Video Latent Code Interpolation for Anomalous Behavior Detection

Valentin Durand de Gevigney, Pierre-François Marteau, Arnaud Delhay, Damien Lolive

Video Latent Code Interpolation for Anomalous Behavior Detection

1st Valentin Durand de Gevigney
Expression
IRISA - UBS
Lannion, France
0000-0002-8714-9355

2nd Pierre-François Marteau
Expression
IRISA - UBS
Vannes, France
0000-0002-3963-8795

3rd Arnaud Delhay
Expression
IRISA - Univ Rennes
Lannion, France
0000-0001-6795-7999

4th Damien Lolive
Expression
IRISA - Univ Rennes
Lannion, France
0000-0002-1110-5444

***Abstract*—Detecting an anomalous human behavior can be a challenging task. In this paper, we present a novel objective function for autoencoders which include a temporal component. Our method is a fully end-to-end semi-supervised approach for video anomaly detection. The autoencoder is trained to reconstruct a sample from a partial input, by interpolating latent codes obtained from this partial input. We show this approach improves over using usual autoencoder objective functions for video anomaly detection and achieves results close to the state of the art on a broad range of datasets. Our code is publicly available on github.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Anomaly detection in video surveillance data streams is an overwhelming task due to the ever increasing volume of data that need to be processed: this volume basically follows the number of cameras that populate massively our streets and buildings. Automatizing the detection of anomaly is thus a crucial and unavoidable need today for optimizing cost and efficiency of the control surveillance tasks. However, the characterization and identification of anomalies is a difficult problem. In general, anomalies are quite rare events, present during a short time span. They resemble sometimes to weak signal, specifically when the event characterizing the anomaly occurs in the background of the scenes that are captured by the camera.

In this article we address the automatic detection of anomalies in video through a semi-supervised learning paradigm, for which mostly anomaly free videos are used to train the model. Our contribution is threefold:

1) We propose the design of an objective function for training temporal autoencoders using past and future information. We show that this objective function improves over using usual reconstruction objective functions.
2) We review commonly used datasets, providing complete information about these datasets in a centralized manner. We also propose labels for the Subway dataset.
3) We compare multiple metrics for computing the regularity score, using multiple common evaluation measures.

This research has been financially supported by the French Ministry of Defense - Direction Générale pour l'Armement (DGA) and the University of Bretagne Sud (UBS) under the MADMAX project.

We first address the related works and position our contribution within the scope of the state of the art methods. The second section details the design of our proposed architecture, while highlighting its main novelty and justifying our choices at the light of the problems we are aiming to solve. The third section presents the experimentation we have carried out. The fourth section aims to assess our model comparatively to state-of-the-art methods that challenge our contribution in this research domain. We finally conclude our study and present some perspectives that possibly extend this work.

II. Related work

A. Video anomaly detection

Most of the time in deep learning approaches, video anomaly detection is performed within the semi-supervised learning paradigm: the training data are only made up of normal samples and the testing data are both normal and anomalous samples. Ideally, all anomalous samples are labelled and no unlabelled samples are left in the training data. Models are then trained on these normal samples, hoping they generalize well on normal samples contained in the testing data but not on anomalous samples.

Since deep learning based methods have proven to be popular and successful on anomaly detection [1], we only review major trends and leading methods entering in this subdomain.

Hasan *et al.* [1] proposed a 2D-convolutional autoencoder receiving handcrafted spatio-temporal features as inputs, namely Histograms of Oriented Gradients (HOG) and Histograms of Optical Flows (HOF) in order to learn temporal regularity in video sequences.

Hinami *et al.* [2] proposed a method that leverages the joint detection and recounting of abnormal events, the resulting models learns to classify concepts related to these abnormal events. However, this requires significantly more detailed labels, as well as labels for training data.

Fan *et al.* [3] used two 2D-convolutional variational autoencoders (VAE), one on raw input frames for spatiality and one on Optical Flow for temporality. They then performed late fusion on the scores resulting from these two streams. This allows them to treat spatial and temporal anomalies mostly separately, which can be very relevant on datasets like UCSD

Pedestrian, where cars are spatial and temporal anomalies for different reasons. The usage of a VAE also allows them to include uncertainty in the process, which is useful because a single situation has several likely outcomes.

Luo *et al.* [4] proposed a temporally-coherent sparse coding (TSC), mapped to a stacked RNN. This TSC itself uses extracted features from a fixed encoder trained on UCF101. One of the advantages of using this sparse coding is that it makes the model learn a dictionary of normal events. They also add a temporal coherence to ensure that similar features give similar sparse codes. Since the dictionary learns normal events, they yield a better regularity score than abnormal events.

B. Predicting the future

An anomalous behavior can be defined as a behavior that could not be expected from historically observed behaviors. That is to say going from a normal state into a state that could not be predicted accurately. This is why some recent works focus on predicting the next state and not only reconstructing the current one.

The method proposed by Liu *et al.* [5] uses a U-net generator to predict the next frame, based on previous frames. PSNR is then used to compute the regularity score of this frame. They train their model in an adversarial setup, also using an optical loss computed from the difference between generated and real Optical Flows evaluated on the next frames.

Zhao *et al.* [6] proposed a 3D-convolutional autoencoder with two decoders. The first one aims at reconstructing input frames, while the second one aims at predicting future frames. This method is fully end-to-end, as it only requires raw input frames to be trained. The addition of the second decoder implicitly encourages the encoder to produce latent codes yielding temporal information, which is useful for detecting temporal anomalies. For anomaly detection, the second decoder is discarded and only reconstruction error is used to compute the regularity score. This work is also proof that 3D-convolutions are suitable for video anomaly detection.

More recently, both Morais *et al.* [7] and Rodrigues *et al.* [8] proposed methods based on skeleton poses to predict the next state based on previous states. Just like [6], the method introduced by [7] aims at both reconstructing current states and predicting the next ones, this time using Gated Recurrent Units (GRU). Conversely, [8] aims at predicting next states from current ones and reconstruct current ones from next ones, consequently proposing a bi-directional architecture.

Abati *et al.* [9] defined a regularity score based on surprisal and reconstruction error. They use a 3D convolutional autoencoder and an autoregressive model. This autoregressive model receives latent codes from the encoder. The autoregressive model is trained jointly with this autoencoder, and measures surprisal, while the autoencoder provides the reconstruction error.

While the method presented here does not use an autoregressive model, it was originally designed to work with one. The initial intent was to use this autoregressive model to transform the latent code of a video, in order to predict the latent code for future frames. Since latent codes generated by unconstrained autoencoders do not necessarily capture well temporality, the presented method was made to alleviate this problem, so that the latent codes would be easy-to-transform inputs for a future autoregressive model. However, the proposed method (without the mentioned autoregressive model) shows very interesting capability to detect short temporal anomalies that we study in detail in this paper, and still shows capability to detect spatial anomalies.

This approach results in a simple 3D-convolutional autoencoder trained with a slightly different objective function. In short, our objective function makes the autoencoder act like a bi-directional autoencoder, similarly to [8].

C. End-to-end methods

We note that previous works making use of handcrafted features such as Optical Flow, HOF, HOG, skeleton poses, *etc.* [1], [3]–[5], [7], [8] usually yield better results than end-to-end approaches [2], [6], [9].

While these handcrafted features are very relevant for surveillance footage for various reasons, they do not generalize well on other types of video data, such as face cam videos. For example, optical flow tends to produce unreliable results on face cam videos, while skeleton poses cannot be applied in the same way. It can also be noted that these handcrafted features cannot be applied to other modalities such as sound and text, through which anomalous behavior can also be observed.

The usage of these handcrafted features also limits the amount of data the network has access to. While this helps by reducing the search space at first, it prevents the network to use crucial information that might have been lost in the process.

III. Method

In most contexts, detecting an anomalous human behavior comes down to detecting a temporal anomaly. In a semi-supervised framework, the autoencoder-based approaches are well suited to detect abnormal behavior, since the reconstruction error is a particularly relevant scoring function. However, traditional autoencoding approaches are not necessarily well adapted for processing temporal information specifically. The model we propose is a regular 3D-convolutional residual autoencoder trained with an objective function that encourages the encoder to yield latent codes that capture normal temporal changes observed in stream inputs.

Since [6] showed that a 3D fully convolutional autoencoder is a suitable basis, we also use it. We add residual connections to all convolutional layers (except the last one), like [9] did. Like in [2], [6], [9], [10], we design an end-to-end method, only using raw video frames as inputs. Similarly to [6], the method we describe below implicitly encourages the encoder to produce latent codes yielding temporal information.

A. Volumes

In this paper, all the datasets we experiment on only contain one modality: video. Because we aim at detecting anomalous

human behaviors, we built our method also considering multi-modal datasets, like YouTube videos or EMO&LY [11]. In multi-modal datasets, we can for example have access to voice, which can allow us to better detect anomalous behaviors.

Video, voice and text can be aligned on the temporal axis but they don't usually share the same frequency, hence why we split our inputs. Inspired by [10], we use the term *volume* to describe a section of a video. This way, all modalities can embedded and reshaped into a tensor with shape $(N, dim_{modality})$, then all modalities can be concatenated (on the last axis).

The main reason we proceed this way is to allow late fusion between modalities while keeping the size of the latent code low, which then requires less parameters, less memory and less processing power.

B. Objective function

Our method was designed to model the latent mechanisms that dynamically transform elements of our sequences over time. Since these mechanisms can be expressed as functions of time, we implicitly train our model to learn such mechanisms through a novel objective function.

Let LM be one of the latent mechanisms (*e.g.* walking), x be an input sequence (video in our case) and Δ be a random value in $[0, N-1]$. We train our model to produce x_Δ such that:

$$x_\Delta = LM(x_0, x_{N-1}, \Delta) \tag{1}$$

More precisely, the encoder e is fed with a partial input, composed of two volumes: the first t frames (x_0) and the last t frames (x_{N-1}), effectively producing two latent codes. The decoder d receives the result of a linear interpolation between these two latent codes, with a factor Δ. The autoencoder is then tasked to reconstruct x_Δ, only knowing x_0 and x_{N-1}.

Instead of using only one Δ, we can ask our model to output N volumes per training step. In this case, $N\Delta s$ are sampled linearly in $[0, N-1]$. We note that this requires significantly more processing power and memory than outputting only one volume at a time, and furthermore, it might not be suited for very large inputs.

In practice, the input x is split into N volumes of size t.

With e and d, the encoder and decoder respectively, let $e_n(x)$ be the latent code of the n^{th} volume:

$$e_n(x) = e(x_n), n \in [0, N-1] \tag{2}$$

And let $f_n(x)$ be the interpolated latent code for the n^{th} volume:

$$f_n(x) = e_{N-1}(x) \times \frac{n}{N-1} + e_0(x) \times (1 - \frac{n}{N-1}) \tag{3}$$

Then our objective function is defined by:

$$loss = \frac{1}{N} \sum_{n=0}^{N-1} ||d(f_n(x)) - x_n||^2 \tag{4}$$

Fig. 1 represents how the model is trained using this objective function. Note that since this linear interpolation occurs on the latent codes, which are the outputs of several non-linear layers, it is able to represent more complex, non-linear transformations in the input space.

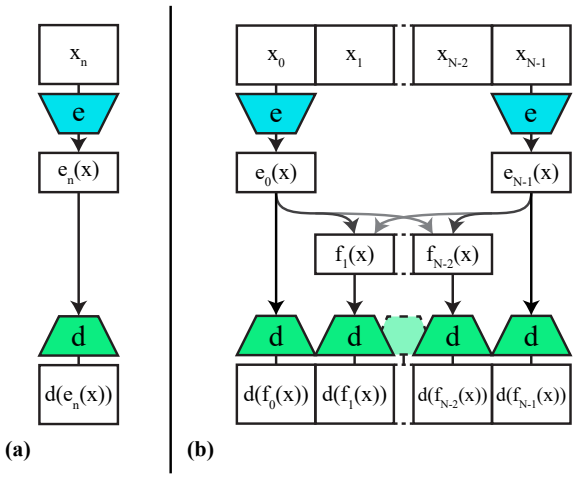

Fig. 1. (a) Our autoencoder for one volume. (b) Our autoencoder used with latent code interpolation for N volumes. Weights are shared across volumes. e and d stand respectively for the encoder and decoder parts.

C. Autoencoder

In this subsection, we detail the implementation of our autoencoder. We use 3-D convolutional residual blocks as described in [12]. In this formulation of residual blocks, a single scalar is used at the end of the residual branch to multiply it. This formulation helps reducing the exploding and vanishing gradients problem and avoid using Batch Normalization. We bring one modification to this version, where we replace single scalar biases with vector biases usually found in convolutional networks. Both the encoder and the decoder use this structure, except that convolutions are replaced by transposed convolutions in the decoder. When performing downsampling (or upsampling), a convolution (or a transposed convolution) is added to the main branch to do this. Detailed implementation of a single residual block can be visualized in Fig. 2.

While we use rectified linear unit (ReLU) as the default activation function for most of the network, we find that it is necessary to employ a bounded activation function for the latent code. For the last layer of the encoder, the last ReLU is replaced by a sigmoid function or a hyperbolic tangent function. In the decoder, we observe better performance when the last activation function is completely removed.

Training schedule. We use a warmup schedule to train our model. The learning rate linearly increases each step until reaching the target learning rate after 1 epoch. We use it to avoid early-overfitting, as datasets like Subway contain a lot of motionless parts.

Data augmentation. To prevent overfitting even further, we apply random cropping/zooming. We first randomly select the size of our window. We then randomly position this window on the frame and apply cropping. The size of the window typically ranges between 90% and 100% of the original frame.

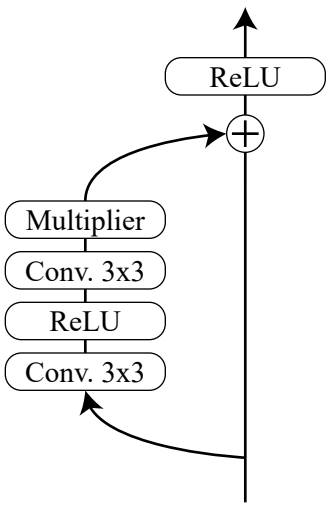

Fig. 2. Scheme of a single residual block.

IV. Experiments

We evaluate our method on six commonly used datasets for video anomaly detection, namely UCSD (Ped1 and Ped2) [13], Subway (Entrance and Exit) [14], CUHK Avenue [15] and ShanghaiTech Campus [4]. Since datasets can widely differ from each others, we adapt some hyper-parameters for training and evaluate with different setups. Hyper-parameters are listed in supplementary materials.

While most previous works perform anomaly detection at the pixel or at the frame level [3], [5], [9], we perform anomaly detection at the sequence level, where the length of each sequence is $N * t$. Since we aim to detect anomalous human behavior, we feel this is more appropriate.

In our experiments, we consider a sequence to be anomalous if the middle frame is anomalous, at the ground truth level. This approach lowered our results but we think this is more robust and less arbitrary than picking a specific percentage of frames.

A. Datasets

1) UCSD Pedestrian datasets.: The UCSD Pedestrian dataset is made of two parts: Ped1 and Ped2. In both parts, videos are taken from a stationary camera overlooking pedestrian walkways. Abnormal events are either the circulation of non-pedestrian objects (such as bikes and cars) or anomalous pedestrian motion patterns (such as walking on the grass).

UCSD Ped1 consists of 34 training videos and 36 testing videos. In addition to having frame-level labels for all 36 testing video, Ped1 also has pixel-level labels for 10 testing videos. All 70 videos have 200 frames. This dataset also contains a few corrupted frames.

UCSD Ped2 consists of 16 training videos and 12 testing videos. Ped2 has frame-level labels for all 12 testing videos. Each video has around 150 to 200 frames.

To our knowledge, the UCSD Pedestrian dataset is the most widely used dataset for video anomaly detection [1]–[6], [8]–[10], [16], [17].

2) Subway datasets.: The Subway dataset is made of two parts: Entrance and Exit. In both parts, videos are taken from a surveillance camera in a subway station. Authors provide timestamps for anomalies, defined as events that would interest a security guard such as entering the subway by the exit, or entering without payment.

Subway Entrance consists of a 1h36m long video and we use the first 18 minutes for training. These 18 minutes do not contain any anomaly. Subway Exit consists of a 43m long video and we use the first 10 minutes for training. These 10 minutes do not contain any anomaly.

This dataset does not directly contain frame-level labels, but rather timestamps at which abnormal events occur. We find that authors evaluating their models on this dataset often use different annotations as explained in [4].

Multiple works seem to use labels introduced by [18], as they all report the exact same number of events (66). We find that these labels lack accuracy, as some large sections of the video are labeled as abnormal while nobody appears on camera. We also find them to sometimes lack relevance, as these labels also include a lot of events categorized as loitering. We think considering events like loitering as abnormal to be subjective, as such consideration would not be useful if applied to the real world.

It is also worth noting that a person loitering closely to the camera can produce a significant amount of reconstruction errors, this is also true for people passing quickly under the camera. This makes these events easy to detect, and be categorized as anomalies because they would be flagged as false alarms otherwise. Unless we make an error of interpretation, we feel this is a case where the task seems to have been shaped for the method.

In our case, we manually define short windows around events like entering with no payment, or going back up the stairs. This makes our definition of anomalies very close to the definition of original authors [14]. These labels are listed in supplementary materials.

In this paper, we compare our model on these three ways of defining labels (from the original authors [14], from [18] and ours). Since original authors only provide the frame at which anomalous events occur, we define short windows around these timestamps.

3) CUHK Avenue dataset.: The CUHK Avenue dataset contains 16 training videos and 21 testing videos. Videos come from a fixed camera at human height. Anomalies include throwing objects, loitering and running. Authors provide pixel-level labels for all testing videos.

4) ShanghaiTech Campus dataset.: The ShanghaiTech Campus dataset contains 330 training videos and 107 testing videos. Videos come from 13 different surveillance cameras and covers a wide spectrum of human-related anomalies. Authors provide both pixel-level and frame-level labels for all testing videos.

B. Metrics

Usually, in previous works using autoencoders, different metrics are used to compare the input and the output of the autoencoder in order to compute the regularity score. We investigate the usage of several metrics and compare them.

The metrics we evaluate include: Mean squared error (MSE), mean absolute error (MAE), negative Structural Similarity [19] (SSIM), negative Peak Signal to Noise Ratio (PSNR). MSE and MAE are used in two different ways: with and without interpolation[1]. When not using interpolation, our model behaves like a regular autoencoder, generating latent codes for each input volume.

Cosine similarity. A special note should be made for the cosine similarity (CS), as we do not use it to compare frames directly. Instead, we compute the cosine similarity between the interpolated latent codes, and the regular latent codes. Since we are performing interpolation on the temporal axis, we are only temporal anomalies (when using cosine similarity). We find this is particularly important in the framing of anomalous human behavior detection. It is also worth noting that a parallel can be drawn with [9], where the latent code likelihood is used to determine surprisal.

C. Sliding window

To perform anomaly detection, we slide a window of size $L = N * t$, which is the total length of our input sequences. This sliding window is moved with a stride of 1. The sample evaluated by the sliding window is considered to be abnormal if the central frame of the sliding window is abnormal.

V. Results

In this section, we first compare multiple measures to compute a regularity score. Secondly, we compare our results to the same autoencoder, without our objective function for training, on the UCSD Pedestrian 2 dataset. Thirdly, we compare our best results to the state of the art. Fourthly, we compare our models on the different ways to annotate the Subway Entrance dataset.

A. Ablation study

1) Choice of metric.: As stated before, we evaluate several metrics to compute our regularity score. We compare them in this section, using the area under the following curves (AUC) : Receiver Operating Characteristic (ROC) and Precision-Recall (PR). We also compute Equal Error Rate (EER) from the ROC curve and Average Precision (AP) from the PR curve.

As it can be seen in results we present in Tables I, II and, III, all regularity scores are fairly consistent with each others. We find we cannot select a single metric for all dataset, as these six metrics all have good theoretical basis and their ranking changes depending on the dataset. Additionally, ROC, PR, EER and AP mostly point a single metric for a given dataset, suggesting the pointed metric is a robust choice for the given dataset.

[1]When using interpolation, *-i* is added after the initials.

TABLE I
METRICS COMPARISON FOR THE UCSD PEDESTRIAN DATASETS.

	Metric	ROC	EER	PR	AP
Ped 1	MSE	0.773	0.273	0.825	0.824
	MAE	0.746	0.316	0.803	0.802
	MSE-i	0.775	0.301	0.832	0.831
	MAE-i	0.753	0.312	0.813	0.813
	SSIM	0.725	0.346	0.791	0.790
	PSNR	0.788	**0.272**	0.836	0.835
	CS	**0.790**	0.290	**0.854**	**0.852**
Ped 2	MSE	0.873	0.236	0.984	0.972
	MAE	**0.933**	**0.165**	**0.993**	**0.982**
	MSE-i	0.869	0.209	0.974	0.973
	MAE-i	0.877	0.203	0.977	0.975
	SSIM	0.873	0.167	0.986	0.976
	PSNR	0.879	0.243	0.985	0.972
	CS	0.898	0.186	0.978	0.977

TABLE II
METRICS COMPARISON FOR THE SUBWAY DATASETS.

	Metric	ROC	EER	PR	AP
Entrance	MSE	0.791	0.261	0.554	0.551
	MAE	0.803	0.255	**0.582**	**0.580**
	MSE-i	0.798	0.268	0.540	0.536
	MAE-i	**0.806**	**0.245**	0.558	0.556
	SSIM	0.802	0.275	0.564	0.562
	PSNR	0.792	0.259	0.564	0.562
	CS	0.725	0.313	0.403	0.400
Exit	MSE	0.923	0.169	0.399	0.394
	MAE	**0.932**	**0.144**	0.439	0.433
	MSE-i	0.908	0.167	0.231	0.229
	MAE-i	0.919	0.164	0.318	0.317
	SSIM	0.916	0.153	**0.468**	**0.462**
	PSNR	0.924	0.168	0.402	0.394
	CS	0.919	0.161	0.277	0.268

TABLE III
METRICS COMPARISON FOR THE CUHK AVENUE AND SHANGHAITECH CAMPUS DATASETS.

	Metric	ROC	EER	PR	AP
Avenue	MSE	0.812	0.255	0.598	0.597
	MAE	0.799	0.284	0.607	0.605
	MSE-i	0.796	0.286	0.577	0.575
	MAE-i	0.777	0.310	0.566	0.564
	SSIM	0.723	0.336	0.508	0.506
	PSNR	**0.823**	**0.254**	**0.618**	**0.615**
	CS	0.755	0.318	0.517	0.514
ShanghaiTech	MSE	0.692	0.353	0.629	0.628
	MAE	0.678	0.360	0.609	0.608
	MSE-i	0.688	0.364	0.620	0.619
	MAE-i	0.674	0.367	0.596	0.595
	SSIM	**0.711**	**0.341**	**0.646**	**0.644**
	PSNR	0.703	0.349	**0.646**	**0.644**
	CS	0.688	0.365	0.617	0.601

2) Training without our objective function.: We compare our model trained with our objective function to the same model trained with the MSE objective function. All others hyper-parameters are the same, including the total number of training steps.

Results are presented in Table IV. These results show our objective function improves over using the usual MSE in our setup by a significant margin.

TABLE IV
COMPARISON OF OUR MODEL WITH AND WITHOUT OUR OBJECTIVE FUNCTION FOR THE UCSD PEDESTRIAN DATASET.

Method	ROC	EER	PR	AP
AE	0.865	0.226	0.974	0.971
IAE	**0.933**	**0.165**	**0.993**	**0.982**

TABLE V
RESULTS (ROC) ON ALL DATASETS. TOP: COMPARISON WITH END-TO-END METHODS. BOTTOM: COMPARISON WITH OTHER METHODS.

Method	Ped1	Ped2	Avenue	ST[1]	Exit	Entrance
[2]		0.922				
[6]	**0.923**	0.912	0.771			
[10]	0.899	0.874	0.803		**0.940**	**0.847**
[9]		**0.954**		**0.725**		
Ours	0.790	0.933	**0.823**	0.714	0.932	0.806
[1]	0.810	0.900	0.702		0.807	**0.943**
[5]	0.831	0.954	**0.849**	0.728		
[4]		0.922	0.817	0.680		
[20]	0.925		0.630			
[3]	0.949	0.922				
[21]	0.902				0.904	
[16]	0.957	0.884				
[17]	**0.968**	**0.955**				
[8]		0.922	0.829	**0.760**		
[7]				0.734		
Ours	0.790	0.933	0.823	0.714	0.932	0.806

[1]ShanghaiTech.

B. Comparison with previous works

In this sub-section, we compare our results with those of previous works. All reported results of previous works were directly taken from their respective papers. We separate results from end-to-end methods from others, as end-to-end methods usually do not perform as well and because using end-to-end models has advantages (as explained above).

As most previous works evaluate their methods using the area under ROC curve (auROCc), we compare our results using this metric. Since we compute several regularity scores using several metrics, we only indicate the best one for each dataset.

For the Subway Entrance dataset, we use labels from [18] to allow comparison. Results are presented in Table V.

C. Subway Entrance labels comparison

Lastly, in this sub-section, we compare the three different ways to annotate the Subway Entrance dataset we mentioned. For all three experiments, we use the same method, with the same hyper-parameters and weights. The only difference is the way the dataset is annotated.

We only report the best results for each way, which was MSE-i for the original labels and ours, and MAE-i for those introduced by Kim *et al.* [18]. Results are presented in Table VI.

D. Sliding window

While using a stride of 1 to move our sliding window is more thorough, using a bigger stride can be necessary to allow real-time detection of anomalies, depending on the hardware.

TABLE VI
COMPARISON OF THE DIFFERENT WAYS TO ANNOTATE THE SUBWAY ENTRANCE DATASET, USING OUR MODEL. ONLY THE BEST RESULTS ARE INDICATED (SEE SECTION V-C).

Labels	ROC	EER	PR	AP
Original [14]	0.855	0.227	0.140	0.140
Kim *et al.* [18]	0.806	0.245	0.558	0.556
Ours	0.834	0.230	0.204	0.203

TABLE VII
RELATIVE DIFFERENCE OF ANOMALY DETECTION RESULTS ON THE SUBWAY ENTRANCE DATASET BETWEEN DIFFERENT STRIDES (1 AND L).

Metric	Stride	ROC	EER	PR	AP
MSE	Δ	0%	0%	3%	3%
MAE	Δ	1%	3%	2%	1%
MSE-i	Δ	1%	2%	3%	2%
MAE-i	Δ	0%	0%	2%	0%
SSIM	Δ	1%	0%	0%	0%
PSNR	Δ	0%	1%	3%	4%
CS	Δ	0%	4%	0%	2%

We perform an experiment with a stride of L on the Subway Entrance dataset (L being the total length of the input).

We find that our anomaly detection results, using this bigger stride, are close to using a stride of 1, with the relative difference between metrics being between 0% and 4%, as it can be seen in table VII. Initially, to perform anomaly detection on the Subway Entrance dataset with a stride of 1, it takes about 6 hours and 32 minutes, while the duration of the dataset is about 55 minutes. When using a stride of L ($L = 32$), it takes about 11 minutes, which is well under the 55 minutes of the dataset, thus showing real-time can be achieved with our model.

VI. DISCUSSION

While we evaluate our method specifically on video, its design allows it to be applied to any type of temporal sequences. For example, by changing 3D-convolutions to 1D or 2D-convolutions, we can apply it to text (with word embeddings), sound or MFCC, for instance.

Our model is only well suited for short anomalies. As our model takes inputs of length L, increasing this size also increases requirements in memory and reduces the likelihood of convergence. This makes it impractical for longer sequences (and longer anomalies). This is why we intend to experiment with an autoregressive model, as mentioned before, building on the basis laid out in this paper. Instead of predicting the present from the near past and the near future, this autoregressive model will aim at predicting the future on longer durations, hence allowing the detection of longer anomalies.

We also argue that our model is conceptually simpler, compared with most previous works such as GAN-based models [5], [17], autoregressive models [9], and others models such as [4], [8]:

1) Our model is not in an adversarial setup. While GANs might produce more detailed output, they can also require a significant amount of time to tune hyper-parameters in order to have a stable training.

2) Our model only uses one encoder and one decoder, and the autoencoder only needs to be able to process temporal data. From there, it can take any shape or form, which makes it very adaptable. This makes our approach very versatile, as it can be applied to any type of temporal data, assuming they can be described using (1).
3) Our model will converge and yield consistent results without the usage of Batch Normalization [22], making training stable across multiple trials. In supplementary materials, we provide curves for the loss of multiple trials, using the same hyper-parameters and random initialization.

Our comparison between reconstruction metrics and latent code cosine similarity shows that the complete model also detects spatial anomalies. This is useful in datasets like UCSD Pedestrian, where cars not only move faster than pedestrians, but also look very different. However, detecting a car (or any anomalous object) and detecting an anomalous human behavior are two distinct tasks and we aim at the later one. The regularity scores obtained from reconstruction error and cosine similarity can also be merged together, as in [9] to provide a regularity score, thus exploiting surprisal and reconstruction error. We provide in supplementary materials results obtained this way.

All experiments were performed on a single GeForce RTX 2080 Ti, which has 11Go of memory. All training took between 3 hours and 12 hours. Time required to train and test our method is provided in supplementary materials. We did not observe significant changes in our results when increasing the stride used for our sliding window, this is why we include two durations for testing, as increasing the stride lowers the required time to perform anomaly detection.

VII. Conclusion

In this paper, we propose a method for video anomaly detection that relies on representing normal behaviors as functions of time. Our main contribution is the conditioning of the autoencoder, using our objective function, to produce latent codes that can be interpolated in the temporal axis.

While we do not achieve state-of-the-art, we show multiple evidences of the benefits of using our architecture and objective function:

1) Using our objective function improves over using the usual MSE objective function, as shown in Table IV.
2) We obtain comparable results to the state of the art on all datasets, demonstrating its ability to generalize well, as shown in Table V.
3) When evaluating the cosine similarity on latent codes, we observe it is a reliable metric to compute the regularity score, as shown in Tables I, II and III. While cosine similarity between latent codes does not yield better results than reconstruction errors, it shows that these representations can be directly compared.

In this paper, we explore and evaluate several metrics to compute the regularity score. Our results show that, depending on the dataset, some metrics can be clearly superior to the others, making this exploration necessary for future works.

Our results also show that choosing a metric for the regularity score is mostly straightforward on a given dataset. Our results also show that on some occasions, the ranking of those metrics will depend on the method used to rank them. Ultimately, if this happens, it will come down to the needs of the application to dictate how to rank these metrics.

We also reviewed datasets commonly used for video anomaly detection (UCSD Pedestrian 1 and 2, Avenue, ShanghaiTech, Subway Entrance and Exit). We also discussed about annotations used for Subway Entrance and provide new annotations that we feel are more appropriate for real world applications. Finally, we evaluated our method on these different annotations to provide a fair comparison.

Lastly, we showed that the stride can be increased if necessary, to provide a valuable speed-up, as the results obtained when using a stride of 1 and a stride of 32 for the sliding window are very similar (see Table VII).

VIII. Future works

The proposed approach opens up interesting leads for future works. Firstly, more work can be done on latent codes. We noticed that anomalous behaviors will often be auto-encoded to incoherent videos: for example, people going the wrong way on the Subway datasets will be decoded as going the right way (and moved instantly at the start of each volume). While we observe this incoherence from a semantic level, it often produces less reconstruction errors than more people having a normal behavior, hence the interest to find a better suited metric to detect these anomalies.

More complicated tasks are also worth investigating. For example, using different volumes as inputs, which would change the time between the start and the end of the video.

Also, latent codes produced by our method could be used in more complex models, like autoregressive models that would aim to predict the next volume, or multi-modal models using latent codes to reduce dimensionality. Finally, we think it could be worth trying to use our model in an adversarial setup, where the discriminator would help producing more detailed outputs.

References

[1] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 733–742.

[2] R. Hinami, T. Mei, and S. Satoh, "Joint detection and recounting of abnormal events by learning deep generic knowledge," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3619–3627.

[3] Y. Fan, G. Wen, D. Li, S. Qiu, and M. D. Levine, "Video anomaly detection and localization via gaussian mixture fully convolutional variational autoencoder," *arXiv preprint arXiv:1805.11223*, 2018.

[4] W. Luo, W. Liu, and S. Gao, "A revisit of sparse coding based anomaly detection in stacked rnn framework," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 341–349.

[5] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection–a new baseline," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6536–6545.

[6] Y. Zhao, B. Deng, C. Shen, Y. Liu, H. Lu, and X.-S. Hua, "Spatio-temporal autoencoder for video anomaly detection," in *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 2017, pp. 1933–1941.

[7] R. Morais, V. Le, T. Tran, B. Saha, M. Mansour, and S. Venkatesh, "Learning regularity in skeleton trajectories for anomaly detection in videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 996–12 004.

[8] R. Rodrigues, N. Bhargava, R. Velmurugan, and S. Chaudhuri, "Multi-timescale trajectory prediction for abnormal human activity detection," *arXiv preprint arXiv:1908.04321*, 2019.

[9] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara, "Latent space autoregression for novelty detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 481–490.

[10] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," in *International Symposium on Neural Networks*. Springer, 2017, pp. 189–196.

[11] C. Fayet, A. Delhay, D. Lolive, and P.-F. Marteau, "Emo&ly (emotion and anomaly): A new corpus for anomaly detection in an audiovisual stream with emotional context." in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[12] H. Zhang, Y. N. Dauphin, and T. Ma, "Fixup initialization: Residual learning without normalization," *arXiv preprint arXiv:1901.09321*, 2019.

[13] V. Mahadevan, W.-X. LI, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1975–1981.

[14] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, "Robust real-time unusual event detection using multiple fixed-location monitors," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 3, pp. 555–560, 2008.

[15] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 fps in matlab," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2720–2727.

[16] M. Ravanbakhsh, M. Nabi, H. Mousavi, E. Sangineto, and N. Sebe, "Plug-and-play cnn for crowd motion analysis: An application in abnormal event detection," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1689–1698.

[17] M. Ravanbakhsh, E. Sangineto, M. Nabi, and N. Sebe, "Training adversarial discriminators for cross-channel abnormal event detection in crowds," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1896–1904.

[18] J. Kim and K. Grauman, "Observe locally, infer globally: a space-time mrf for detecting abnormal activities with incremental updates," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 2921–2928.

[19] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[20] Y. Feng, Y. Yuan, and X. Lu, "Learning deep event models for crowd anomaly detection," *Neurocomputing*, vol. 219, pp. 548–556, 2017.

[21] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette, "Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes," *Computer Vision and Image Understanding*, vol. 172, pp. 88–97, 2018.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.