# The Solution of USTC-NERCSLIP team for UrbanPipe Track on Fine-grained Video Anomaly Recognition

Jun Yu[1], Xiaohua Qi[1], Zhihong Wei[1], Teng Zhang[2], Mohan Jing[1], and Zepeng Liu[1]

[1] University of Science and Technology of China
[2] Zhejiang University

**Abstract.** Video anomaly analysis is very important for industrial applications in the real world. In particular, urban pipeline system is one of the most important infrastructure of the city. In order to ensure its normal operation, our team uses a combination of video based methods and image-based methods to intelligently detect pipeline defects. The report focuses on the data preparation, task definition, model selection, training process and reasoning process of the team during the competition. This solution provides a feasible adjustment strategy, and we will also provide complete training and reasoning code for others to reproducing. The source code is available at https://github.com/tenforests/eccv_ubranpipe_track.

## 1 Model

In this UrbanPipe Challenge on Fine-grained Video Anomaly Recognition, we mainly used the following two models to complete the task.

### 1.1 Video method

The video-based method refers to taking the task as a conventional video classification task and using the video classification network to train and predict it. This task is a multi-classification problem of video.

We use Video Swin Transformer[7] as a model for video classification. This model is a pure transformer architecture for video recognition that is based on spatiotemporal locality inductive bias. This model is adapted from the Swin Transformer for image recognition, and thus it could leverage the power of the strong pre-trained image models. The proposed approach achieves state-of-the-art performance on three widely-used benchmarks, Kinetics-400 [2], Kinetics-600, and Something-Something v2 [5].

The overall process of the video-based method is a common video multi-classification process. We implement the overall framework based on the MMaction2 [3]. First, collect all the previously extracted frame images, sample the

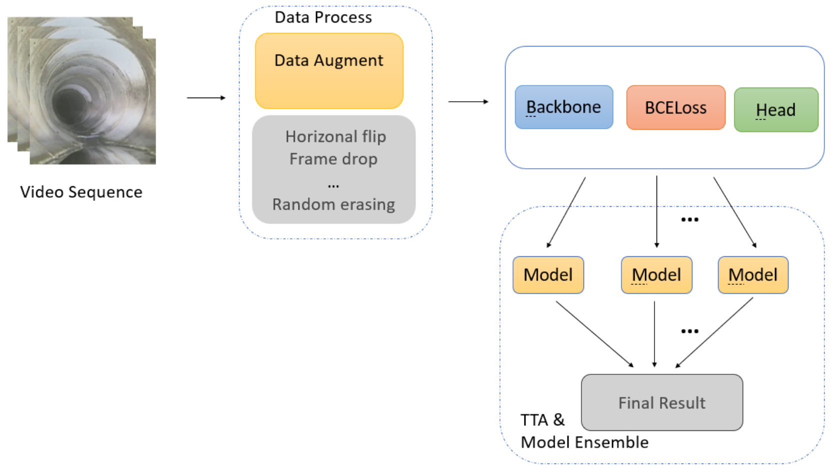

**Fig. 1.** Video method framework

| Model | Pretrain | Lr Schd | Params | bacbone | val mAP(%) |
|---|---|---|---|---|---|
| Video Swin Transformer | kinectic 400 | Onecyle 30E | 28M | Swin-T | 54.709 |
| Video Swin Transformer | kinectic 400 | Onecyle 30E | 50M | Swin-S | 62.230 |
| Video Swin Transformer | kinectic 400 | Onecyle 30E | 88M | Swin-B | 62.872 |
| Video Swin Transformer | kinectic 600 | Onecyle 30E | 88M | Swin-B | 63.667 |

**Table 1.** The training details of video swing transformers model and the results on the validation set

images using a unified sampling method, and send them to the dataloader. Secondly, it directly enters the training and verification process of the model, so it is a complete end-to-end network. In addition to the data preparation process, there is no other data preprocessing and post-processing, and the training process is clean, easy to understand, and repeatable.

We conducted relevant experiments on the model size, loading pre-training parameters, and other super parameters. In order to get the results faster, we sampled less than 12 frames from each video, with an interval of 5 frames, and divided the dataset into five parts, four of which were used for training and the other for verification. We used the results on the verification set to evaluate the performance of the model. Considering that the video reasoning is slow, we verified it every five epochs, training details are shown in Table 1.

## 1.2 Super-image method

We use the super-image method in Sifar [4] paper to transform the video classification task into an image classification task. The super-image method is to sample frames in the video and align several input frames into a super-image.

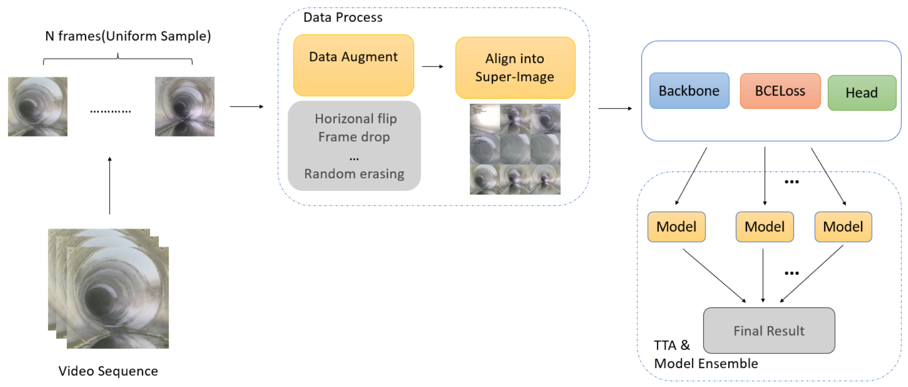

**Fig. 2.** super-image method framework

| Backbone | pretrain | Input Size | val mAP% |
|---|---|---|---|
| Nfnet F3 | B IN 1k (Input Size 416) | 672*672(224*3) | 64.6 |
| F Convnext Base | IN22Kft1K (Input Size 384) | 1334*1334(448*3) | 66.3 |
| Sifar (Swin-B) | kinetics400 (Input Size 224x3) | 672*672(224*3) | 65.9 |

**Table 2.** Super-image method training details and results on the validation set

In the experiment, we used the shape of 3x3 as the layout of the super-image grid. In the super-image experiment part, we replace the backbone network. The backbone networks used include Convnext Base [8], Nfnet F3 [1], and Sifar. We use the PyTorch framework to implement this part, and the pretrained model is from the Timm library.The detail are shown in table 2.

In the super-image method, we also carried out experiments on the model architecture. By connecting the ML-Decoder [10] after the Convnext Base, the classification head with the global average pooling and Full Connection is replaced with the attention-based classification head. The results show that the GAP classification head has better performance in the pipeline defect task.

## 2 Dataset preprocessing

The UrbanPipe dataset is a multi-label dataset with long tail distribution. Therefore, for the original video data, we divide the original data set into 5 different folders through iterative stratification method from scikit-multilearn [11]. This method ensure that the distribution of training and verification samples in each fold basically conforms to the distribution of the original dataset. Each time, we use four of folders for training and one for verification. We evaluate the model through AP and mAP on the validation set during the training process.

### 2.1 Video method

Considering that the video-based method occupies a large amount of video memory and our computing resources are relatively limited, we sample 12 frames of data from each video by simple sampling, and the interval between each frame of data is 5 frames. All the training sets are used to train the model.

### 2.2 Super-image method

In the Super-image method, we deal with the dataset. In addition to directly uniform sampling in the original video data, we also conducted offline processing. According to the given number of divisions, the video is uniformly sampled, and the corresponding frames are taken out and saved as pictures, so that the pictures of the corresponding frames can be read during the training process. For example, in the 9-frame dataset, we divided the video into 9 segments and saved the first frame of each segment. This method significantly helped us solve the IO bottleneck and accelerated the speed of training and verification process in the experiment. In this part, we try uniform sampling of 9, 13 and 27 frames.

When we review the data in the dataset, we found that There is non-pipe information at the beginning of some videos, such as street view. At the same time, there is also a time-series relationship between some data, and the beginning part of the latter data is the same as the end part of the previous data. Therefore, in the 13-frame sampling, we try to discard the frames 1, 2, and 12, 13 of the video to verify whether the information in the middle frames of the video covers the main defect information of the pipe. However, through verification, we find that most front and back frames of the video still play an important role in recognition. The loss of this part of the frame information leads to the decline of some class AP and leads to an unsatisfactory mAP.

We also compared the training results on the video dataset and the 27,9 frame dataset. Obviously, for the previous two datasets, random selection can be made in a larger sample space. For example, on the 27-frame dataset, for the first sub-image in a super-image, random selection can be made in between 1 and 3 frames each time.

After our verification, under the same training epochs, fixed nine frames showed the best generalization on the test set. Therefore, the later experiments were all completed on the 9-frame dataset.

## 3 Data Augmentation

### 3.1 Video method

We first used the data augmentation methods of the initial configuration of the MMaction 2 project, including random scale, crop, and flip.

We have also tried other data augment methods through experiments, such as cutout, autoaug, and some common data augment methods, such as Gaussian noise, random cropping, rotation, etc. through experiments, we found that these methods will reduce our scores, so we do not use these data augment methods.

### 3.2 Super-image method

Data augment in the super-image method is carried out for each sub-image. In terms of preprocessing, for each frame image, we do not change its scale, scale its short side to x, and then cut it into an image of size [x, x] by using the center crop method. We also conducted experiments on the impact of the input size of the sub-image on the model. The input size of 448 had better performance, but also brought greater training costs. After weighing, we finally used 448 input size on Convnext base, and 224 input size on other models.

We tried horizontal flip, random erasing, drop of the whole sub-image and random Gaussian blur of the sub-image, grid shuffle of the sub-image, TrivialAugment [9], sharpening and contrast change in data augmentation experiments. We also try to denoise the frames through the CBDNet [6]. The table 3 showing the verification set test results when these methods are used alone on the baseline.

However, when Some data augmentation methods are mixed, the performance of the model will decline. Therefore, for the model in the final verification stage, we only use two methods of horizontal flip and random erasing for training.

| Method | Result(Val mAP%) |
|---|---|
| Horizonal flip | +0.4 |
| Frame drop | +1.8 |
| Random erasing | +1.7 |
| Gaussian blur | -1.0 |
| Grid shuffle | -3.2 |
| Sharpen | -9.0 |
| Contrast | -4.0 |
| TrivialAugment | +1.0 |
| CBDNET | +0.3 |

**Table 3.** Data augmentation result in the validation set

## 4 Loss

In order to solve the problem of multiple tags and long tail distribution of data, we have carried out experiments on various loss functions on video-based methods.

Each pipeline may contain multiple tags, so we first used a binary cross entropy loss function. However, we found that the scores of "CQ", "TL", "SL" and other categories were very low by observing the scores of various categories on the validation set of the model, and we found that there was an imbalance in the data set categories by observing the distribution of the UrbanPipe dataset.

Therefore, we used class balanced focal loss to solve this problem. This class balanced focal loss is shown in Formula 1, 2 and 3.

$$CB_{focal}(z, y) = -\alpha(y) \sum_{i=1}^{C} (1 - p_i^t)_\gamma log(p_i^t) \tag{1}$$

$$p_i^t = sigmoid(z_i^t) \tag{2}$$

$$\alpha(y) = \frac{1-\beta}{1-\beta^{n_y}} \tag{3}$$

where $\beta$ controls the per class loss weight(Default: 0.9999). $\gamma$ is a hyperparameter of the focal loss(Default: 2.0). $z_i^t$ predicted output from the model. $n_y$ is the number of samples per class.

In Formula (3), $n_y$ is the number of samples per class, which is for a single classification task, but for a multi-classification task here, there may be multiple categories for each sample, and it is a problem to choose the number of categories as the number of samples.

Given this problem, we will first Part is removed for experiments, and then we select the number with the least category among all categories of the sample; Select the number with the largest category among all categories of the sample; The average of the number of all categories in the sample is selected, but through our experiments, these methods reduce the scores on the validation set. Therefore, we do not have a special method to solve the class imbalance problem but increase the robustness of the overall model through the fusion of multiple models.

Therefore, in the super-image part, because other loss functions do not perform well in video experiments, we only use binary cross entropy as the loss function.

## 5 Training strategy

### 5.1 Video method

In the video-based method, using the AdamW optimizer, the initial learning rate is 0.002.

### 5.2 Super-image method

In the super-image method, our batch size is controlled between 4 and 32 according to the model backbones used. At the same time, we used the EMA strategy in the training process, and 20 rounds of training and the same parameters were used in the training. The previous 10 epochs and the last 10 epochs of each model adopt different training strategies, and the specific training parameters are as table 4.

| epoch | init lr | optimizer | scheduler | warmup | min lr | high lr |
|---|---|---|---|---|---|---|
| previous 10e | 1e-6 | adamw | cosine | 5 | 1e-5 | 1e-4 |
| later 10e | 1e-5 | adamw | cosine | 0 | 1e-6 | None |

**Table 4.** Train detail in super-image method

## 6 TTA

### 6.1 Video method

In the video task, we use two TTA strategies, random framing, and randomResizedCrop, on the training set, random framing and centerCrop on the verification set, and random framing and threeCrop on the test set.

### 6.2 Super-image method

In the super-image task, we also tested the above two TTA strategies. However, the results show that random framing can not improve the model performance in the super-image method. The threeCrop method is not always effective. In the experiment, we found that this method has a positive effect on backbone models such as Sifar and Nfnet F3, but a negative effect on the Convnext Base. Therefore, in the model verification, we only implemented the threeCrop method for those models.

For the post-processing of the model, we observed that the model has a strong classification ability for class 0, so we ranked the classification results by the accuracy of Class 0. Set the prediction score of class 0 in the top 800 video to 1.0, and the prediction results of other class to 0. We verified this post-processing method on the single-fold model, which can improve the mAP performance by 0.1% - 0.2%.

## 7 Model Ensemble

### 7.1 Video method

Through the hyperparametric experiment, we can see that when using the swing base backbone and loading kinectic600, the performance on the verification set is the best. Therefore, we use this hyperparameter. To get higher scores, we train the other four data sets and the full data set again. For the other four data sets, we select the model that performs best on the verification set; For the full amount of data, we roughly selected the best round number position of the other five data sets when training the model; Then, the six models were fused at the ratio of 0.16 0.16 0.16 0.16 0.16 0.2. Through the fusion of the six models, the score after fusion is about 3-4 higher than that of the best one in a single model, which surprised us. Therefore, we consider fusing some other models that do not perform well, such as the model that uses focal loss and the model that samples 32 frames of images for training. But considering the limitation of our computing resources and the urgency of time, we did not conduct 50-fold cross-validation, but directly used the full data set for training, and then roughly selected a round of models to integrate into the six models above, In this way, we have eight models to fuse, and finally, the video-based method improves by about 2%.

### 7.2 Super-image method

In the super-image task, we trained the model with 5 folders data and full data respectively. The full data model was selected from the 20th epoch. The 5 folders model was selected from the best-performing epochs in the validation set. For the ensemble of the 5 folders model, were use 0.35, 0.3, 0.25, 0.2, and 0.1 to weight each folder according to the individual mAP performance in the test set, which can improve the result of the 5 folders ensemble by 0.1% - 0.3% compared with the equal weight ensemble. Then, the 5 folders result and the full quantity result are ensemble with half weight, which makes the model structure improve by more than 1%.

### 7.3 Dual model ensemble

Finally, we ensemble the models of the above two tasks after preliminary ensemble. We experimented with two ways: weighted and equal weight. In the weighting mode, the weight of the model is related to the test set mAP of the model. We give the weighting of one of the final experiments in table 5.

| model | weight |
|---|---|
| Sifar | 0.1 |
| Convnext Base | 0.4 |
| Nfnet F3 | 0.2 |
| Video Swin Transformer | 0.3 |

**Table 5.** Model ensemble weight

## 8 Conclusions

We first used the method based on video classification to solve this task, but we found that the video-based method requires a lot of computing resources and training time. Through observing the data, we found that there was no strong time-series information in the video, so we guessed that it might not be necessary to recognize their defects as actions, but only to learn the features in a single frame image. he Super-image method is the most effective and attractive method in this task because it can not only reduce computing resources, shorten training time, but also achieve higher scores. At the same time, we also found that adding random damage to Super-image can even improve our scores, which also proves that time information is not very important in this task. In view of the unbalanced distribution of data sets, we tried focal loss and improvements for multi-classification problems, but none of them achieved good results.

## References

1. Brock, A., De, S., Smith, S.L., Simonyan, K.: High-performance large-scale image recognition without normalization. In: International Conference on Machine Learning. pp. 1059–1071. PMLR (2021) 3
2. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017) 1
3. Contributors, M.: Openmmlab's next generation video understanding toolbox and benchmark. https://github.com/open-mmlab/mmaction2 (2020) 1
4. Fan, Q., Chen, C.F., Panda, R.: Can an image classifier suffice for action recognition? In: International Conference on Learning Representations (2021) 2
5. Goyal, R., Ebrahimi Kahou, S., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., et al.: The" something something" video database for learning and evaluating visual common sense. In: Proceedings of the IEEE international conference on computer vision. pp. 5842–5850 (2017) 1
6. Guo, S., Yan, Z., Zhang, K., Zuo, W., Zhang, L.: Toward convolutional blind denoising of real photographs. 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 5
7. Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., Hu, H.: Video swin transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3202–3211 (2022) 1
8. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11976–11986 (2022) 3
9. Müller, S.G., Hutter, F.: Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 774–782 (October 2021) 5
10. Ridnik, T., Sharir, G., Ben-Cohen, A., Ben-Baruch, E., Noy, A.: Ml-decoder: Scalable and versatile classification head. arXiv preprint arXiv:2111.12933 (2021) 3
11. Sechidis, K., Tsoumakas, G., Vlahavas, I.: On the stratification of multi-label data. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 145–158. Springer (2011) 3