



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공학석사학위논문

안정적인, 비디오 기반의 공정 이상 탐지

Robust Industrial Video Anomaly Detection

2021 년 8 월

서울대학교 대학원

기계공학부

진 영 욱

안정적인, 비디오 기반의 공정 이상 탐지

Robust Industrial Video Anomaly Detection

지도교수 박종우

이 논문을 공학석사 학위논문으로 제출함

2021년 4월

서울대학교 대학원

기계공학부

진영욱

진영욱의 공학석사 학위논문을 인준함

2021년 6월

위원장 : 이 경 수

부위원장 : 박 종 우

위 원 : 이 동 준

ABSTRACT

Robust Industrial Video Anomaly Detection

by

Young-Uk Jin

Department of Mechanical Engineering

Seoul National University

Industrial video anomaly detection is an important problem in industrial inspection, possessing features that are distinct from video anomaly detection in other application domains like surveillance. No public datasets pertinent to the problem have been developed, and accordingly, robust models suited for industrial video anomaly detection have yet to be developed. In this thesis, the key differences that distinguish the industrial video anomaly detection problem from its generic counterparts are examined: the relatively small amount of video data available, the lack of diversity among frames within the video clips, and the absence of labels that indicate anomalies. We then propose a robust framework for

industrial video inspection that addresses these specific challenges. One novel aspect of our framework includes a model that masks regions in frames that are irrelevant to the inspection task. We show that our framework outperforms existing methods when validated on a novel database that replicates video clips of real-world automated tasks.

Keywords: Industrial Video Anomaly Detection, Robustness, Mask, Attention Map, Weakly Supervised Learning, Cycle

Student Number: 2019-28107

Contents

Abstract	i
List of Tables	vi
List of Figures	viii
1 Introduction	1
1.1 Related Works	8
1.2 Contributions of Our Work	15
1.3 Organization	16
2 Preliminaries	18
2.1 Weakly Supervised Learning	19
2.1.1 Supervised Learning and Unsupervised Learning	19
2.1.2 Demystification on Weakly Supervised Learning	21
2.2 Class Activation Maps	22
2.2.1 Overview on Visualizing Activations	22
2.2.2 Overview on CAM	24

2.2.3	Overview on Grad-CAM	25
2.2.4	Overview on Eigen-CAM	26
2.3	Dynamic Time Warping	27
2.4	Label Smoothing	28
2.4.1	Review on Cross Entropy Function	30
2.4.2	Summary on Label Smoothing	31
3	Robust Framework for Industrial Video Anomaly Detection	32
3.1	Components of the Framework	34
3.1.1	Anomaly Detection Model	34
3.1.2	Background Masking Model	34
3.1.3	Fusing Results from the Components of the Framework	37
3.2	Details of the Weakly Supervised Learning Method	37
3.2.1	Partition Order Prediction Task	37
3.2.2	Partition Order Labels	39
3.2.3	Conditioning the Labels	43
4	Experiments	45
4.1	Database for Industrial Video Anomaly Detection	46
4.2	Ideal Background Mask	48
4.2.1	Acquiring Ideal Masks	48
4.2.2	Enhancing Robustness in VAD Using Masks	48
4.3	Masking Using the Proposed Method vs Using an Ideal Mask	49
4.4	Performance Enhancement Using the Proposed Method	50
4.5	Ablation Study	54
4.5.1	Number of Layers for Eigen-CAM	54
4.5.2	Threshold on Attention Maps	54

4.5.3	Temporal Smoothing Window Size	56
5	Conclusion	57
A	Appendix	59
A.1	Experimental Results for All Tasks in the Database	59
	Bibliography	60
	국문초록	68

List of Tables

4.1	Statistics of the Database	47
4.2	Results regarding task0. Results obtained using an ideal mask. . .	51
4.3	Results regarding task 0. Comparison of results obtained when using an ideal mask and the mask generated from the background mask- ing model.	52
4.4	Results regarding task 0. Comparison of results obtained when using no masks and the masks generated from the background masking model.	53
4.5	The effect of changing the number of layers used for Eigen-CAM. .	55
4.6	The effect on the performance of the framework when changing the threshold applied to visual attention maps to obtain masks.	55
4.7	The effect on the performance of the framework when changing the window size for smoothing applied along the temporal axis of visual attention maps to obtain masks.	55

A.1	Results regarding task 0. Comparison of results obtained using no masks, an ideal mask, and the mask generated from the background masking model.	60
A.2	Results regarding task 1. Comparison of results obtained using no masks, an ideal mask, and the mask generated from the background masking model.	61

List of Figures

1.1	Examples of samples within datasets for image-based product inspection. [1] The DAGM texture dataset, where the objective is to find cuts, ruptures, and stains on images of various fabric samples. [2] The MVTEC-AD dataset, where the task is to recognize and locate anomalies on the surface of diverse industrial artifacts.	3
1.2	An intuitive example illustrating the importance of detecting erroneous processes that occur during manufacturing tasks.	4
1.3	Exemplary normal and abnormal samples extracted from the UCSD Pedestrian Dataset.	5
1.4	An illustrative example that depicts the case when existing frameworks for video anomaly detection fail in the problem’s industrial variant. When unseen, random, yet irrelevant changes occur in videos, the model fails to generalize to these clips, thus triggering unwanted alarms.	7
1.5	Representative scenes in single scene video anomaly detection and multi-scene video anomaly detection. [1] The single scene case, and [2] the multi-scene variant.	9

1.6	An example of a popular dataset used for video segmentation. [1] The input frame. [2] The corresponding segmentation ground truth map.	13
2.1	Examples of self-supervised tasks. [1] Jigsaw puzzle case, [2] occluded fragment prediction case.	20
2.2	Comparison of methods used to obtain visual attention maps. [1] Grad-CAM, [2] Eigen-CAM.	23
2.3	Illustration on the method implemented in Eigen-CAM.	27
2.4	Illustration of a situation where hard, one-hot encoded labels may degrade performance when training a model.	29
3.1	Entire grasping pipeline.	33
3.2	Illustration of the CLSTM auto-encoder used to detect anomalies in video frames.	35
3.3	Illustration of the network that creates masks that veil regions in the frames irrelevant to the video inspection task.	36
3.4	Illustration on the method used to obtain weak labels.	40
3.5	Comparison of the one-hot encoding labeling scheme and the smoothed labeling scheme.	42
4.1	Exemplary scenes sampled from the proposed industrial video anomaly detection database.	47
4.2	Results regarding task 0. Visualization of the effect of using an ideal mask, generated using an FCN. The segmentation labels used to train FCN were obtained by manually designating regions irrelevant to the inspected task.	51

4.3	Results regarding task 0. Comparison of the visualization of ideal masks and masks obtained using the background masking model. As shown in this figure our model masks out regions similar to that of the ideal mask.	52
4.4	Results regarding task 0. Comparison between the qualitative results obtained when no masks were applied and when the mask obtained from the background masking model is applied.	53
A.1	Result obtained on task 0. A comparison on the appearance of the raw frames, those masked using ideal masks, and ones overlaid with masks obtained using the background masking model.	60
A.2	Result obtained on task 1. A comparison on the appearance of the raw frames, those masked using ideal masks, and ones overlaid with masks obtained using the background masking model.	61

1

Introduction

The rapid expansion in automated manufacturing triggered the need for faster and accurate inspection. Unlike the significance of the task, however, automated inspection has not received much attention for quite a long time. Recently though, the field gained popularity, and various computer vision algorithms for automated inspection have emerged. This interest in the field was reinforced with an increase in interest in developing machine learning algorithms.

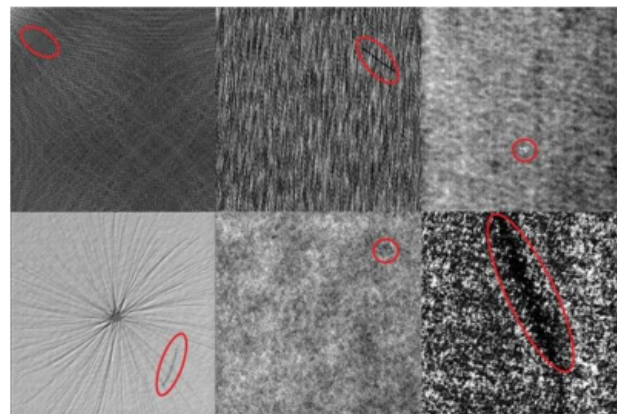
Unlike their rule-based counterparts, [1], machine-learning-based computer vision inspection algorithms are flexible and cost-effective. When programming inspection algorithms in a rule-based fashion, the method's deterministic style becomes troublesome in industrial inspection because the appearance and location of defects are almost impossible to predict. Unless all possible abnormal depictions of a product are considered, some cases of anomalies may never be labeled as erroneous. On the other hand, machine learning-based algorithms exploit a model trained on various images and learns patterns that aid in discerning errors out of regular instances. Therefore the method demonstrates greater flexibility compared

to deterministic methods for anomaly detection.

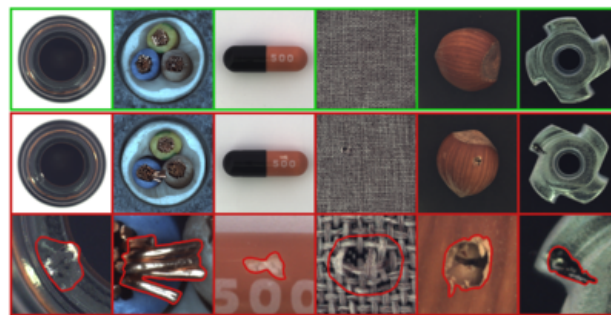
Moreover, rule-based models quickly become obsolete when unexpected changes in the environment surrounding products occur since most of the variables defined within their algorithms were tuned under specific conditions. When such abrupt alternations in the surroundings do occur, all variables require tuning, or lines of code may need to be modified completely. Contrary to rule-based models, frameworks that use machine-learning methods only require their models to be trained on new data acquired after the shift in the environment. During this procedure, parameters that define the model and algorithm are tuned automatically and more precisely than when it is done by hand. Machine-learning methods require less effort and time than their deterministic counterparts. Thus they are easier to deploy to various industrialized settings.

Most research regarding industrial inspection is centered on image-based inspection on the exterior of products. Various public datasets have been proposed to facilitate research on the problem [2], and diverse methods have been devised to tackle the image inspection task [3, 4, 5, 6].

In retrospect, though, faulty products are usually the results of erroneous manufacturing processes. For instance, incorrect assembly of a part moving on a conveyor belt may cause a sequence of errors downstream, with the end result being a grasping error that damages the part. Thus, industrial inspection based on video clips is as critical as the image-wise inspection problem. Despite its significance, the subject has yet received sufficient interest. Rule-based methods for video-based inspection algorithms do indeed exist. However, the methods lack flexibility and are costly compared to their machine-learning-based counterparts. Therefore, it is natural to seek machine-learning video anomaly detection algorithms, which has not received much attention in the literature.



(a)



(b)

Figure 1.1: Examples of samples within datasets for image-based product inspection. [1] The DAGM texture dataset, where the objective is to find cuts, ruptures, and stains on images of various fabric samples. [2] The MVTEC-AD dataset, where the task is to recognize and locate anomalies on the surface of diverse industrial artifacts.

In contrast to its application to industrial problems, video anomaly research on more general use-cases has enjoyed a recent surge in interest due to the increasing need for everyday applications based on analysis of video clips. The most prevalent

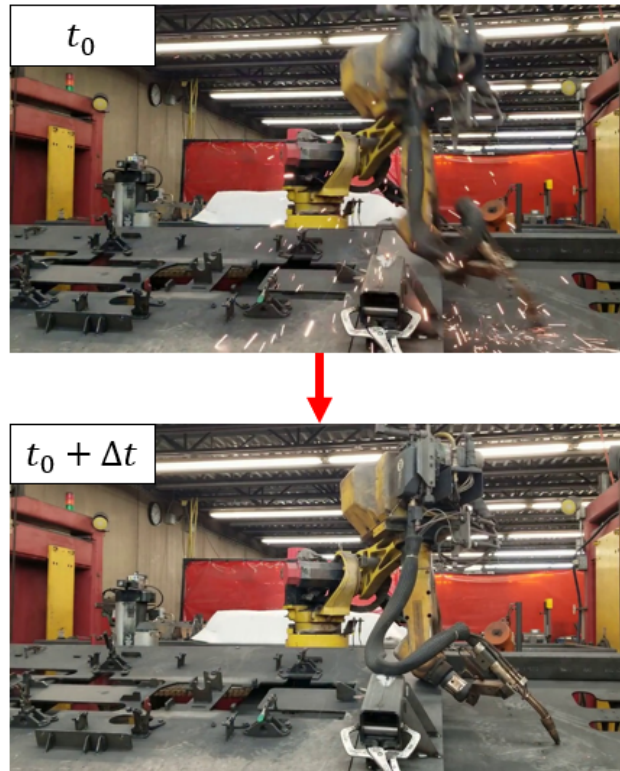


Figure 1.2: An intuitive example illustrating the importance of detecting erroneous processes that occur during manufacturing tasks.

problems among such tasks are those regarding surveillance. Numerous frameworks have been proposed for video anomaly detection on these surveillance video clips gathered from cameras installed in various places [7, 8, 9, 10].

These frameworks, however, are not suitable for the industrial variant of video anomaly detection for three reasons. First, surveillance video clips contain typical actions and abnormal actions, usually depicted with a stark difference in appearance. On the other hand, due to the perfection of machinery, almost no abnormal actions are observed nor recorded in manufacturing video clips. Thus unlike in the



(a) CUHK Avenue



(b) UCSD Pedestrian

Figure 1.3: Exemplary normal and abnormal samples extracted from the UCSD Pedestrian Dataset.

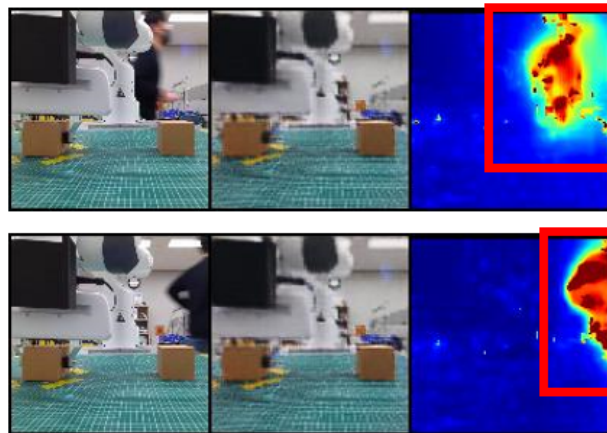
surveillance application of video anomaly detection, where clips can be labeled, and such clip-label pairs can be used to teach a model to discriminate between normal and abnormal clips, supervised learning is not a valid option in industrial video anomaly detection. Frameworks for industrial video anomaly detection should be trained solely on normal data without labels, which hint the model on which features within the data should be prioritized.

Second, unlike surveillance video clips, which can readily be gathered or updated, unless permission to access the data has been declined, industrial video clips

are challenging to accumulate due to the short time available to gather the data before the manufacturing task is updated. Due to the same limitations, repetitive frames with similar appearance and hardly any variations are usually sampled and used to train an inspection framework. These restrictions make industrial video anomaly detection more complicated to solve. Usually, in machine learning, data used to train models are expected to satisfy two crucial requirements for the resulting model to show more significant performance and generalize better: 1) sufficient size and 2) sufficient variation among data. However, as emphasized, data available in the industrial video anomaly detection setting lack these aspects, making the problem more complicated.

Not surprisingly, when adopted to industrial video anomaly detection, existing frameworks show poor performance. The main issue that degrades their performance is their failure in generalizing to irrelevant features in video clips. Because abnormal clips are not available for training the models, frameworks must rely only on usual clips to learn patterns. Furthermore, these clips are gathered in small numbers due to the insufficient time to film the data.

Third, the data lacks variations in appearance. Industrial videos usually feature a worker, a workspace, and the background, where all features irrelevant to the workspace exist. Anomalies to catch occur in the workspace and not in the background. Therefore, any variations in the background are out of interest. The problem is that there are hardly any changes in the background of data in the train-set, causing the inspection model to fail to generalize to these irrelevant and subtle deviations.




 : False anomaly due to changes outside the region of interest

Figure 1.4: An illustrative example that depicts the case when existing frameworks for video anomaly detection fail in the problem’s industrial variant. When unseen, random, yet irrelevant changes occur in videos, the model fails to generalize to these clips, thus triggering unwanted alarms.

Due to the limitations in data used to train the frameworks, they become susceptible to irrelevant changes outside the region of interest within frames, triggering a myriad of false alarms. This frequent occurrence of false alarms may be troublesome when the method is deployed to the field. Frequently responding to the alarms and halting industrial processes leads to a decrease in total yields. Thus, a framework that demonstrates greater robustness to the problem is sought-after.

In this work, we propose a novel framework robust to changes in video frames irrelevant to industrial video anomaly detection. Instead of observing the entire image within each frame, our framework applies a learned mask to the input frames,

which veils regions that lay outside the zone in which the industrial process is taking place. This way, although random changes, not shown when training the inspection framework, occur in parts of each frame where no work is being done, the mask guides the framework not to consider these partitions. For convenience, let us now shorten the phrase "the region where work is being done" into a compact term "work-space" and the region outside the work-space, "background."

Our background masking framework consists of two parallel models: 1) the anomaly detection model and 2) the background masking model. Any existing method developed previously for video anomaly detection may be applied for the anomaly detection model, but we adopted an auto-encoder-based approach for convenience. The novel component of our framework is the visual attention-based background masking model, which conveys the masking mentioned above on the background of each scene and encourages the framework to focus on regions pertinent to the workspace.

Moreover, we propose a small database that suits the conditions that differentiate data available for industrial video anomaly detection from its more general counterparts, such as surveillance. Using this small database, we train and evaluate our model and show that using our framework enhances robustness to the data compared to when no such background masking is used.

1.1 Related Works

Video anomaly detection (VAD) has recently gained popularity, and numerous papers on new methods have been published. The problem can be classified into two sub-problems on the grandest scale: 1) multi-scene anomaly detection, 2) single-scene anomaly detection. In multi-scene anomaly detection, a camera is attached



Figure 1.5: Representative scenes in single scene video anomaly detection and multi-scene video anomaly detection. [1] The single scene case, and [2] the multi-scene variant.

to a moving platform; thus, the background in the scene is constantly changing. On the other hand, in the single-scene case, the camera is fixed to a particular position and orientation. In industrial VAD, most cameras for inspection are installed on a fixed holder. Therefore the problem is one example of the single-scene case.

Before introducing some methods previously devised for single-scene VAD, we summarize some existing datasets for the problem. The UCSD Pedestrian dataset

provides gray-scale video clips filmed on roads on campus, where anomalies include scenes of people riding a bicycle or employees traveling through the sidewalk on a golf cart. The CUHK dataset features scenes filmed on a busy road. Unlike in typical scenes where people show dull movements such as moving in the same direction, reading a newspaper, or standing still, abnormal scenes contain unforeseen motions such as a man throwing his bag or walking in the wrong direction. The Street Scene dataset is the most recent single monocular surveillance video dataset, filmed on several streets, where anomalies are abnormal activities on the road.

Various research has been done on single-scene VAD, though the specific use-case had been limited chiefly to surveillance tasks. Although many other approaches exist, most research has focused on using an auto-encoder as the primary choice of model for anomaly detection. The simplest among such methods was introduced in [11]. In this work, the authors implement an auto-encoder with an LSTM model embedded in the bottleneck. Unlike most auto-encoders, the LSTM auto-encoder encodes the latent information along the temporal axis, which is required since frames within video clips are dependent on each other concerning time.

[12] introduce a variant of [11], where they use a 3D convolution network to encode temporal information instead of using a separate LSTM model for the same purpose. A 3D convolution block is a simple variation of the more popular 2D convolution filter, where an additional depth axis is added apart from the existing height and width. By manipulating the stride of the filter applied along the depth axis, part of the sequence of video frames is compressed, where the compressed information contains time-wise information extracted from the input video. The model is more compact and straightforward to implement than the former LSTM auto-encoder model. However, it has a smaller receptive field along the temporal

axis than the LSTM or RNN based models.

Unlike the studies mentioned above, The authors of [13] propose a prediction-based video anomaly detection method. In this method, when provided several frames in a sequence, the prediction model learns to predict the contents of the unseen part of the same sequence. If an anomaly occurs, the model fails to predict this change. An advantage of this approach is that the bottleneck of data is more intense than that of the reconstruction case. Unlike reconstruction-based methods, prediction-based models use compressed, encoded data to guess the most probable appearance of consecutive images with dimensions of much greater size. The drawback, however, is that when too many frames are to be predicted, the models perform poorly.

Finally, in [14] , and [15] the authors apply a memory-guided auto-encoder model to industrial VAD. In this approach, instead of using RNN modules to encode temporal information, a memory module is used to memorize representations of frames input to an encoder. During the training phase, the model receives clips containing only typical frames, learning to extract and memorize compressed features that contain information sufficient to reconstruct frames in train-set video clips. After training, when the model is shown clips similar to the train-set data, it successfully reconstructs the frames. On the contrary, when abnormal clips are provided, the model tries to reconstruct frames within the train-set, which are most similar to these anomalies. Because irregular frames look very different from typical frames, the model triggers an alarm, indicating an anomaly within the clip.

Although these models demonstrate state-of-the-art performance on public VAD datasets, the models are unsuitable for video anomaly detection. As aforementioned in this literature, public datasets are limited to the specific task of surveillance, which shares characteristics distinct from industrial VAD. Models developed

and evaluated on the existing datasets, thus, fail to show robust performance when adapted to industrial VAD tasks.

Indeed, video frame segmentation may be applied to enhance the robustness of video anomaly detection models. By segmenting videos into partitions according to their similarity in semantic information they carry, a mask that hides the background of industrial tasks can easily be obtained. However, since data with little or no variations in appearance is provided to train models, many existing methods for segmentation on video clips become outdated. For instance, the methods introduced in [16, 17] are developed on popular public video datasets such as the DAVIS dataset [18], those that contain ground-truth segmentation masks.

In [16], the authors suggest a simple framework that trains on only the first frame of each video clip to produce masks for the whole clip during the inference phase. The segmentation model is implemented by taking the backbones of popular image classifiers pre-trained on image datasets and replacing the final linear layers with upscaling layers that magnify the embeddings of the backbone to the size of input images.

[17] proposes a novel approach based on the intuition that most consecutive frames in video clips do not show a significant discrepancy between each other. The method consists of two steps: 1) coarsen and feed the model with a mask predicted on the frame preceding the current frame in time, 2) upon receiving this mask and the current frame, predict the mask for the current frame. The authors use both offline and online training to boost the performance of the model.

Both methods, though they perform well on video segmentation tasks, rely on ground-truth segmentation masks. In the industrial VAD problem, due to the limitation of diversity and size of acquired data, video clips with varying appearances



Figure 1.6: An example of a popular dataset used for video segmentation. [1] The input frame. [2] The corresponding segmentation ground truth map.

paired with ground truth segmentation results are challenging to obtain. Therefore, the two methods and other methods not listed in this literature that share similar strategies do not suit our framework.

[19] introduces a method that considers the human gaze for segmentation tasks. In this research, pairs of data and human gaze fixation maps are provided for training the model. Here, the term human gaze fixation map refers to a map with the size of the input frames, whose elements that constitute the array indicate whether a human volunteer has stared at the respective pixel or not. However, this method is inappropriate for industrial VAD due to the difficulty of gathering the human gaze labels corresponding to each frame. A gaze tracker must gather information on the human gaze, which is costly and redundant.

An unsupervised line of research on video frame segmentation has recently emerged [20, 21]. These methods usually exploit optical flow maps obtained from frames adjacent to each other in videos. In [20], the authors first introduce a generator that attempts to guess a mask that hides relatively static regions in an optical flow map. Then they propose another model named the inpainter, which attempts to predict the contents of the regions that were masked out using the mask generated from the generator. The two models are trained simultaneously, where the generator tries to learn a perfect mask that separates dynamic and static regions in each flow map, whereas the inpainter seeks to complete the masked region perfectly so that the prediction matches the input flow map. By competitively training the generator and inpainter, one can obtain a mask that effectively hides video frames' background.

The more recent paper, [21], demonstrates that an auto-encoder with a slot attention module embedded in its bottleneck may be used to create a mask that removes the background region from input videos. A slot attention module [22] is a model that groups regions in each image according to the semantic similarity of features that constitute them, like that of capsule networks [23]. The module was initially introduced with test results obtained from an evaluation dataset consisting of features with simple textures and appearances. The authors of [21] assert that optical flow arrays consist of pixels that show similar values to their neighbors, thus displaying an image with simple textures. The slot-attention auto-encoder then learns and generates segmentation maps for video clips using these flow maps as given inputs.

Although both methods are relatively favorable compared to those mentioned above, they both rely on optical flow maps, arrays that are relatively difficult to determine or estimate. State-of-the-art models for optical flow maps indeed exist,

but most of these models are trained on labeled video datasets, which contain synthetic video clips. Furthermore, the flow map estimators require input video clips with high resolution, data that is difficult to acquire from relatively cheap cameras installed in factories, in great numbers.

1.2 Contributions of Our Work

The main contribution of our work is the proposal of a novel and robust framework for industrial video anomaly detection. To elaborate more deeply into our proposed framework, the model consists of two main components: 1) an anomaly detection model and 2) a visual attention-based background masking model.

The former is the part of the framework trained on normal samples to detect abnormal frames in video clips during the inference phase. Any existing method for video anomaly detection may be applied to this part of the framework, but for convenience, we applied an LSTM auto-encoder with a structure similar to that of [11].

The latter is the novel part of our framework and is the critical contribution of our research. The model predicts a mask using a visual attention map computed from a model trained to leave features relevant to the task being done in each scene of industrial video clips intact. On the other hand, the background is masked out and unused for computing the final score map that decides whether or not to indicate an anomaly. Because random changes in the background do not correspond to an error in an industrial process, masking them out prevents the framework from generating false alarms, which leads to enhanced robustness of the method.

The background masking model is trained on a sequence order prediction task.

To elaborate on this task, a cycle, the most prominent feature in the industrial inspection video clips, is divided into several partitions, which are assigned labels that indicate the temporal order of the partitions within the cycle. The model has to sort each frame in a video to one of the partitions that constitute a complete cycle. To make a correct guess, the model focuses on the repeating motions in video clips. If so, regions in the output activation maps of each layer of the model pertinent to such component in each frame will be activated the most, which results in high visual attention values.

Since there are no existing datasets to test our framework designed for industrial VAD, we make our second contribution: a database designed to satisfy the characteristics of the problem. With this simple database, we show that our method effectively reduces false anomaly alarms while correctly detecting actual anomalies in videos.

1.3 Organization

In chapter 2, we cursor through the prerequisite terms required to understand the methods adopted to implement our framework. First, we summarize the difference between weakly supervised learning to unsupervised and supervised learning. This short explanation will help the readers understand why our method of training the background masking model is classified as a weakly supervised task. Second, we briefly review the method of class activation map (CAM) and those of its variants, grad CAM and Eigen CAM. This summary will help demystify the method used to obtain the background masks from the respective model. Third, we introduce the concept of dynamic time warping (DTW), a strategy used to map sequences that are asynchronous but display very similar aspects. Explanation of this idea

will guide the readers to understand how we obtain the weakly supervision labels. Fourth, we take a brief look at label smoothing, a method required in our research when labeling frames that are adjacent in time and look similar to each other but are assigned to different partitions in each cycle.

Upon moving on to chapter 3, we introduce how we implemented our robust framework for industrial video inspection. We use two models, each for anomaly detection and background masking. The framework can easily be added to existing video anomaly detection models, where the background masking module minimizes the susceptibility of the framework to unseen random yet irrelevant changes in the frames of the video clips.

Then in chapter 4, we share implementation details for our model and database for evaluation. We then reveal the results of experiments done using our framework. We compare the performance of our method to that of a simple implementation of one of the existing methods. We also compare the robustness of our framework to another framework that uses an ideal mask to veil the background.

Finally, in chapter 5, we sum up all results and contemplate the advantages and drawbacks of our model. We then propose further plans that extend from this research and some lines of studies that may probably be undertaken in the future.

2

Preliminaries

In this chapter, we skim over some crucial concepts required to understand the successive literature in this paper. First of all, we clarify the difference between weakly supervised learning and other lines of machine learning, namely, supervised and unsupervised learning. Then we briefly overview the class activation map (CAM) concept and compare two variants of the method, Grad-CAM and Eigen-CAM. Furthermore, we review the dynamic time warping required to train the background masking model when creating the weak labels. Finally, we summarize the concept of label smoothing, a method widely used when data is noisy, and the boundaries that divide clusters of data are fuzzy.

2.1 Weakly Supervised Learning

2.1.1 Supervised Learning and Unsupervised Learning

Before scrutinizing the concept of weakly supervised learning, and comparing the method to other approaches, let us shortly review the definitions of supervised learning and unsupervised learning. In a supervised learning problem, a straightforward reference, or in other words, ground truth labels, is provided to guide a model to learn to operate on a specific machine learning task such as classification, detection, or segmentation. The point here is that such reference corresponds to the desired output of the model.

On the other hand, unsupervised learning is defined as a machine learning problem, where no labels are provided when training the model. Under this setting, models should exploit features within the given data to obtain hints that help them perform well on the task at hand. Various clustering methods introduced in classical machine learning literature are examples of solutions to the unsupervised learning problem.

With the advent of powerful CNN-based feature extractors and various deep-learning-based techniques, new methods designed for unsupervised learning, known as self-supervised methods, have been introduced. The term self refers to the data itself, whereas the word supervision indicates that the provided data is used to supervise the model. Combining the two words leads to self-supervised learning, which means that a model uses the input data as a reference to guide them to extract useful features. To do so, some tasks that can exploit the input data are precariously designed to guide models to extract semantic information within them.

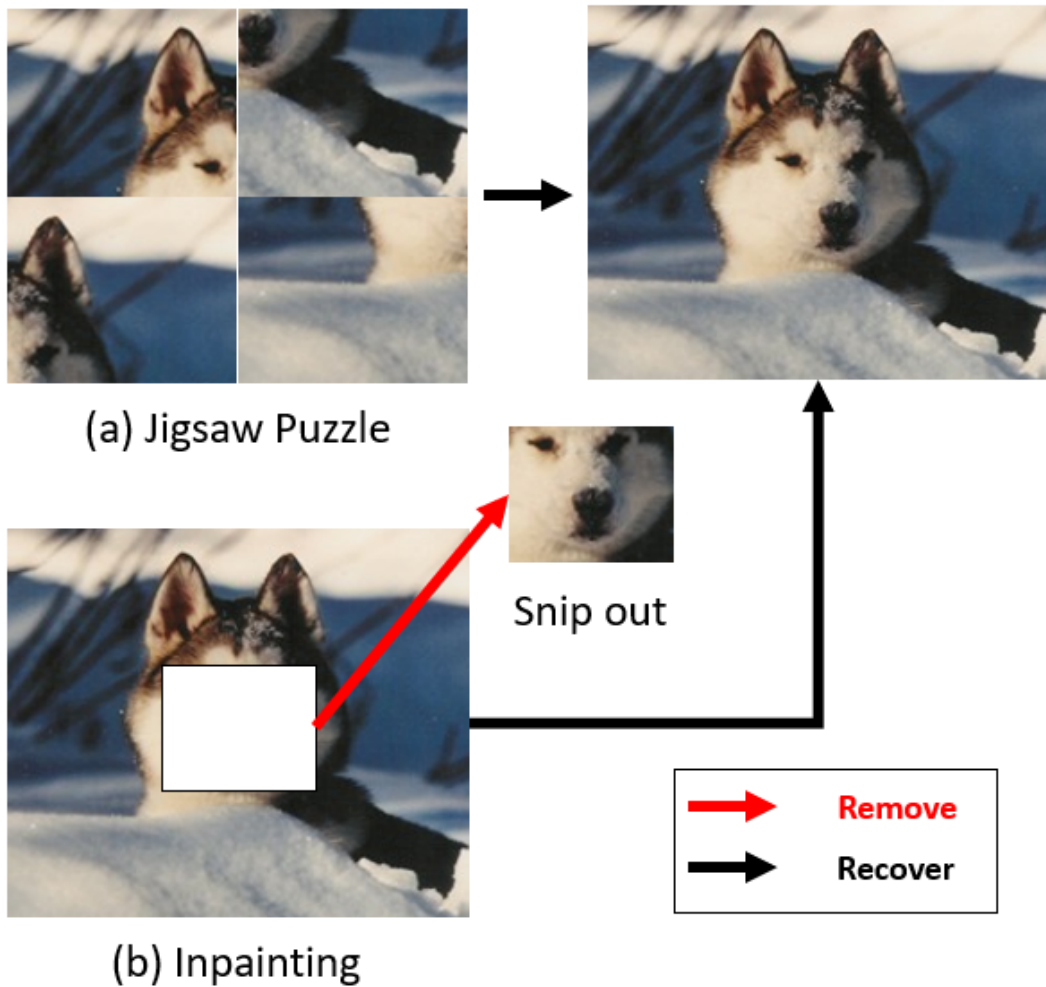


Figure 2.1: Examples of self-supervised tasks. [1] Jigsaw puzzle case, [2] occluded fragment prediction case.

One representative example of such a task is compressing the data to a more miniature representation and reconstructing them back to their original scale, using an auto-encoder. Here, supervision is provided by comparing the input frame and its reconstructed counterpart. Other tasks include the jigsaw puzzle problem

[24], where input data is fragmented and shuffled, and the model has to learn to arrange the partitions back in its original place, or the inpainting problem [25], where one part of the given data is removed, and the model should precisely predict the empty segment of data.

2.1.2 Demystification on Weakly Supervised Learning

Weakly supervised learning is a method defined under a setting where indirect and noisy labels supervise models on machine learning tasks. Unlike the direct reference that corresponds to the ideal outputs of models used in supervised learning, weak labels only provide hints to the desired outputs of models.

For instance, assume a model is assigned to segment a video clip of an athlete doing push-ups in an empty room into a region including the person and one that does not. Supervised learning will require an exact ground truth mask that covers the background and leaves the partition of each video frame that depicts the person intact. On the other hand, weakly supervised learning can be performed even if the model is given a sequence of integers, indicating how many push-ups had been done until the frame within a video clip was filmed. Undoubtedly, the integers do not directly indicate the region where the person is illustrated, but it alludes to the model that the person is in the part of each frame where certain features appear and disappear periodically.

Moreover, weakly supervised learning is distinguished from unsupervised learning, specifically self-supervised learning, because labels that require human intervention are required in its implementation. In self-supervised learning, only the data or characteristics that can be readily extracted from them are available when training models. To be specific, the method does not exploit any labels that can only be obtained via human effort and intuition.

Let us elaborate on this discrimination using the example above of detecting an athlete doing push-ups in an empty room. We observed that a practitioner that opts on applying a weakly supervised approach to the problem would attempt to count the repetition of the motion and use the numbers as labels. In contrast to this approach, unsupervised learning uses no such reference that requires human intuition and effort, such as counting. Instead, a practitioner may choose to show the model several frames of the video and then expect the model to predict the expression of the unseen frames of the same clip. In the latter case, no reference requires any form of human effort, and all self-supervision labels can be obtained solely using the information within the input data themselves.

2.2 Class Activation Maps

2.2.1 Overview on Visualizing Activations

Ever since deep learning has gained unforeseen popularity, a question arose on what neural nets see and learn and how they use such information to make decisions. This enthusiasm led to the development of various methods applied to achieve a common goal of unveiling the black-box embedded in neural networks [26, 27, 28, 29], which is better known as explainable artificial intelligence or XAI in short. One channel of research focuses on visualizing the activations triggered within convolutional neural networks (CNN) [30, 31, 29]. Methods pertinent to this field are designed to obtain activation maps, also known as attention maps, highlighting regions in image data where the neural network models pay the most attention.

To understand how activations obtained from the output of layers within CNN's can lure which part of an image contributes most to make a particular decision,

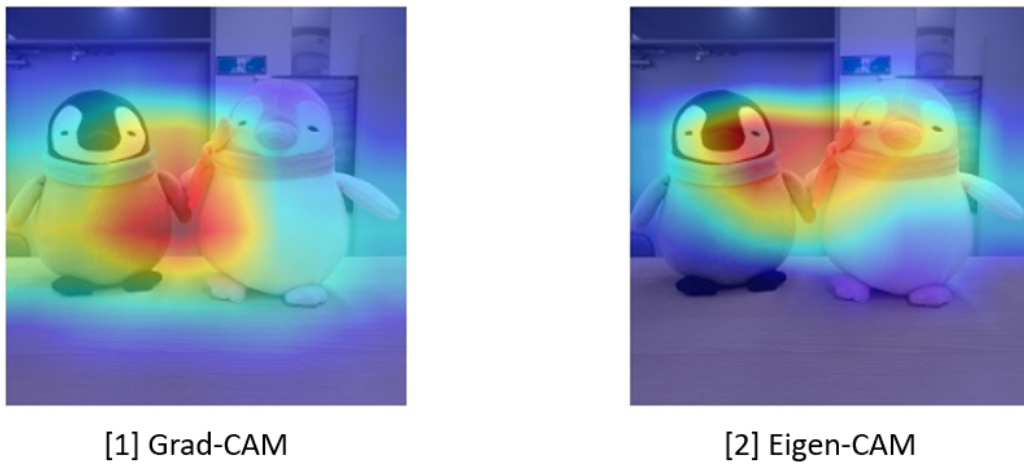


Figure 2.2: Comparison of methods used to obtain visual attention maps. [1] Grad-CAM, [2] Eigen-CAM.

we take a simple glimpse into the characteristics of convolutional neural networks (CNN). CNN's are models that consist of layers containing convolution filters. A convolution filter is a matrix or tensor-like entity that extracts prominent features within an image by simply moving them along the axes of the data and computing a linear transform between the filter and the region covered by the moving filter. This convolution outputs a compression of the data, where local regions along the data are embedded, but spatial variations are preserved.

To be specific, a filter encodes a patch of pixels as it slides over an image. After the filter traverses all regions in the image, every section is summarized into a compact representation. The encoded patches represent a region within the input image. Therefore, the information of each encoded patch corresponds to that of a particular region in the original image.

2.2.2 Overview on CAM

The earliest method introduced in the sector of XAI uses attention maps in class activation mapping (CAM) [29]. In CAM, assuming the target task is a classification task, the activation map is obtained by exploiting the output array of the final convolution layer in a deep neural network. It assumes that the last four layers of the CNN consist of 1) a convolution layer, 2) a global average pooling layer, 3) a linear layer whose length matches the number of classes, and 4) an activation layer such as soft-max. The authors compute a weighted sum on the units of the convolution layer’s output array, where the weights match that of the linear transform between the global average pooling layer and linear layer. They assert that this results in a map that highlights which regions within an image contribute most to the model’s decision.

Let us take a deeper look into the idea of CAM. Assume $f_k(x, y)$ indicates the activation result of unit k in the output of the last convolution layer, F_k , its pooled representation, and w_k^c , the weight that maps each unit to its corresponding node in the final linear layer. Here, (x, y) refers to a specific region within the input image, and c denotes a specific class. The authors point out that an attention map for class c , M_c , can be obtained by simply taking the weighted sum $\sum_k w_k^c f_k(x, y)$ over all regions lying along with the image, which may be expressed in the following form,

$$M_c = \sum_{x,y} \sum_k w_k^c f_k(x, y). \quad (2.2.1)$$

The regions emphasized for each class using activation maps obtained from CAM reasonably comply with the actual parts of images that display features pertinent to the class. However, application of CAM is limited since the method is valid for target models with an architecture similar to GoogLeNet [32], where

global average pooling is applied to the output of the last convolution layer, whose result is mapped to a linear layer followed by an activation layer that outputs the final decision of the networks.

2.2.3 Overview on Grad-CAM

Grad-CAM is a variant of CAM, applicable to a broader spectrum of models [30]. Instead of using the weights that map the activation output array of the final convolution layer in a GoogLeNet-style CNN to compute a weighted sum of activations, the method uses the gradients computed from the loss between the predicted output ground truth label for the weights. Furthermore, the activation maps are not constrained to the outputs of the final convolution layer and may be extracted from any layer in CNN's, which enables the method to be applied to more diverse types of CNN-based models.

The specific intuition behind Grad-CAM can be summarized into the following. Let us express the activation obtained from a layer within a CNN model as A and express its k 'th unit as A^k . Also, assume y_c denotes the score or objective computed for class c with respect to the ground truth. Then the gradient of the score y_c with respect to unit A^k is written in the form $\frac{\partial y_c}{\partial A^k}$. Taking an average over the spatial dimensions, to be specific, the elements along the directions of the width and height of the tensor, $\frac{1}{WH} \sum_{i=1}^W \sum_{j=1}^H \frac{\partial y_c}{\partial A_{i,j}^k}$, where W and H are the width and height of a unit of the activation, we obtain α_k^c , the weight of unit A^k for class c . The weighted sum of activation maps thresholded using a ReLU operator, $ReLU = \max(X, 0)$, where X is the input to the operator, is used to obtain the Grad-CAM,

$$L_{Grad-CAM}^c = ReLU\left(\sum_k \alpha_k^c A^k\right). \quad (2.2.2)$$

2.2.4 Overview on Eigen-CAM

Another variant of CAM named Eigen-CAM [31] obtains an attention map by applying singular value decomposition (SVD) on an output activation of layers in a CNN. Borrowing some notations from the specifications mentioned above on Grad-CAM, let us express the activation from a specific layer in the CNN model as A , where $A \in \mathbb{R}^{W \times H \times C}$ and W , H , C denote the width, height, and the number of channels of the layer output, respectively.

Unlike Grad-CAM, where gradients are computed on these components, SVD is applied to A in Eigen-CAM. To be specific, after flattening A along its spatial dimensions each with size W and H into $A_{flat} \in \mathbb{R}^{W \cdot H \times C}$ for convenience, we apply SVD to this array to get,

$$A_{flat} = U \Sigma V^T. \quad (2.2.3)$$

We then extract and express the first column of V^T , the column of the matrix that corresponds to the most significant singular value of A , as V_1^T for simplicity in notation. This vector is then multiplied to the activation A to obtain the desired attention map,

$$L_{Eigen-CAM} = AV_1^T. \quad (2.2.4)$$

One notable difference between Eigen-CAM to CAM and Grad-CAM is that the method does not require any knowledge on the output class of a classifier layer that lies after the CNN layers of a neural network. This independence to the class terms enables the method to be easily applied to various models trained with diverse tasks. It is even possible to apply Eigen-CAM to layers of neural networks trained in an unsupervised fashion, where no ground truth labels salient in supervised classification tasks are provided, making the idea more flexible for various applications than Grad-CAM.

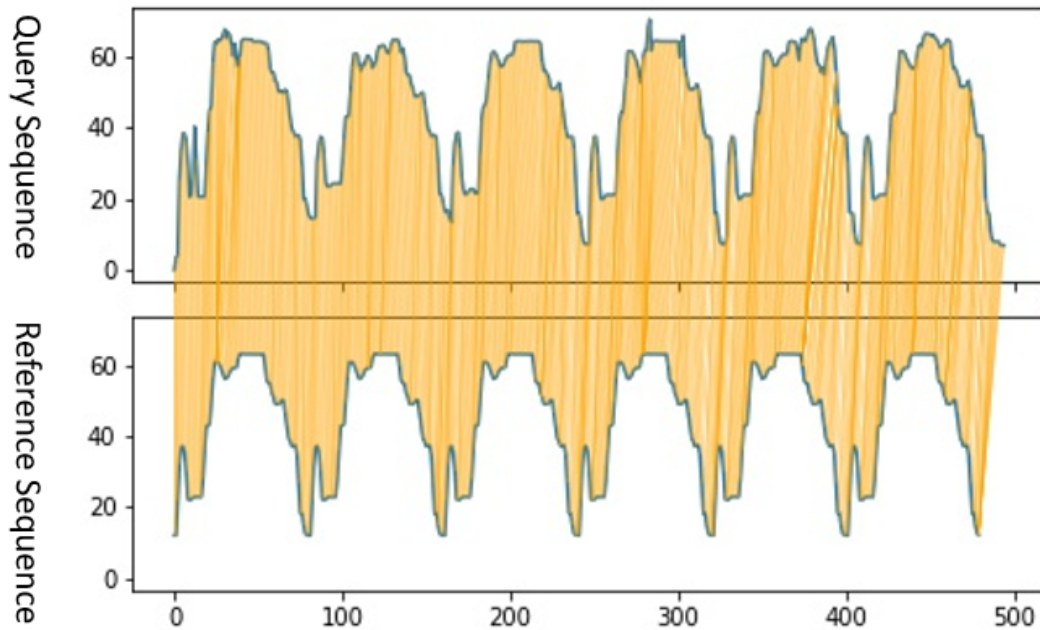


Figure 2.3: Illustration on the method implemented in Eigen-CAM.

2.3 Dynamic Time Warping

Imagine having to match points that lie along two strings of data that vary with time. Assume the two strings of data show similar features. To be specific, similar patterns in the change of values with respect to time are prominent in each stream. Human intuition easily maps one point from the first sequence to another point on the other.

However, the same task becomes problematic when developing a rule-based algorithm that determines the correspondence between the points without any human intervention. Simply comparing each pair of points using crude metrics such as the widely used euclidean or L2 norms may lead to excessive computation expenses or erroneous mapping results. Fortunately, one line of a method known as

”dynamic time warping” exists to resolve this task and is currently being used in various domains that require template matching between two different streams of sequential data.

Ever since its introduction to speech recognition [33], dynamic time warping has widely been used to correlate points that lie along with two or sequences with periods that do not coincide but change in time with a similar pattern [34]. The method is designed based on dynamic programming [35], and computes an optimal path between data under the following constraints:

- (i) The starting point of one sequence is always mapped to the starting point of the other sequence.
- (ii) The ending point of one sequence is always mapped to the ending point of the other sequence.
- (iii) Between the starting and ending points of the sequences, each index from one sequence must be mapped to at least one index from the other sequence, where the opposite should also hold.
- (iv) The mapping index from one order to another must be assigned in a monotonically increasing order.

2.4 Label Smoothing

First introduced in [36], label smoothing has been applied throughout various pieces of work on image classification, where the boundary that divides categories of data is opaque or lacks confidence. Strictly speaking, the method is applied to situations where the classifier cannot, or should not, make confident predictions. One illustrative example is when a handful of data within a dataset have been

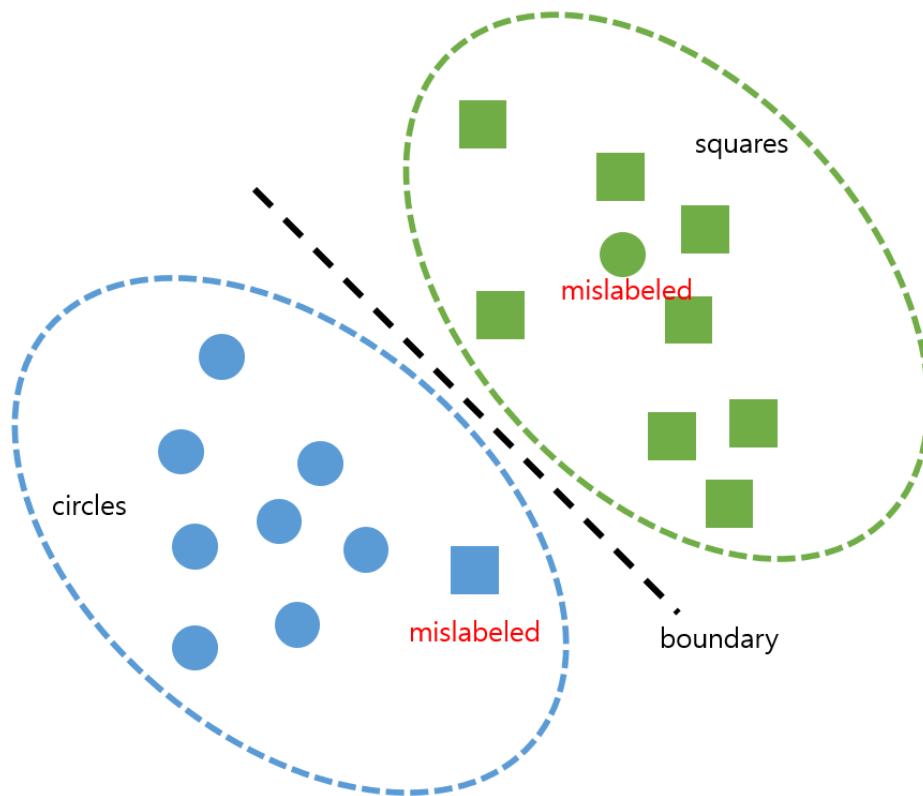


Figure 2.4: Illustration of a situation where hard, one-hot encoded labels may degrade performance when training a model.

incorrectly labeled. Under such circumstances, although most pairs that have been assigned precisely provide solid and valid references for classifiers, the erroneous instances may, on the contrary, degrade the performance of the model.

Another situation can be seen in action classification tasks with images depicting human activities. Most actions are consecutive; in other words, depictions of actions do not vary abruptly but instead show a smooth transition in appearance. Then, for example, when predicting the time-wise order of the tasks, classification on such images can be complicated. This burden is caused by frames that lie at

the boundary of different phases in a specific action, which are usually adjacent to each other in time and may show a slight discrepancy. Label smoothing mitigates these situations by feeding less confident reference labels, created by smoothing out the one-hot encoded label vectors when training classifiers. To elaborate on this opaque and concise statement about the method, let us firstly skim through the mechanism used to train classifiers, then briefly review the details of the label smoothing algorithm.

2.4.1 Review on Cross Entropy Function

To understand how to train classifiers, we first look into the objective function that drives the procedure of interest. The most common choice of objective function used to train classifiers is the cross-entropy loss function, a method usually used in information theory to compare the similarity of two distributions. Assuming p and q each denote the reference distribution, and the predicted distribution, respectively, and x denotes an element in the support for the distributions, $X \subset \mathbb{R}^C$, where C is the number of classes, cross-entropy is defined in the form

$$H(p, q) = - \sum_{x \in X} p(x) \log q(x). \quad (2.4.5)$$

It may seem awkward to apply a function intended to compare distributions, to measure the similarity between reference and predicted labels in a classification setting. However, this method is valid because classifiers' output indicates predictions on probability distributions of the actual class the input data belongs to. Specifically, each element of the output vector of a classifier indicates the probability that the input data corresponds to a particular class, which lures the distribution the data was sampled from.

2.4.2 Summary on Label Smoothing

In common practice, the reference labels are one-hot encoded. To be specific, when a vector is one-hot encoded, it is converted in a manner in which a value of one is assigned its element with the greatest value, and all other elements are substituted with a value of zero, hence the notation, "one-hot." Such labels depict a probability distribution function which resembles the Dirac delta function, expressed as $\delta_{c,y}$, whose value along the vector of length c is 1 only if the label y , matches the vector index $c \in 1, \dots, C$. Let us now denote the ground truth label for class c as $p_c = \delta_{c,y}$.

As aforementioned, however, this relatively sharp label may lead to detrimental effects when training classifiers under cases where the predictions made should be less confident. Thus, in label smoothing, the one-hot encoded ground-truth vectors are replaced with a weighted sum of a delta function $\delta_{c,y}$ and a fixed distribution $u(k)$ in the form $p' = (1 - \epsilon)\delta_{c,y} + \epsilon u(k)$. Here, the weight ϵ refers to the smoothing parameter that reduces the confidence of the largest value in the label vector while simultaneously adding uncertainty to the rest. Upon applying label smoothing, we then obtain an objective function, when using cross-entropy, of the form

$$H(p, q) = - \sum_{y \in Y} ((1 - \epsilon)\delta_{c,y} + \epsilon u(k)) \log q(y), \quad (2.4.6)$$

$$Y \subset \mathbb{R}^C. \quad (2.4.7)$$

3

Robust Framework for Industrial Video Anomaly Detection

In this chapter, we introduce our framework designed for industrial video anomaly detection. In the first section, we overview the two main components of the proposed framework, 1) the anomaly detection model and 2) the background masking model. We introduce the ideas behind each model that drive them to detect anomalies and mask out regions irrelevant to the manufacturing task of interest, respectively.

We then present some methods used to train the background masking model in the next section. We share the details of the weakly supervised task, which is designed to train the model used to create background masks. We also introduce a variant of the label smoothing method, which is required to create weak labels that better suit the characteristics of video data and the weakly supervised task used to train the masking model. Figure 3.1 depicts a full view of the components and processes of the framework.

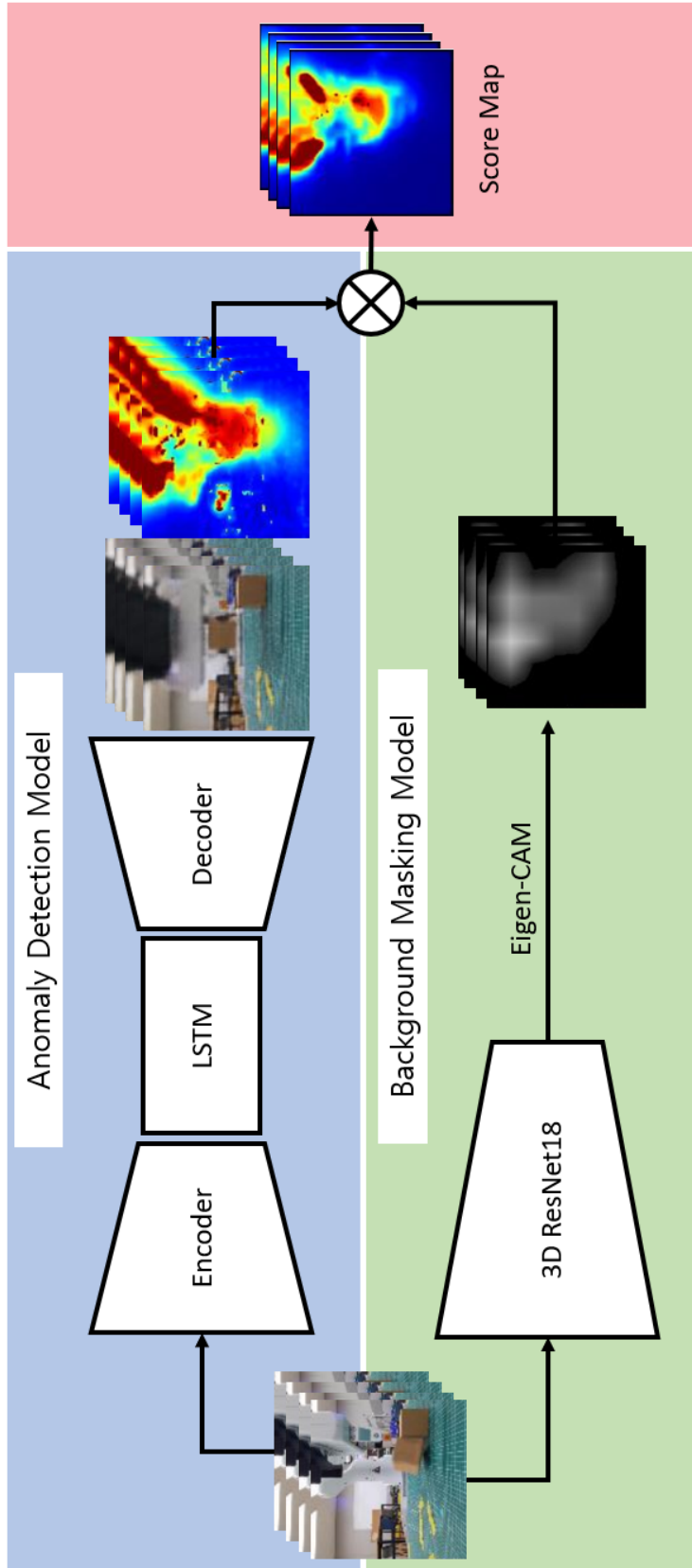


Figure 3.1: Entire grasping pipeline.

3.1 Components of the Framework

3.1.1 Anomaly Detection Model

The anomaly detection model receives consecutive frames in a video clip and detects the existence and location of an anomaly within an anomalous frame. Any existing video anomaly detection models that are trained without supervision, such as the methods using auto-encoders for surveillance tasks [14, 13, 11, 15, 12] may be adapted to this part of our framework. The reason this model should be trained in an unsupervised fashion is, as emphasized in chapter 1, because it is challenging to acquire video clips depicting anomalous events from an automated manufacturing process. For convenience, we exploited that of [11], which has a relatively simple architecture, as a baseline for our detection model.

The model is trained on clips that contain only regular events and are then used to infer whether an anomaly exists in videos not seen when training them. Here, anomalies are detected based on the inability of auto-encoders to reconstruct frames that contain contents that were not shown when training them. Using this discrepancy in the input frame and reconstruction, the model can recognize an anomaly within a frame and highlight the region's location, indicating an anomalous event.

3.1.2 Background Masking Model

The background masking model enhances the robustness to rare but irrelevant variations in video frames, making it the critical component of our framework. The model creates a mask that covers regions unrelated to the manufacturing task being inspected, such as a component of another machine operating behind the point being observed or a factory worker passing behind the machine.

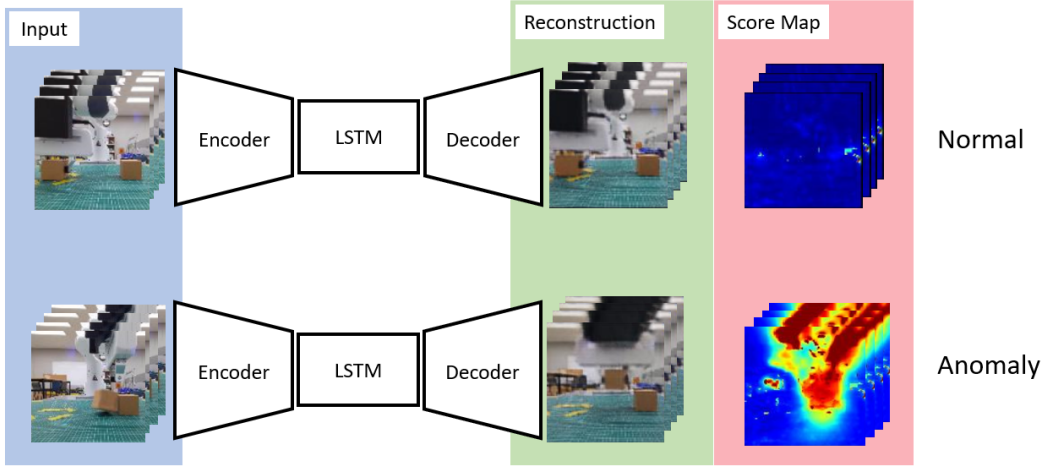


Figure 3.2: Illustration of the CLSTM auto-encoder used to detect anomalies in video frames.

The mask is generated using an attention map extracted from a CNN model trained on the very video clips used to train the anomaly detection model. The backbone of the CNN model we used in our framework is a 3D variant of the 18 layer ResNet model, R3D-18 [37]. We used a 3D CNN instead of the more common 2D CNN to make use of sequential contexts within the clips used for training, which is not supported when using 2D CNN architectures.

Among the diverse methods developed to acquire an attention map from a CNN model, we chose Eigen-CAM [31] since it does not require explicit ground truth labels, unlike Grad-CAM, and can be applied to a relatively wide range of CNN architectures and a greater variety of layers within them compared to CAM. Borrowing the notation from chapter 2, we express our attention map as $L_{Eigen-CAM,t} = A_t V_t^T$, where $t \in \{0, \dots, N_{sequence}\}$ and $N_{sequence}$ denote a single frame in a sequence fed into the model and the length of such sequence. We did

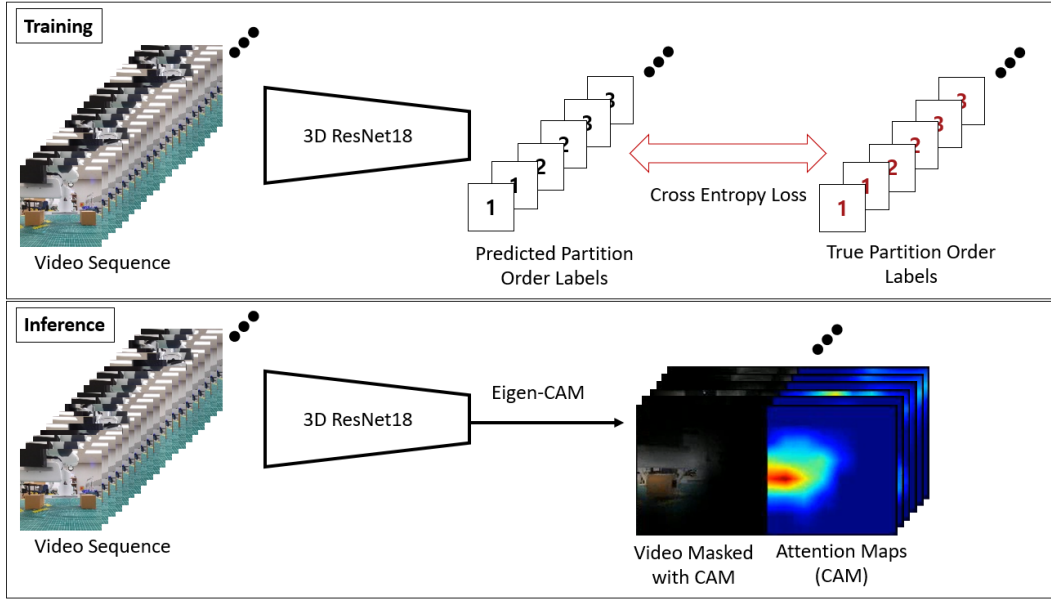


Figure 3.3: Illustration of the network that creates masks that veil regions in the frames irrelevant to the video inspection task.

not apply Eigen-CAM along the temporal dimension to obtain a map that morphs synchronous to the change in frames.

A raw Eigen-CAM attention map consists of elements with real values in the range $[0, 1]$. To effectively map out regions out of interest, we threshold the values of the attention map, where values above the threshold retain the current value, whereas those below are reduced to zeros. Also, the maps obtained are usually noisy; thus, we apply average pooling along the temporal dimension with small window size to smooth out the abrupt changes in the masks with respect to time. To sum up, the process required to obtain the background mask consists of three phases: 1) apply Eigen-CAM to the CNN backbone to extract an attention map, 2) threshold the attention map to obtain a noisy mask, and 3) smooth the mask

with respect to time.

3.1.3 Fusing Results from the Components of the Framework

Upon computing the reconstruction and background mask from each component of the framework, we now use both results to detect anomalies in the videos more robustly. To do so, we first apply the mask, through element-wise multiplication of two arrays, to the reconstruction obtained from the auto-encoder and the reference frame, which is a mere replica of the frame input to the auto-encoder. Instead of computing a pixel-wise score on the whole reference and reconstructed frames, we now compute the score only on regions that were not masked out. Thus we do not consider any failure in reconstruction that occurs out of the region of interest. By ignoring redundant regions in frames, the framework becomes robust to random and unpredictable events in video frames irrelevant to the task to be inspected at the moment.

3.2 Details of the Weakly Supervised Learning Method

3.2.1 Partition Order Prediction Task

In order to obtain masks that effectively mask out regions that contain features not pertinent to the industrial task, the model used to extract them should learn to focus on parts of the frame whose contents depict the task of interest. To do so, we focused on one preminent aspect of depicting an automated task operated by the worker, which is its periodic motion. Based on this observation, we define a task for weakly supervised learning, where the model has to predict which part of the cycle each frame in the input sequence belongs to.

To better understand the training method and how it results in a model that effectively pays attention to the automated worker, let us thoroughly examine the details of our weakly supervised training process. Firstly, we observe the video clips and determine the length of cycles that repeat within the clips. We then divide each cycle into several parts and assign integer labels to the partitions, which indicate the relative position of each partition within a single cycle. Here, all frames that belong to the same partition are assigned the same label. For example, assume that we divide a cycle into three parts and label each frame within each partition with an integer value. Then, the first partition of the cycle is assigned 1, the second, 2, and the last, 3. Let us denote such labels as partition order labels for convenience.

Then, we feed a sequence of frames into our model, which predicts the label assigned to each frame. Ground truth labels are then compared to the predictions using a cross-entropy loss, whose gradients update the model’s parameters. For this objective function to be minimized, the model should correctly predict the cycle partition order labels. Ultimately, the same model should recognize regions in videos, which depict a machine showing repetitive motion. Thus, when the model is trained sufficiently on this task, its CNN backbone learns to highlight regions in videos containing illustrations of the automated worker while not paying much attention to other regions in the same clips.

3.2.2 Partition Order Labels

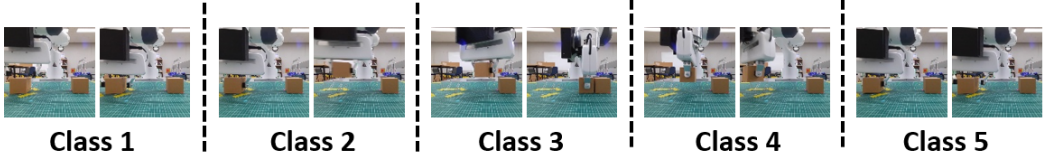
We take a deeper look into the steps and respective methods applied to obtain the partition order labels. The first step required is to determine the size of the period shown in the repetitive task. To do so, we sample $N > T_{clip}/N_{cycles}$ consecutive frames, where T_{clip} and N_{cycles} each indicate the length of a single clip sampled from the train-set and the number of apparent cycles observed in the clip. Then, pairs of frames are made, where the first element of the pair is the first frame of the clip and the second element is any frame sampled from the clip, including the first frame itself.

We then measure the L2 norm distance between pixel values of the pairs of frames and compute its mean. The outcome of this process is a list of mean values of the L2 norm distance between the first frame and all frames of a sequence, which displays a periodic pattern. We divide this sequence of scalar values into partitions of nearly equal length and similar shape and measure the length of one of these partitions to obtain the size of the period, T_{period} . In addition, we save one of the partitions T_{period} and use it as a template for the step consecutive to the current step.

Before moving on to the next step, some may wonder why such a redundant method should be used to determine the length of each cycle, instead of the more straightforward method of dividing T_{clip} by N_{cycles} . This method is not preferred because each cycle may have different lengths, which may occur due to small fluctuations in the motion of the machine or the mismatch in the frame rate of the camera and the speed of the automated worker. Under such circumstances, if the length of the cycle is obtained through $\frac{T_{clip}}{N_{cycles}}$, the clip may be divided erroneously.

Upon obtaining the partition template and its length, we copy the partition

1. Divide clips into partitions



2. Label frames

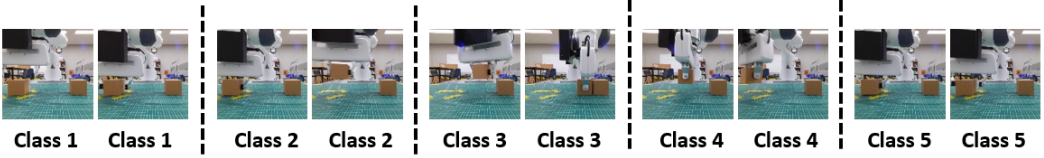


Figure 3.4: Illustration on the method used to obtain weak labels.

N'_{cycles} times to obtain a sequence consisting of N'_{cycles} repetitions of a task in which all cycles have completely equal length, where N'_{cycles} indicates the number of cycles in each clip that compose the whole dataset used for training.

Now, we iterate over all clips in the train-set and compute the pixel-wise L2 distance-vector, whose elements indicate the distance between pairs of the clip’s first frame and all frames in the clip, as it was done when obtaining the length and template of a cycle. These vectors are then mapped to points along the ideal repetition template using dynamic time warping. This way, we can safely and feasibly obtain frame-wise labels that point out the position of each frame within a cycle. Afterward, we divide the period T_{period} into C equal classes, where C indicates the number of partitions each cycle will be divided into and $c \in 1, dots, C$ indicate an index of a single partition. The outcomes of this step are the weakly supervised partition order labels required to train the background masking model. By applying one-hot encoding to our integer value labels, we finally obtain weakly

supervised labels of the form,

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_c(t) \\ \vdots \\ y_C(t) \end{bmatrix} \quad (3.2.1)$$

$$y_c(t) = \begin{cases} 1 & k = c \\ 0 & k \neq c \end{cases}, \quad k = \frac{t - \lfloor t/T_{period} \rfloor T_{period}}{T_{period}/C} \quad (3.2.2)$$

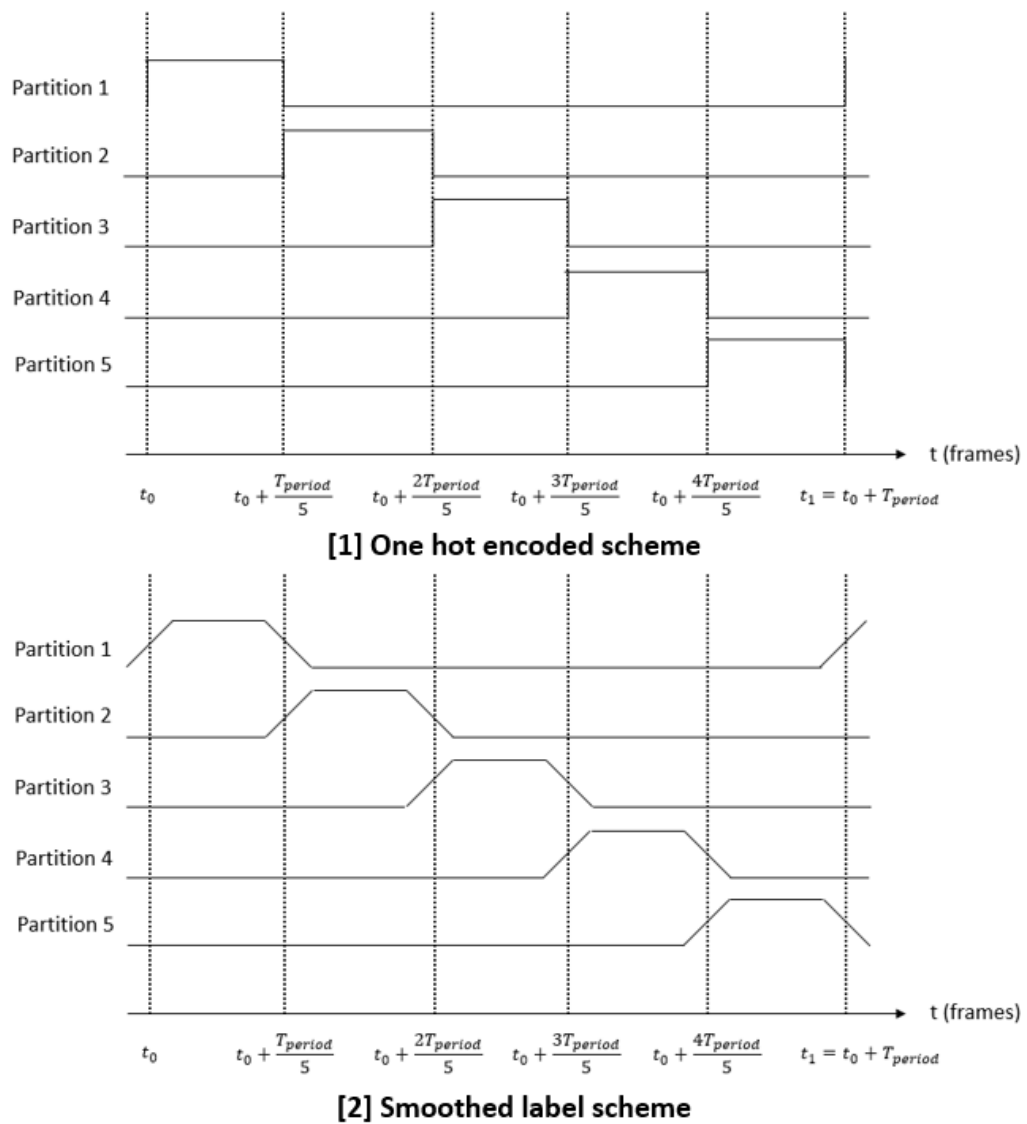


Figure 3.5: Comparison of the one-hot encoding labeling scheme and the smoothed labeling scheme.

3.2.3 Conditioning the Labels

One last problem remains yet to be resolved. Indeed, the labels obtained seem to reasonably divide an input clip into partitions that build up cycles when considering only the temporal context. However, when scrutinizing events that occur at the boundary between two partitions, one may quickly notice that adjacent frames have a very similar appearance and that the labels currently in use do not reflect this continuous change in representation. Therefore, we applied a method that smooths out the values of labels that lay at the border between two different partitions of a cycle, an idea similar to label smoothing but implemented and used differently.

Let us elaborate further on our novel smoothing method. Before applying any smoothing, each element in the one-hot encoded partition order vector is assigned according to the equation mentioned above 3.2.2. We apply a simple twist to this label assignment scheme, where we replace the unit step function-like transition from 0 to 1 and vice versa with a smoother transition. For convenience, we use a linear function resembling a slope, unlike its step-like counterpart. Assume the transition takes place for T_{trans} number of frames. Moreover, assume the boundaries that differentiate the class of interest from its neighbors is represented as b_c and b_{c+1} . Having t denote the index of a frame in a sequence of length T_{clip} , the

slope-like transition is then defined as,

$$y_c(t) = \begin{cases} 0 & t < t_c, \\ \frac{t - t_c + \lfloor \frac{T_{trans}}{2} \rfloor}{T_{trans}} & t_c - \lfloor \frac{T_{trans}}{2} \rfloor \leq t < t_c + \lfloor \frac{T_{trans}}{2} \rfloor, \\ 1 & t_c + \lfloor \frac{T_{trans}}{2} \rfloor \leq t < t_{c+1} - \lfloor \frac{T_{trans}}{2} \rfloor, \\ \frac{t_{c+1} - t - \lfloor \frac{T_{trans}}{2} \rfloor}{T_{trans}} & t_{c+1} - \lfloor \frac{T_{trans}}{2} \rfloor \leq t < t_{c+1} + \lfloor \frac{T_{trans}}{2} \rfloor, \\ 0 & t \geq t_{c+1}. \end{cases} \quad (3.2.3)$$

4

Experiments

In this chapter, we present the details of experiments done to validate the performance of the proposed industrial video anomaly detection framework. We also share the results obtained from the experiments.

In section 4.1, we introduce a database that we have gathered to evaluate the performance of our industrial video anomaly detection framework. We then introduce, in section 4.2, how we obtained masks, used as a reference to evaluate the performance of ones created from our framework and verify the effectiveness of using masks in industrial VAD. Afterward, in chapter 4.3, we measure the performance of our masking network. We do so by comparing the quality of its outcomes to that of the ideal masks. Next, we evaluate the enhancement in performance when applying our framework to industrial video anomaly detection, when compared to using only the anomaly detection model, the setting used in previous studies [11] [CLSTM-CAE]. Finally, we present some results of several ablation studies conducted to test the effects of several parameters required to configure the background masking model on the performance of our framework.

4.1 Database for Industrial Video Anomaly Detection

This paper presents a novel video database for the development and evaluation of industrial video anomaly detection frameworks. The video clips in this database were filmed on two automated tasks operated by a robot manipulator. The tasks include carrying a wooden block next to another block placed earlier and stacking a block on top of an array of blocks. For each task, we then devised four abnormal motions that might occur when conveying the tasks, such as moving the blocks to another position, dropping them on the floor, or failing to pick up the blocks in the first place.

For the clips featuring the regular operation on two tasks, we gather a total of 80 clips, with 26 clips containing some form of unexpected change in the appearance of the frame and 54 clips without any such change. Examples of unexpected change include a person doing activities behind the robot or several people gathering or discussing near the robot. Each regular clip contains portrayals for six repetitions of the task. Meanwhile, the four abnormal sets of clips that correspond to a specific task contain ten clips, with five clips containing random changes in the robot’s background and the rest without any variations. Each video clip contains two repetitions of the task, one showing a regular operation of the worker and the other showing an abnormal motion. Table 4.1 represents the number of regular frames, abnormal frames, and the total number of frames of clips filmed on each task. In the literature, however, we only share results obtained on the first task, task 0. We present the results on the other tasks in the appendix.

The clips were filmed on a camera with a three-channel RGB format, 5fps frame-rate, and 1280 x 720p resolution. We then crop out a region of interest within frames within each clip and resize the cropped section to frames with a

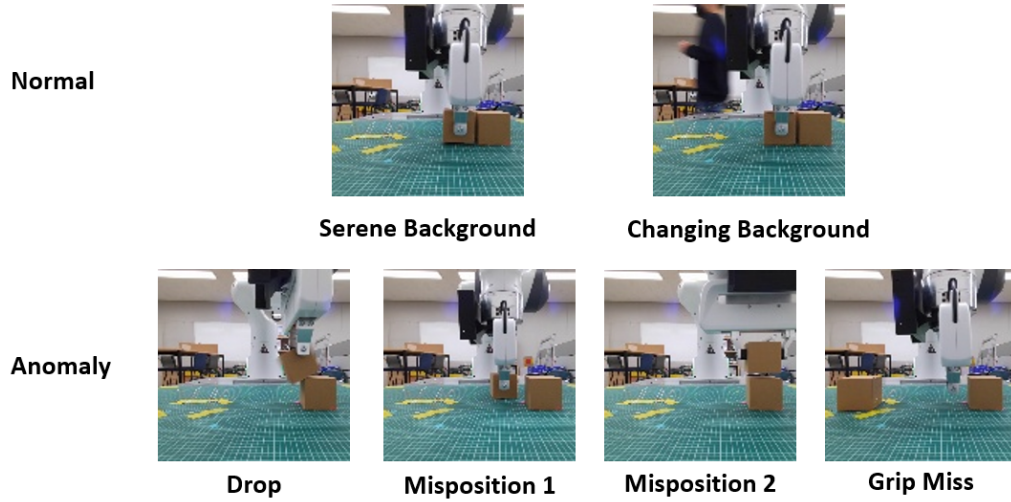


Figure 4.1: Exemplary scenes sampled from the proposed industrial video anomaly detection database.

Table 4.1: Statistics of the Database

	normal frames	abnormal frames	total frames
task 0	1896	22792	24688
task 1	1822	17762	19584

size of 128 x 128p. For convenience, we convert clips into folders with frames extracted from each clip. Each image is encoded in a JPEG format to minimize the total memory load of the database.

4.2 Ideal Background Mask

Before evaluating our model on the proposed dataset, we acquired ideal masks for our database for two primary purposes: 1) verifying the idea of applying masks to images to facilitate robust industrial video anomaly detection, and 2) using the masks as a reference to measure the effectiveness of masks obtained from the masking network of our model.

4.2.1 Acquiring Ideal Masks

In order to generate the ideal masks for the clips in our database, we first sample several clips per task, whose contents include random changes in the background and contain no abnormal motions. Then, we select all frames within the clips that display drastic changes in their appearance. On these selected frames, we designate the region within each image that shows the region of interest at the respective moment to obtain segmentation labels, repeating the process on all the chosen frames. We then train an FCN network [38] on these frames and their corresponding segmentation labels. The trained network is then fed all of the clips filmed on the same task used to create the labeled train-set, which outputs the desired ideal masks.

4.2.2 Enhancing Robustness in VAD Using Masks

Using the ideal masks, we verify whether applying masks enhances the robustness of a video anomaly detection framework to irrelevant changes that occur randomly and abruptly. As discussed in chapter 3, we apply a mask to the reconstruction output and the reference frame, a mere copy of the input fed into the auto-encoder. We replace the mask, which is obtained from the masking network

of the proposed framework, with the ideal mask. In figure 4.2 we visualize the effect when the mask is applied to target frames, and in table 4.2 we show the statistics of the results of this experiment. As shown in the column corresponding to the number of false-positive alarms, such alarms are significantly reduced when applying the ideal mask to the data and its respective reconstruction. The false-positive alarms indicate the alarms triggered by irrelevant changes in scenes. Thus, minimizing these alarms indicates enhanced robustness of the detection framework. Therefore, we conclude that the idea of applying a mask for enhanced robustness in video anomaly detection is valid.

4.3 Masking Using the Proposed Method vs Using an Ideal Mask

In this section, we evaluate the performance of the proposed framework, which uses a background masking model that generates masks from attention maps. The reference, which is to be compared to our framework, is a framework with the same anomaly detection model as ours but uses the ideal mask obtained from the methods introduced in the previous section instead of those created from the background masking model. As shown in table 4.3, although some false negatives occur when using our framework, the discrepancy in the number of false positives between the setting where we use our framework and one that uses ideal masks instead is reduced than one between a framework without any masks and one that uses the ideal masks.

4.4 Performance Enhancement Using the Proposed Method

In this section, we verify that robust video anomaly detection can be achieved by using our framework. We compare the performance of our framework to that shown when using a framework that uses no masks. As shown in table 4.4, although some false negatives occur, the number of false-positive alarms is decreased on a notable scale when using our framework. Therefore we prove that our framework, as we have intended, is robust to random events in the scenes that are not relevant to the task being inspected.

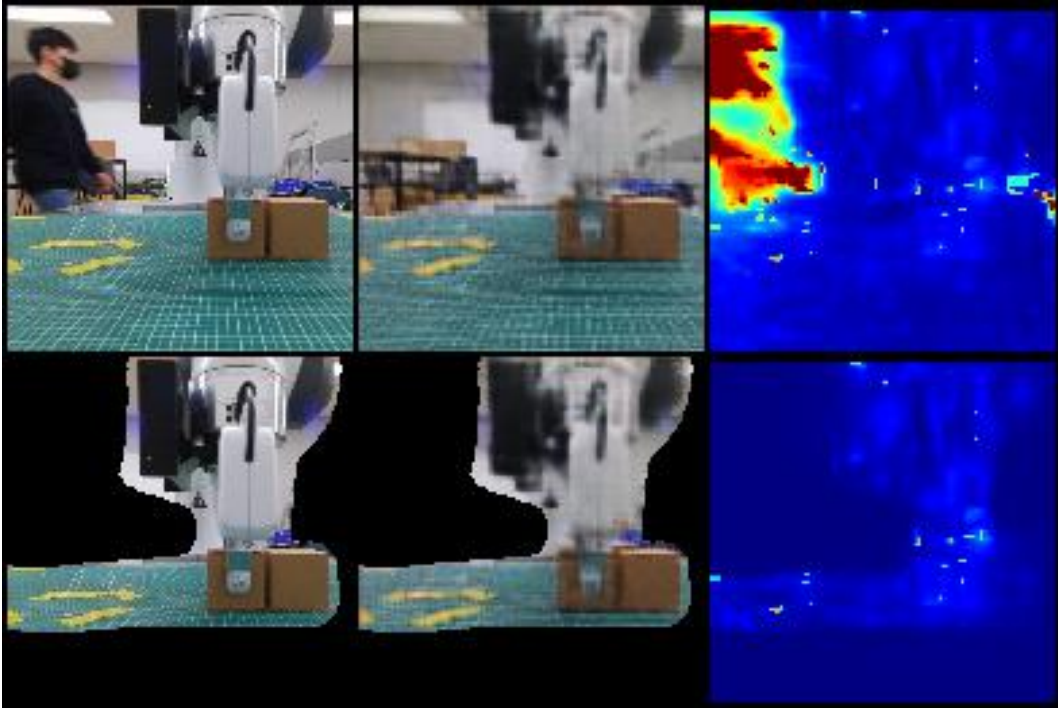


Figure 4.2: Results regarding task 0. Visualization of the effect of using an ideal mask, generated using an FCN. The segmentation labels used to train FCN were obtained by manually designating regions irrelevant to the inspected task.

Table 4.2: Results regarding task0. Results obtained using an ideal mask.

	True Positives	False Negatives	False Positives	AUROC	AUPR
No mask	1896	0	5354	0.99	0.92
Ideal mask	1896	0	1335	0.99	0.91

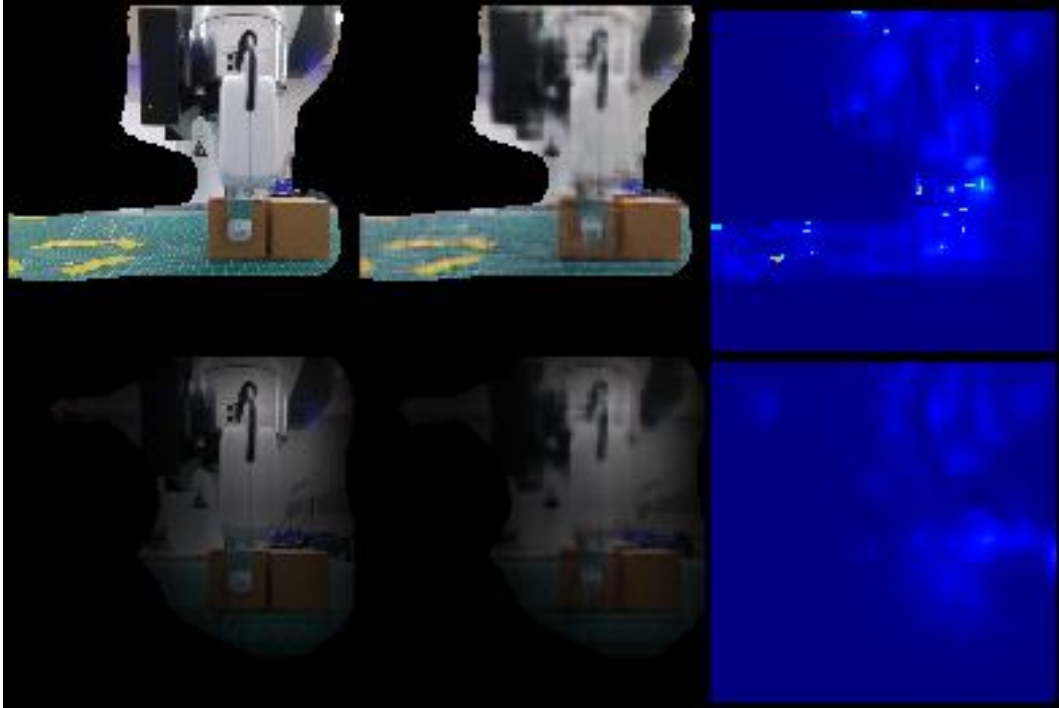


Figure 4.3: Results regarding task 0. Comparison of the visualization of ideal masks and masks obtained using the background masking model. As shown in this figure our model masks out regions similar to that of the ideal mask.

Table 4.3: Results regarding task 0. Comparison of results obtained when using an ideal mask and the mask generated from the background masking model.

	True Positives	False Negatives	False Positives	AUROC	AUPR
Ideal mask	1896	0	1335	0.99	0.91
Our mask	1835	61	2151	0.98	0.81

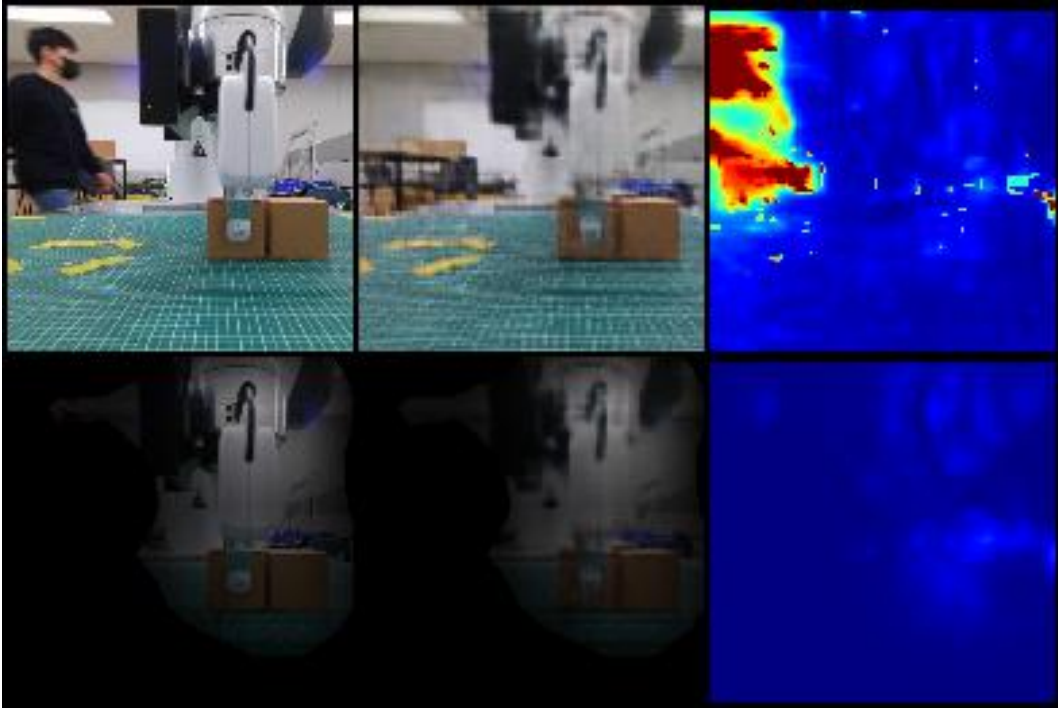


Figure 4.4: Results regarding task 0. Comparison between the qualitative results obtained when no masks were applied and when the mask obtained from the background masking model is applied.

Table 4.4: Results regarding task 0. Comparison of results obtained when using no masks and the masks generated from the background masking model.

	True Positives	False Negatives	False Positives	AUROC	AUPR
No mask	1896	0	5354	0.99	0.92
Our mask	1835	61	2151	0.98	0.81

4.5 Ablation Study

While tuning several parameters required when obtaining the background mask, we noticed that three parameters affect the quality of the mask and the overall performance of the framework, the most: the number of layers to apply Eigen-CAM to, the value of the threshold applied to the visual attention maps, and the size of the window to smooth the maps along the temporal dimension. We conducted an ablation study on these parameters and shared the results. The target data we used for these tests was the clips filmed on the first task, task 0.

4.5.1 Number of Layers for Eigen-CAM

We tested the effect of changing the number of layers used when computing Eigen-CAM. An R3D-18 network consists of 4 consecutive blocks, with each unit composed of two convolution units and one down-sampling unit. We restrict the range of layers chosen for our experiment to the two convolution units of the fourth block of the model, which results in the most global activation results. For a fair comparison, we fixed the threshold applied to the attention maps to obtain masks to 0.0 and the window size for smoothing along the temporal dimension of activation maps to 1. As shown in table 4.5, using more layers leads to more remarkable performance. Of course, more layers may be added to enhance the framework’s performance further, but this comes with a great price: the increased time required for Eigen-CAM computation.

4.5.2 Threshold on Attention Maps

As described in chapter 3, we apply a threshold to the attention maps obtained from Eigen-CAM to obtain masks that hide irrelevant regions in the video clips

Table 4.5: The effect of changing the number of layers used for Eigen-CAM.

	True Positives	False Negatives	False Positives	AUROC	AUPR
Last convolution	1781	115	2234	0.97	0.71
Last 2 convolutions	1864	32	2299	0.98	0.81
Last 4 convolutions	1881	15	2507	0.98	0.82

Table 4.6: The effect on the performance of the framework when changing the threshold applied to visual attention maps to obtain masks.

	True Positives	False Negatives	False Positives	AUROC	AUPR
Threshold=0.0	1881	15	2507	0.97	0.98
Threshold=0.1	1782	114	2012	0.97	0.71
Threshold=0.2	1473	115	2234	0.97	0.71

Table 4.7: The effect on the performance of the framework when changing the window size for smoothing applied along the temporal axis of visual attention maps to obtain masks.

	True Positives	False Negatives	False Positives	AUROC	AUPR
Window=3	1868	28	2289	0.98	0.82
Window=7	1886	10	2485	0.98	0.83
Window=11	1887	9	2658	0.99	0.85

more effectively. As shown in table 4.6, higher thresholds result in lower false positive values, where we fixed the layers referred to compute the activation maps to the last four layers of the backbone of the masking network, and the window size for smoothing along the temporal dimension of activation maps to 1. We observed that when thresholds with high values are applied to obtain the mask, the number of false negatives increases, leading to poor performance. The results indicate that a threshold with a value around 0.1 is the most appropriate option.

4.5.3 Temporal Smoothing Window Size

In our framework, we apply smoothing along the temporal dimension to attention maps to obtain masks with better quality. As shown in table 4.7, longer smoothing window lengths result in lower false negative values. However, when the lengths become excessively long, the number of false-positive alarms increases, leading to poor performance. The quality of the model is also marred since a large window size means fewer variations in the mask over time. We observed that a window size within the range [5,10] was most appropriate as a choice for the length of the window for smoothing. In this experiment, we fixed the layers considered for activation maps to the last four layers of the backbone of the masking network and the threshold applied to the attention maps to obtain masks to 0.0.

5

Conclusion

In this paper, we introduce a novel framework for robust industrial video anomaly detection. Although the problem of industrial video anomaly detection is significant, the problem has yet been dealt with. Previous works on video anomaly detection for surveillance exist, but due to the different characteristics between surveillance and industrial video inspection, the models developed on surveillance show poor performance on industrial video anomaly detection tasks. Specifically, these models are susceptible to random and diverse events within the clips that are irrelevant to inspection and raise countless false positive alarms.

To remedy this issue, we present a framework consisting of an anomaly detection model, paired with a separate model that masks out regions in the frames that do not contain any information on the inspected industrial task. To obtain the masks, we used Eigen-CAM to obtain attention maps, conditioned using a threshold, and smoothed along the axis of time to create the desired masks. The masks were then applied to the reference frames that are copies of input frames and images reconstructed by the auto-encoder. Furthermore, we gathered video

clips filmed on repetitive automated tasks that resemble real-world industrial inspection video clips to evaluate our method’s performance.

We created ideal masks crafted manually to verify whether using masks to hide regions irrelevant to the task being inspected leads to enhanced robustness in industrial video anomaly detection. We applied these masks to the input and output frames of the auto-encoder of the anomaly detection model, where we observed that the number of false-positive alarms decreases in noticeable numbers. We then applied the same ideal masks to a VAD framework and compared its performance to our framework. This experiment shows that our framework displays a similar performance to that of the framework using an ideal mask. Finally, we tested whether our framework demonstrates an increase in robustness to irrelevant variations in video frames compared to methods introduced previously and observed that it outperforms previous work regarding this aspect.

We have, however, noticed some flaws in our framework; the masking backfires and decreases the sensitivity of the framework to actual anomalies. We plan to tune our model further to reduce the number of anomalies not being appropriately noticed while minimizing the false positive alarms close to the results shown when using manually assigned ideal masks. We also plan to add more clips to the database, increase the frame rate and resolution of each clip, and accelerate the motion of the robot manipulator featured in the clips to strengthen our database’s quality. Ultimately, we expect to expand it to a public dataset, hoping this dataset will facilitate further research in the field of industrial video inspection.

A

Appendix

A.1 Experimental Results for All Tasks in the Database

This section summarizes the performance of frameworks on the three tasks within our proposed industrial video database. We compare frameworks with the same anomaly detection method but use different methods to obtain masks applied to enhance the robustness of the framework: one that uses no masks, another that uses ideal masks that were obtained manually, and the other that applies masks acquired from a background masking model proposed in our framework.

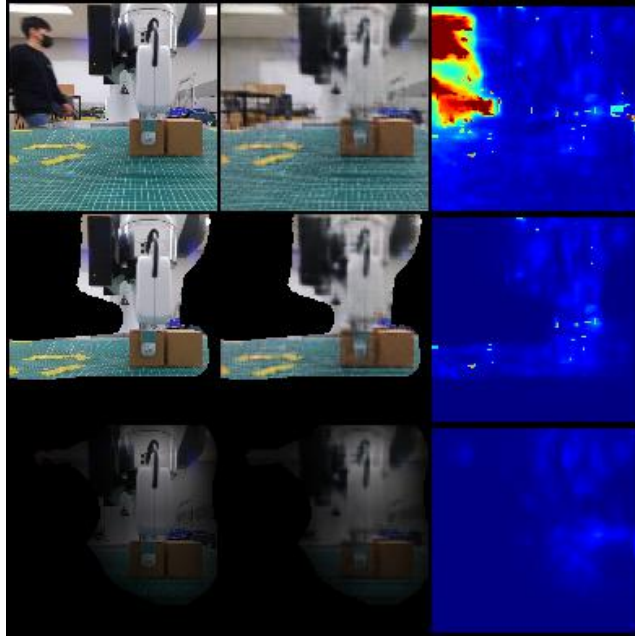


Figure A.1: Result obtained on task 0. A comparison on the appearance of the raw frames, those masked using ideal masks, and ones overlaid with masks obtained using the background masking model.

Table A.1: Results regarding task 0. Comparison of results obtained using no masks, an ideal mask, and the mask generated from the background masking model.

	True Positives	False Negatives	False Positives	AUROC	AUPR
No mask	1896	0	5354	0.99	0.92
Ideal mask	1896	0	1335	0.99	0.91
Our mask	1835	61	2151	0.98	0.81

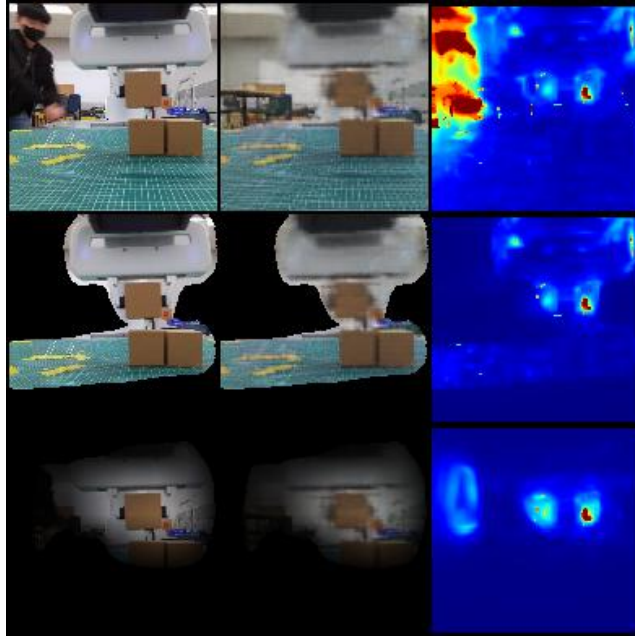


Figure A.2: Result obtained on task 1. A comparison on the appearance of the raw frames, those masked using ideal masks, and ones overlaid with masks obtained using the background masking model.

Table A.2: Results regarding task 1. Comparison of results obtained using no masks, an ideal mask, and the mask generated from the background masking model.

	True Positives	False Negatives	False Positives	AUROC	AUPR
No mask	1822	0	3970	0.98	0.88
Ideal mask	1822	0	1328	0.98	0.79
Our mask	1805	17	2079	0.98	0.86

Bibliography

- [1] Jain Darwish. A rule based approach for visual pattern inspection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(1):56–68, 1988.
- [2] Fauser Bergmann and Steger Sattlegger. Mvtec ad—a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9592–9600, 2019.
- [3] Fauser Bergmann and Steger Sattlegger. Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4183–4192, 2020.
- [4] Setkov Defard and Audigier Loesch. Padim: a patch distribution modeling framework for anomaly detection and localization. *arXiv preprint arXiv:2011.08785*, 2020.
- [5] Yoon Yi. Patch svdd: Patch-level svdd for anomaly detection and segmentation. In *Proceedings of the Asian Conference on Computer Vision*, 2020.
- [6] Qi Yang, Shi. Dfr: deep feature reconstruction for unsupervised anomaly segmentation. *arXiv preprint arXiv:2012.07122*, 2020.
- [7] Vasconcelos Li, Mahadevan. Anomaly detection and localization in crowded scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):18–32, 2014.

- [8] Shi Lu and Jia. Abnormal event detection at 150 fps in matlab. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2720–2727, 2013.
- [9] Jones Ramachandra. Street scene: A new dataset and evaluation protocol for video anomaly detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2569–2578, 2020.
- [10] Jones Ramachandra and Vatsavai. A survey of single-scene video anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [11] Liu Luo and Gao. Remembering history with convolutional lstm for anomaly detection. In *IEEE International Conference on Multimedia and Expo*, pages 439–444, 2017.
- [12] Deng Zhao, Liu Shen, and Hua Lu. Spatio-temporal autoencoder for video anomaly detection. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1933–1941, 2017.
- [13] Luo Liu and Gao Lian. Future frame prediction for anomaly detection—a new baseline. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6536–6545, 2018.
- [14] Liu Gong, Saha Le, Venkatesh Mansour, and Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1705–1714, 2019.

- [15] Noh Park and Ham. Learning memory-guided normality for anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14372–14381, 2020.
- [16] Maninis Caelles, Leal-Taixe Pont-Tuset, and Gool Cremers. One-shot video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 221–230, 2017.
- [17] Khoreva Perazzi, Schiele Benenson, and Sorkine-Hornung. Learning video object segmentation from static images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2663–2672, 2017.
- [18] Pont-Tuset Perazzi, Gool McWilliams, and Sorkine-Hornung Gross. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–732, 2016.
- [19] Song Wang, Shen Zhao, Hoi Zhao, and Ling. Learning unsupervised video object segmentation through visual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3064–3074, 2019.
- [20] Loquercio Yang and Soatto Scaramuzza. Unsupervised moving object detection via contextual information separation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 879–888, 2019.
- [21] Lamdouar Yang, Zisserman Lu, and Xie. Self-supervised video object segmentation by motion grouping. *arXiv preprint arXiv:2104.07658*, 2021.

- [22] Weissenborn Locatello, Mahendran Unterthiner, Uszkoreit Heigold, and Kipf Dosovitskiy. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020.
- [23] Frosst Sabour and Hinton. Dynamic routing between capsules. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3859—3869, 2017.
- [24] Favaro Noroozi. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, Cham, 2016.
- [25] Krahenbuhl Pathak, Darrell Donahue, and Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016.
- [26] Karen Simonyan and Zisserman Vedaldi. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [27] Han Noh, Hong. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.
- [28] Kundaje Shrikumar, Greenside. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3145–3153, 2017.
- [29] Khosla Zhou, Oliva Lapedriza, and Torralba. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the*

- IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.
- [30] Cogswell Selvaraju, Vedantam Das, and Batra Parikh. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.
- [31] Yeasin Muhammad. Eigen-cam: Class activation map using principal components. *International Joint Conference on Neural Networks*, pages 1–7, 2020.
- [32] Liu Szegedy, Sermanet Jia, Anguelov Reed, Vanhoucke Erhan, and Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
- [33] Chiba Sakoe. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.
- [34] Munshi Yadav and Alam. Dynamic time warping (dtw) algorithm in speech: a review. *International Journal of Research in Electronics and Computer Engineering*, 6(1):524–528, 2018.
- [35] Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.
- [36] Vanhoucke Szegedy, Shlens Ioffe, and Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

- [37] Wang Tran, Ray Torresani, and Paluri LeCun. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.
- [38] Shelhamer Long and Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

국문초록

공정 검사라는 다소 방대한 분야의 여러 문제 중에서, 산업용 비디오 이상 탐지는 큰 중요성을 지닌 문제이지만, 그 중요성에 비해 충분히 주목을 받지 못하고 있다. 이 문제를 연구할 때 사용할 공적인 데이터셋이 부재하며, 이를 기반으로 고안된 산업용 비디오 이상 탐지에 특화된 기법에 대한 선행 연구도 진행 된 적이 없었다. 본 논문에서는 일반적인 비디오 이상 탐지 문제와 산업용 비디오 이상 탐지 문제의 상이한 특성들을 분석하여 규명하였다. 일반적인 비디오 이상 탐지에서와 달리, 산업용 비디오 이상 탐지 문제에서는 사용 가능한 데이터의 양이 한정되어 있으며, 학습에 필요한 라벨이 없기 때문에 이를 활용한 모델을 개발하는 것이 불가능하다. 이와 같은 이유로 인해, 기존 모델을 산업용 비디오 이상 탐지 문제에 적용할 시, 검사하고자 하는 동작과 무관한 요소의 출현과 움직임으로 인한 거짓 알람이 지나치게 자주 발생한다. 분석을 기반으로, 강건한 비디오 이상 감지가 가능한 산업용 비디오 이상 탐지 방안을 고안하였다. 이 기법에서는 이상 탐지를 위한 모델과 별개로, 영상 내의 요소들 중 동작 감지와 상관 없는 것들을 가리는 모델을 활용한다. 제안하고자 하는 방안의 효용성을 검증하기 위해, 실제 공정 영상과 유사한 특성들을 보이는 로봇 동작을 촬영해 수집한 데이터베이스를 구축하였으며, 이를 활용해 모델의 성능들을 측정하였다. 본 연구에서 제시하는 강건한 비디오 이상 탐지 방안과 데이터 베이스를 논문을 통해 공개함으로써, 이 분야와 관련한 더 다양한 연구를 촉진하는데 기여 할 수 있을 것이라 기대한다.

주요어: 산업용 비디오 이상 탐지, 강건, 마스크, 어텐션 맵, 약지도학습

학번: 2019-28107