Time Series Anomaly Detection With Time and Spacial Transformations

Ariel Elnekave
M.A student HUJI
Ariel.Elnekave@mail.huji.ac.il

Abstract

In this paper I consider the problem of anomaly detection in time series. I show some extension to the method presented in [1] that work on data with time aspect. In their paper ([1]), Golan and El-Yaniv suggest training a classifier of geometric transformations applied on images and using its classification score as anomaly score once its fed with unseen image. In order to extend this idea to data with time aspect, I tried two types of transformations that suit time-series : (1) Classify geometric transformation applied on each of the time series items, (2) Classify permutations on the series items. In both cases, once the classifier was trained, I followed the above method of using the classifier as an anomaly score generator. This method showed success in some of the experiments and revealed its weaknesses in others.

1 Introduction

Anomaly detection is a very crucial task in artificial intelligence. Detecting when what you get as input is different than ever seen before can give the advantage of handling it with more precaution. While Anomaly detection in images or multivariate data is dealt with extensively, the task of detecting anomalies in time series is less common and seems harder: While anomaly in images could be just an unrecognized object appearing in them, in time series, an anomaly can be embodied in the form of a series of “normal” events that just didn’t occur in the "right" order.

In this paper, I try to extend the method described in [1], (and improved in [2]) of using a transformation classifier to detect anomalies and extend it to fit time-series. I tried to tailor data transformations that will suit data with time aspect (i.e video, or multivariate time-series) , train a classifier on their combination with some time-series data and evaluate on "anomalous" data.

Throughout this document I will refer to the entries of a time series at each time step as ’frames’ even when the series is not a video.

2 Previous work

The method presented in [1] and [2] is a case of the more general method, learning via pretext tasks or self-supervision. As well described in [4], there are lots of methods that try to learn a vital understanding of the training data by solving a "pretext task", for which annotations comes with the data and can be extracted easily. The pretext tasks are designed so that understanding the data will be crucial to solve them and so, a model trained on the self-supervised data will have a good "understanding" of the data.

The transformation classification I use in this work answers the above description quite accurately. Specifically, The permutation classification is very similar to one of the well known pretext tasks,

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

designed for image representation learning, which is the jigsaw-puzzle solving [5], reordering the pieces of a jigsaw-puzzle created from a training image.

3 Method

3.1 general framework

As described in [1] and [2], in order to detect anomaly we train a classifier over different transformations on samples from the data. Say we have N normal data samples and T transformations then a training example for the classifier will be a couple (n_i, t_j) and the task will be predicting t_j (or j) while getting only $t_j(n_i)$ as input.

Once training is done, we can start feeding the classifier with both normal and abnormal data samples and use it to differentiate them: Given a test data sample, d, we apply all the transformations on it to get $t_1(d)....t_T(d)$. Considering the training loss was a cross-entropy classification loss, we use the correct class negative log likelihood as an anomaly score,

i.e

$$S(d) = \sum_{i \in 1..T} -log\left(P(t_i|t_i(d)\right)$$

Since the classifier is trained on "normal" data, it's expected to perform worse on abnormal samples and the more the sample is abnormal the higher the score be.

3.2 The classifier

1D and 2D models are models created for the 1d/2d datasets accordingly (see part 4)

1D: convnet: A couple of 1d convolutions over the entire series. Each followed by RELU layers. I use 2 max pools with kernel of size 4 in between the convolutions. The results is fed into two FC layers that output a probability vector.

2D: Multi-headed convnet: This architecture has a 4 consecutive convolution layers , each followed by a RELU non-linearity that serves as a feature-extractor and applied on all the series frames. The extracted features are then concatenated and pushed through a 2 FC layers separated with a RELU

2D: 3dcnn: An off-the-shelve (Taken from [3]) 3dcnn architecture that is normally used for video classification.

2D: RNN: I also tried an RNN architecture but it didn't show better results than the 3dcnn

3.3 Success measures

Similarly to what's done in [2], I use TPR and FPR to measure the success of a trained model. Using the scoring method presented above, the model emits a score for each sample and a TPR and FPR can be extracted given the ground truth label and a threshold on the score. An AUC score can also be generated over a number of thresholds.

Apart from an AUC score I also report the FPR values at two fixed TPR values 1 and 0.5. I choose the matching thresholds by using minimal and median score of all the Positive ("Abnormal") examples

4 Data,Transformations and Experiments

4.1 Datasets

I created two types of synthetic datasets for my experiments, one was one dimensional and the other was 2d, i.e a synthetic video dataset.

1d continuous functions: I used a mixtures of 1d continuous functions: sinusoidal, linear and higher order polynomial, and applied them on the same 1d discrete range to get 1d ,same-size, series of their values over this range.

In order to create an anomaly sample, I added random noise to either parts of the series or all of it.

Figures 1 and 2 shows typical linear and 2nd order polynomial-sinusoid series in their normal, permutaed and anomalous modes.

Bouncing balls dataset: I created a simple video generator that simulates the physical behaviour of balls bouncing off the video frame. The videos vary in their background color and the shape color and behaviour (starting location, direction and speed) of the balls in them.

Anomaly was introduced in different ways in each experiment, starting from the balls shape going through adding discontinuous jumps in their trajectory or even adding an invisible wall they bounce off in the middle of the frame.

Figure 3 shows selected frames from a normal bouncing ball video , an affine transformed frame and an anomalous video with different shape.

Remarks: I added noise to both the videos and 1d series to make augment the data variation.

4.2 Transformations

I used 2 types of transformation that fit the time series scenario.

Random frame permutations: Assuming the data samples are all of the same length, L, I created a fix subset of all the available permutations of L frames. The transformations in this case are simply applying the permutations over the frames.

I applied this kind of transformation to both 1d and 2d data in a similar way. I added an additional parameter to this permutation, 'segment-size' , it changes the transformation to permute the series by chunks. I used 'segment-size' of 1 and 16 for 2d and 1d data accordingly

This transformations seemed apt for dealing with order related anomalies (like discontinuous movement) but less for spacial anomalies (like object shapes anomalies).

Affine transformations: The Affine transformation differ between the 1d and 2d scenarios

- 1d: Inspiring from [2] I used a LxL random matrices to multiply the 1d series.
- 2d: The affine transformation in this case are 2d image affine transformation composed of scale, sheer, shift and rotation. I created a set of such transformations, each to be applied on all of a series frames (actual video frames in this case). This should be a direct extension of the transformation in [1] to time-series and my expectation were that this could at least serve as a an average anomaly score over the frames, Series with at least one frame where an anomaly is encounter will be less easy to classify. The problem with these transformations is that it is not so sure how classifying them can help catching anomalies in the relation between frames, i.e where each frame is OK alone but the the time ordering is abnormal.

4.3 Experiments

1D:

1. 2nd order polynomial sinusoidal with affine transformations
2. 2nd order polynomial sinusoidal with permutations

In these experiments I trained a classifier over 5000 series and 100 transforms for 100 epochs

2D:

1. BB videos with different shapes as anomalies and affine transformation

2. BB videos with different shapes as anomalies and Permutations
3. BB videos with discontinuous movement as anomalies and affine
4. transformation
5. BB videos with discontinuous movement as anomalies and Permutations

In these experiments I trained a classifier over 4000 series and 50 transforms for 20 epochs

4.4 Results

1D: Both type of transformation achieves high AUC score. A significant improvement is introduced using permutation..

2D: As expected, using permutations as the target transformations worked better with discontinuous ball movement as anomaly comparing to the object shape anomalies (higher AUC and lower FPR rates in both points). This result make sense since in order to rearrange the frames in the right order a model only need to track the moving object and need not focus in its details (like shape). On the contrary, once discontinuous movement is introduced, even a human-been will find it hard to reorder the frames. This fits the general attitude of self-supervision as presented in [4]:

> "These pretext tasks must be designed in such a way that high-level image understanding is useful for solving them."

Both of the 2d experiment with affine transformation had failed to converge in training. This may be again because the synthetic videos are highly symetric and contain too few details to bare noticeable differences when an affine transformation s applied on them.

All the results are shown in Table 1.

5 Conclusions

The experiments I performed show that anomaly detection by classifying transformation can be extended to time-series data. The comparison between the two types of anomalies in the 2d scenario, object shape and behaviour over time, show that fitting the pretext tasks to different types of anomalies is crucial for the success of the method and bare the promise of finding a better pretext task for "real world" data.

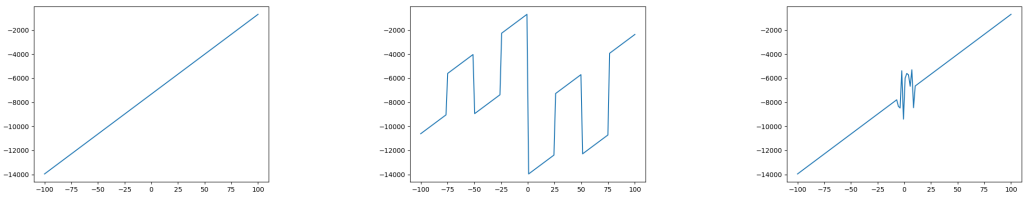

Figure 1: Linear series
Left - Normal series; Middle - permuted series; Right - Anomalous series.

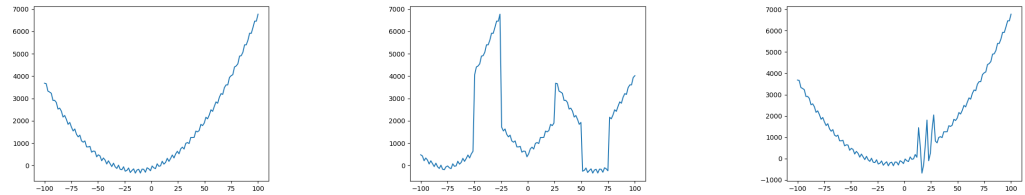

Figure 2: 2nd order polynomial sinusoid.
Left - normal serie ; Middle - permuted series; Right - Anomalous series.

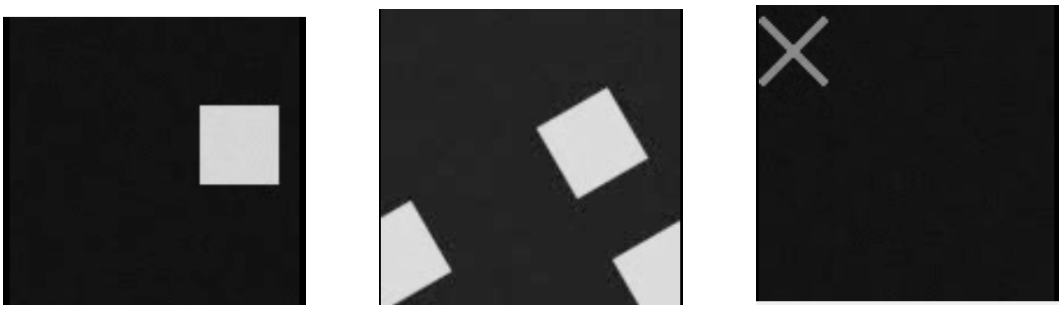

Figure 3: Frames from 2d data.
Left - normal serie ; Middle - Affine transformed series (with reflection); Right - Anomalous (different shape) series.

Table 1: Experiments result

Experiment	AUC	FPR at 1 TPR	FPR at 0.5 TPR
1d: 2nd-order-poly, Affine	0.82	0.81	0.12
1d: 2nd-order-poly, Perms	0.89	0.8	0.02
2d: Shapes, Affine	0.5	1	0.5
2d: Shapes, Permutation	0.56	1.0	0.33
2d: Discontinuous, Affine	0.5	1	0.5
2d: Discontinuous, Permutation	0.74	0.71	0.21

6 References

[1] Izhak Golan & Ran El-Yaniv. Deep anomaly detection using geometric transformations. In NeurIPS, 2018.

[2] Liron Bergman & Yedid Hoshen. Anomaly Detection by Classifying Random Transformations

[3] https://github.com/HHTseng/video-classification

[4] Alexander Kolesnikov, Xiaohua Zhai & Lucas Beye. Revisiting Self-Supervised Visual Representation Learning

[5] Mehdi Noroozi & Paolo Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles