Temporal Defect Localization on the CCTV Pipe Dataset

Santosh Thoduka
Department of Computer Science,
Hochschule Bonn-Rhein-Sieg
53757 Sankt Augustin, Germany
Email: santosh.thoduka@h-brs.de

***Abstract*—This report describes the method, training and inference process for the VideoPipe Challenge on Temporal Defect Localization. The CCTV Pipe dataset consists of long video clips recorded inside underground pipes, with annotations consisting of moments (i.e. timestamps) with the associated defect at that moment. A moment can contain multiple defects. There are a total of 16 defect classes. We train a ResNet-18 model, initialized by weights trained on the ImageNet dataset, with the addition of two linear layers for the final classification of a given frame into up to 17 classes (in a multi-label setting). For the training dataset, we sample 15 frames in a window of $\pm 2.5s$ around each annotated defect, and select additional random frames from the video, most of which are likely to be nominal frames. Several forms of data augmentation are performed on the images during training. During inference, frames sampled at 1 FPS. The code can be found on Github[1].**

I. Introduction

The VideoPipe Challenge on Temporal Defect Localization is concerned with detecting and classifying pipe defects that appear in long, untrimmed videos recorded in urban pipes. This report describes the method, training and inference process to solve the task. Submission of results on the test set are done through the Codalab website.

II. Dataset

The CCTV Pipe dataset consists of 346 training videos and 229 test videos, with an average duration of 545 seconds[2].

A. Training and Validation Split

We split the provided training set into 300 videos for training and 46 videos for validation, so that performance on the validation set can be used to validate models and methods before submission of results to the Codalab leaderboard. No additional or external data was used, with the exception of using weights for a ResNet-18 model trained on the ImageNet dataset (as described in Sec. III).

B. Class Distribution

The class distribution of defects is imbalanced. Moreover, since we consider nominal frames as an additional class, the nominal class has a much higher occurrence in the dataset, compared to any of the defect classes. We do not use any strategy to balance the class distribution of the defects. Nominal frames are sampled as described in the next section.

C. Sampling

Training is performed on single images; therefore we sample individual frames from the videos based on the following procedure. For each annotated defect moment in a training video, we sample 15 frames in a $\pm 2.5s$ window around the moment[3], with a minimum distance of $0.1 \times FPS$ between sampled frames. Therefore, if there are N annotated moments in a video, we sample $15N$ defect frames. In addition to sampling defect frames, we also randomly sample an additional $7.5N$ frames from the video. Since a majority of the frames are nominal, we expect these sampled frames to be primarily nominal frames. Labels are assigned to each frame based on the minimum distance to annotated moments. We label frames which are within 5 seconds of an annotated moment as a defect frame[4]. Since multiple defect classes can occur at a given moment, the label is a multi-hot encoded vector. All other frames are labelled as nominal.

For validation and testing, we sample frames evenly from the video at 1 FPS. For validation, the labels are assigned in the same way as for training.

D. Augmentation

We first resize each frame to have a height of 300 pixels, perform random cropping to size 300x300 and finally resize the image to 224x224 pixels. During inference, the process is the same, except that we center crop the image to 300x300 instead. During training, additional augmentation such as `RandomAdjustSharpness`, `RandomAutocontrast`, `RandomHorizontalFlip`, `RandomVerticalFlip`, `RandomErasing`, `GaussianBlur`, `RandomRotation` and `RandomAffine` are performed. The pixel values are then normalized to have mean = $[0.485, 0.456, 0.406]$ and standard deviation = $[0.229, 0.224, 0.225]$.

[1]https://github.com/sthoduka/cctvpipe_localization
[2]https://videopipe.github.io/cctvpipe/index.html

[3]The evaluation metric allows for defect detections within 5, 10 and 15 seconds of the annotated defect; therefore, in principle the window can be extended to $\pm 15s$.
[4]Similar to the sampling window, this threshold can also be extended up to 15 seconds

III. Method

As the base model, we use a ResNet-18 model which was pretrained on the ImageNet dataset (obtained from the pytorch repositories[5]). The last classification layer is removed, and are replaced by two linear layers with a `Hardswish` activation and `Dropout` layer in between. The ResNet-18 weights are unfrozen and allowed to train with the same learning rate as the rest of the network.

The final layer has 17 outputs, one for each defect class and the nominal class.

1) Training Process: We use a batch size of 512 for training, using Stochastic Gradient Descent as the optimizer, with momentum of 0.9 and a weight decay of $5 \times e^{-4}$. We use a One Cycle learning rate scheduler[1] with two phases, with a maximum learning rate of 1.0. We use focal loss[2] as the loss function (i.e. each output of the final layer is expected to be a value between 0 and 1, with 1 as the positive class). The labels are represented as a multi-hot encoded vector with the positive class represented by 1, with the remaining classes set to 0. The mean average precision (mAP) on the validation set is monitored through the training process, and the model with the best mAP is chosen for producing the results with the test set. Since the validation mAP is not calculated in the same way as mAP calculated by the test server[6], the validation mAP does not follow the same trend as the test mAP.

2) Inference Process: During inference, no data augmentation is used and, as mentioned previously, frames are classified at 1FPS. The scores for all sampled frames are stored for both the validation and test set.

In a second step, we construct an ROC curve on the validation scores for each class and determine the best threshold for each class[7]. Since the validation set is quite small, and some classes already occur quite rarely, one of the classes does not occur at all in our fixed validation set. Therefore, we use a fixed threshold of 0.1 for this class (FZ).

For the test set, we first use the identified best threshold for the nominal class to determine if a frame is nominal or contains a defect. If the score is below the threshold (i.e. it contains a defect), we select the defect classes at that moment which exceed the threshold for their respective classes. This process results in a high number of false positives (though it is lower than if a fixed, class-agnostic threshold is used); therefore a more robust method of binary classification of frames to nominal and defect should be developed.

IV. Conclusion

This report describes our approach for the Track on Temporal Defect Localization of the VideoPipe challenge, including the use of the dataset, learning model used, training and inference process and data augmentation. A ResNet-18 network was used as the base-model (pretrained on ImageNet), and focal loss was used due to the imbalanced dataset. Training frames were sampled based on their proximity to defects in addition to random frames, whereas validation and test frames were sampled evenly at 1 FPS. The scores from the validation set were used to determine the ideal threshold for each class. These thresholds were used on the test set, first to determine if a frame had any defect, and second to determine the defects at that moment if their scores exceeded their respective thresholds.

[5]https://pytorch.org/vision/0.9/models.html#classification

[6]i.e. our validation mAP does not consider 3 separate temporal windows for assigning true positives

[7]we choose the threshold at which we get the maximum geometric mean of the TPR and 1 - FPR

References

[1] L. N. Smith and N. Topin, "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, 2019, p. 1100612.

[2] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.