model	Feature Extractor	Local Avg.mAP(%)	LB Avg.mAP(%)
LACP	I3D	4.86	2.719
	ConvNeXt	3.09	4.044
ActionFormer	ConvNeXt	1.91	N/A

IV. BOOSTING TRICKS

We summarize some tricks used in the competition, as shown in the Figure IV and Figure V. At the data level, pseudo-label generation with larger safety margin is proved effective. When sampling frames, sampling with 5 seconds

TABLE IV
BOOSTING TRICKS

Level	Type	Description	Boosted(%)
Frame-Level Predictions Based	Data	Large margin size	2
		Sample frame every 5 frames	Local Val
		Iterative stratification by frame labels	Local Val
	Model	Use early models for preventing overfit	3
	Post-process	Adjust post-processing parameters	4.5
LACP	Feature Extraction	I3D Use small step	0.4
		I3D Keep flow features	0.2
		Change feature extractor to ConvNeXt (without flow features)	1.4
Action Former	N/A		N/A

TABLE V
LOWERING TRICKS

Level	Type	Description	Boosted(%)
Frame-Level Predictions Based	Data	Shuffle frame labels	Local Val
		Sample all frame	Local Val
		Iterative stratification by video labels	Local Val
	Model	Use 5 folders ensemble	-3.1
		Use bigger network	-1
LACP	Feature Extraction	Larger input size	-0.1
		Large output dim	-0.3
		More accurate optical flow estimation algorithm	-0.2
	Model	Change default training parameters	-0.5
Action Former	N/A		N/A

frame-to-frame intervals instead of sampling all frames also boosted our results. For 5 folder iterative stratification, split folder based on frame labels instead of video labels is proved to be effective on local mAP.

At the model level, since the verification uses mAP based on frame labels, which is completely different from the metric used in the competition. Therefore, after training several epochs, we manually select the models of some stages for testing. We found that models in the early and mid-stages perform much better than the models in the later stages. At the post-processing level, fine-tuning the post-processing parameters can also boost our score greatly.

For the model of action localization based on frame-annotations, the experiment mainly focuses on the feature extractor. For the I3D feature extractor, using a smaller step size in inference and retaining the optical flow information can slightly boost our score. But for larger input sizes, higher feature encoding dimensions, and using more accurate optical flow estimation algorithms have be proved ineffective.

V. CONCLUSIONS

- After several experiment on general action localization networks, we found that video action localization network based on the human action dataset may not suitable for this task, which is largely due to the shift of the domain. If the feature extraction network can be effectively pretrained on the dataset of this domain, the score will be improved a lot.
- We proposed a temporal defect localization method based on frame-level prediction, Although the method is effective and concise, it is not a long-term solutions, because this method largely relying on manual feature engineering and post-processing. We are still looking for an end-to-end solution.

REFERENCES

[1] F. Ma, L. Zhu, Y. Yang, S. Zha, G. Kundu, M. Feiszli, and Z. Shou, "Sf-net: Single-frame supervision for temporal action localization," 2020. [Online]. Available: https://arxiv.org/abs/2003.06845

[2] P. Lee and H. Byun, "Learning action completeness from points for weakly-supervised temporal action localization," 2021. [Online]. Available: https://arxiv.org/abs/2108.05029

[3] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," 2022. [Online]. Available: https://arxiv.org/abs/2201.03545