Region Based Anomaly Detection With Real-Time Training and Analysis

P. Adey, M. Bordewich, T. Breckon
Computer Science, Durham University, UK
{philip.a.adey, toby.breckon, m.j.r.bordewich}@durham.ac.uk

O. Hamilton
Cosmonio
oliver.hamilton@cosmonio.com

***Abstract*—We present a method of anomaly detection that is capable of real-time operation on a live stream of images. The real-time performance applies to the training of the algorithm as well as subsequent analysis, and is achieved by substituting the region proposal mechanism used in [9] with one that makes the overall method more efficient. where they generate thousands of regions per image, we generate far fewer but better targeted regions. We also propose a 'convolutional' variant which does away with region extraction altogether, and propose improvements to the density estimation phase used in both variants.**

***Index Terms*—anomaly detection, machine learning, deep learning, kernel density estimation**

I. Introduction

We consider the problem of detecting abnormal or anomalous events that may occur within a video stream. Here, abnormal is defined with respect to a reference video stream in which all events are normal. Anomaly detection has potential application in the automatic surveillance of CCTV footage, in scanning for abnormalities on production lines, and in many other contexts where commonly used image classification and object detection techniques are unsuitable. These techniques aim to identify a predefined set of classes for which numerous training examples exist; whereas in anomaly detection, we aim to identify an open set of anomalous classes for which few or no training examples exist. A typical anomaly detection algorithm is designed to take only normal inputs during training and to detect deviations from this normality during analysis.

We present a method of anomaly detection that operates on a live stream of images in real-time. It is helpful to break the method down into three phases: exposure, model formation, and analysis. During the exposure phase the stream contains only normal events, which the algorithm silently observes. During the analysis phase the stream may contain anomalous events, which the algorithm aims to detect. The model formation phase is a period of time between exposure and analysis in which the algorithm forms its model of normality. The exposure and model formation phases may be collectively considered as a training phase.

We require real-time operation during exposure and analysis, with a negligible model formation period. The proposed method is able to achieve this so long as the number of normal events observed during exposure remains below an order of magnitude of 10^5. When this condition is met, the stream is processed at over two frames per second and the model is formed in approximately two seconds. These measurements were taken on a computer with a GTX 1080Ti graphics card.

Supported by Cosmonio

II. Related Work

Reconstruction based anomaly detection methods learn encoding and decoding functions to compress and reconstruct inputs. Since the functions are trained on normal inputs only, they should reconstruct anomalous inputs less accurately. Reconstruction accuracy is taken as a measure of normality. Hasan et al. [7] use a convolutional autoencoder to encode and decode stacks of input frames. Ribeiro et al. [23] take a similar approach, but only use a single frame at a time. They try to reintroduce the temporal information by fusing the input with optical flow features. Zhao et al. [28] use three-dimensional convolutions to preserve temporal information, while Chong et al. [3] preserve temporal information by performing two-dimensional convolutions over a sequence of frames before passing the sequence of resulting feature maps to convolutional Long Short-Term Memory (LSTM).

Another class of anomaly detection methods uses future frame prediction. The rationale for this approach is that prediction accuracy should be superior for normal events, since they are easier to predict. Medel et al. [20] use a convolutional LSTM architecture to predict future frames. Mathieu et al. [19] suggest new loss functions to reduce the blurriness of predicted frames, and Liu et al. [14] apply them to anomaly detection using the U-Net [24] architecture.

In sparse coding and dictionary methods, a subset of features extracted from the normal source material are used to form a dictionary of features. Test features are approximated as sparse linear combinations of the dictionary features, and those that are anomalous are expected to incur a greater approximation error. Cong et al. [4] introduce the Multiscale Histograms of Optical Flow (MHOF) algorithm as a feature extractor. Lu et al. [15] significantly speed up sparse coding at test time by learning an optimal set of dictionary elements in advance. Ren et al. [21] learn a dictionary whose elements are grouped into separate behaviours. They constrain the selection of dictionary features to use in a sparse representation by enforcing that they must all be selected from the same behaviour group. Luo et al. [16] add temporal coherency by enforcing that neighbouring frames have similar sparse coefficients.

Generative models have also been applied to anomaly detection. Schlegl et al. [25] use a Generative Adversarial Network (GAN) to learn the manifold of normal examples. Akçay et al. [1] achieve greater efficiency and performance using a novel encoder-decoder-encoder pipeline, which they later improve further by the addition of skip connections [2].

III. Method

The proposed anomaly detection pipeline is closely related to that of Hinami et al. [9] and consists of three stages: region extraction, feature extraction and density estimation. One image from the stream passes through this pipeline at a time.

The objective of the region extraction stage is to choose promising rectangular areas within the image to consider, when it is not known in advance what kinds of object to expect (Subsection III-A).

Feature extraction maps the visual data from each extracted region to a point in a feature space (Subsection III-B). Features are extracted from a layer within AlexNet [12], which has been modified following the procedure of Hinami et al. [9] to perform multi-task classification. We extract features from the fully-connected 7th layer as in Hinami et al. [9] and the convolutional 3rd layer. We refer to these as the 'fully-connected' and the 'convolutional' variants respectively.

During the exposure phase, this is the end of the path taken by an image through the pipeline. The extracted features are stored for use in the model formation phase, and the pipeline begins again with a new image. Only during the analysis phase does the pipeline continue on to the density estimation stage. Here, the probability density of each extracted feature is estimated using the normality model (Subsection III-C). A feature whose probability density is less than a certain threshold is determined to be anomalous.

A. Region Extraction

Hinami et al. [9] use Geodesic Object Proposals [10] (GOP) and Moving Object Proposals [5] (MOP) to extract approximately 2,500 regions from each frame. Both the model formation *and* the analysis phases are negatively affected when the number of features extracted during exposure is high. To cope with this, Hinami et al. [9] divide the input frames into a grid with 12 cells, and form a separate normality model for each.

In contrast, the proposed method extracts regions using a Faster-RCNN [22] model with a ResNet50 [8] backbone. This model was pretrained on the Microsoft COCO dataset [13] and obtained from the MaskRCNN-Benchmark project [18]. Since it is unknown what objects might occur, the post-processing of the RCNN regions needs to be altered to yield regions that are more scattered over the input image yet cluster preferentially around promising targets. We take the raw object detections from Faster-RCNN and discard those that are background and those whose class score is less than 0.001. We then order detections by class score before applying Non-Maximum Suppression (NMS) with an Intersection over Union (IoU) threshold of 0.5. Note that since we are uninterested in detecting individual classes, NMS is applied once on all detections rather than once per class. Typically, this method produces only tens of regions per frame, which negates the need to form numerous normality models.

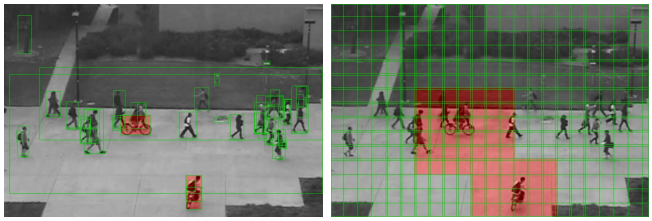

Fig. 1. Examples of regions extracted using two methods. Left: regions are extracted using a Faster-RCNN model with a ResNet50 backbone and modified post processing. Right: regions are determined by the receptive fields of units within the third convolutional layer of AlexNet

B. Feature Extraction

Hinami et al. [9] pass the extracted regions along with the input image to a Fast-RCNN [6] model that has an AlexNet [12] backbone. In one variant of their method, they adapt the architecture by removing the bounding box regression output layer and adding two new output layers: one for attribute labelling and one for action labelling. Attributes are descriptive labels such as those for describing colour and shape, while actions include verbs such as 'playing' and 'flying'. These new output layers are trained using the Visual Genome dataset [11], while the original object classification output layer is trained using the COCO dataset [13]. These three outputs receive activation from a shared fully-connected layer, which is taken as the feature representation for a region.

The fully-connected variant presented here employs exactly the same feature extraction stage, except that the feature extractor is trained using region proposals from the Region Proposal Network (RPN) of Faster-RCNN [22] rather than GOP and MOP.

The convolutional variant uses the same model but extracts features from the third convolutional layer. Specifically, one feature vector is extracted per spatial location in the layer by taking the channel activations as vector components. Consequently, the convolutional variant can dispense with the RCNN region extractor, since each feature vector is already associated with a spatial location. This method processes frames about five times faster than the fully-connected variant, since it does not need to process a separate region extraction stage. Figure 1 shows an example of the regions produced by the convolutional variant.

C. Density Estimation

In one variant of their method, Hinami et al. [9] use Kernel Density Estimation (KDE) to estimate the probability density function over the feature space. They use a Gaussian kernel with the band-width determined by Scott's Rule [26].

The fully-connected layer of AlexNet taken as the feature representation consists of 4,096 units, which is a feature

space of too high a dimensionality for KDE. To solve this, Hinami et al. [9] use Principal Component Analysis (PCA) to compress the features down to 16 dimensions. The code they provided showed that the 16-dimensional feature vectors are then normalised before being fed into KDE.

We follow the same procedure except that we report results on a range of PCA components, finding that 48 components is optimal for this dataset (Section V-A). Furthermore, instead of normalising the feature vectors, we scale them down by a constant factor so that information contained in their lengths is preserved. The factor chosen is the magnitude of the largest vector seen during exposure, which is saved so that it may be used to scale the vectors obtained during the analysis phase. Scaling rather than normalising significantly enhances the results that we obtain, bringing them more in line with those obtained by Hinami et al. [9] (Section V-B).

The third convolutional layer of AlexNet [12] has 192 channels, resulting in an uncompressed feature space with far fewer dimensions. Nevertheless, the procedure described above is still performed when using the convolutional variant.

PCA and KDE will cause an increase in the model formation period and a decrease in the analysis frame rate respectively if the number of features stored during the exposure phase is too high. If the number of features approaches 10^5, then the performance requirements set out in the introduction, are no longer satisfied.

The region extraction method of the fully-connected variant reduces the number of extracted features to an acceptable level, but the convolutional variant produces too many. Therefore, throughout the exposure phase of the convolutional variant, we only accept features that are novel relative to those already collected. To achieve this, a normality model is formed after every 25 collected features, and this model is used to measure the probability densities of subsequent features. We reject a feature if its log probability density is greater than 13. The model creation period of 25 is chosen to be small, so that it can adapt to the changing probability density function and not give too much weight to the first set of features that are unconditionally accepted. The rejection threshold of 13 is chosen based on observation of the lower end of log probability density values for normal inputs.

IV. Qualitative Analysis

The variants described in this paper are tested on the pedestrian based UCSD Ped2 dataset [17] using a training stride of five frames and a testing stride of two frames. Figure 2 shows examples of success and failure cases of the fully-connected variant at a log probability density threshold of 12, which balances the true and false positives. The method is effective at detecting anomalies based on their appearance, such as the bikes and the van; however, skateboarders are seldom detected since they appear similar to the pedestrains who are considered normal. Such anomalies would benefit from a consideration of motion, requiring the method to analyse multiple frames at a time.

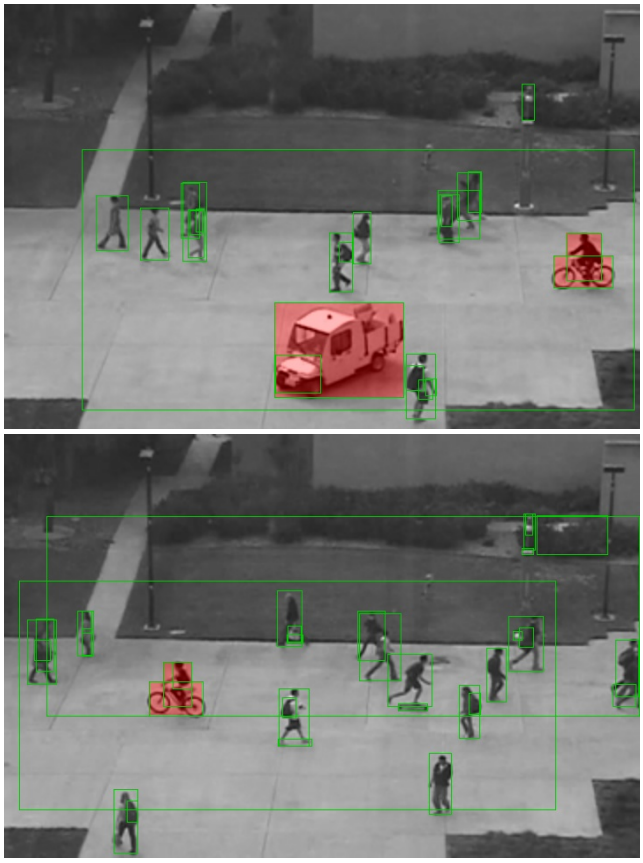

Fig. 2. Examples of anomaly detection outputs obtained from the fully-connected variant. Top: The vehicle is successfully detected. Bottom: The bike is successfully detected, but the skateboarder is missed.

The convolutional variant behaves similarly, but is less consistent, resulting in slightly worse performance as shown in the quantitative analysis in Section V.

V. Quantitative Analysis

The Area Under the ROC Curve (AUC) is used to measure the performance quantitatively. The ROC curve is constructed following the regular definitions of true-positive and false-positive for pixel-level analysis. When analysing a frame that contains at least one positive pixel, the algorithm scores a true-positive if it labels at least 40% of the positive pixels as such. When analysing a frame that contains no positive pixels, the algorithm scores a false-positive if it labels any of the pixels as positive. For reference, Hinami et al. [9] achieve an AUC of approximately 0.83 when applying KDE on the features extracted from their multi-task Fast-RCNN model.

A. PCA Components

Figure 3 compares the results obtained when using the fully-connected variant with varying numbers of PCA components. Considering that the original feature space has 4,096 dimensions, a reduction down to 16 dimensions is a significant compression and results in the lowest AUC score. Performance increases up to 48 components, but plateaus thereafter. It is

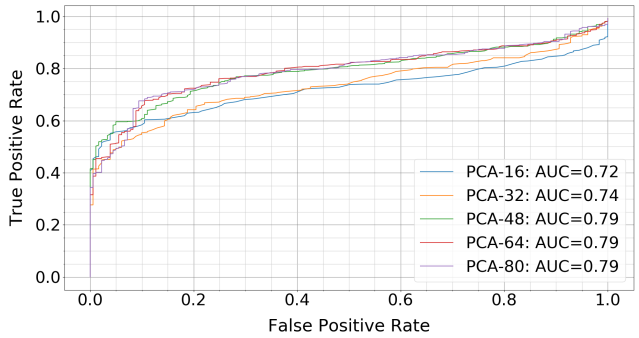

Fig. 3. ROC curves obtained from the fully-connected variant with varying number of PCA components.

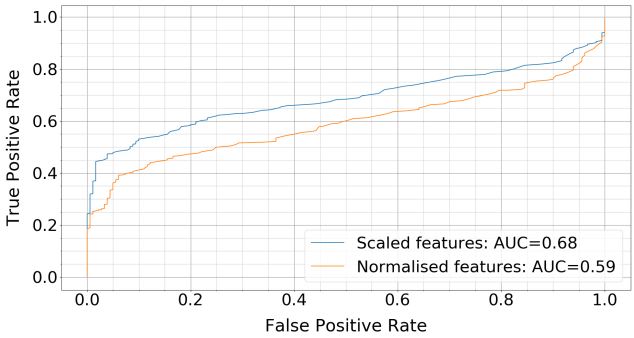

Fig. 4. ROC curves for the fully-connected variant with features scaled, and features normalised. 16 PCA components are used as in the original method of Hinami et al. [9]

reasonable to speculate that the results obtained in Hinami et al. [9] could be similarly improved via an increase in PCA components.

B. Normalising vs Scaling Features

We depart from Hinami et al. [9] in the way that we pre-process the features before density estimation. Figure 4 illustrates the difference resulting from scaling the features rather than normalising them. This simple alteration makes a significant difference in our adapted method.

C. Multi-Task AlexNet vs Original AlexNet

Figure 5 compares the results obtained when the features are instead extracted from the third convolutional layer. The multi-task model is compared with the default AlexNet model obtained from the model zoo in TorchVision [27], which is pretrained on ImageNet. The original AlexNet [12] model performs significantly better. Given this result, it would be interesting to compare with the original AlexNet model at the fully-connected layer.

VI. Conclusion

By obtaining features extracted from networks that have been trained on general recognition tasks, we are able to perform real-time training and analysis in anomaly detection tasks. Keeping networks fixed during training and analysis then avoids the computational cost of adapting network weights, and there is no need to iterate over training examples multiple times.

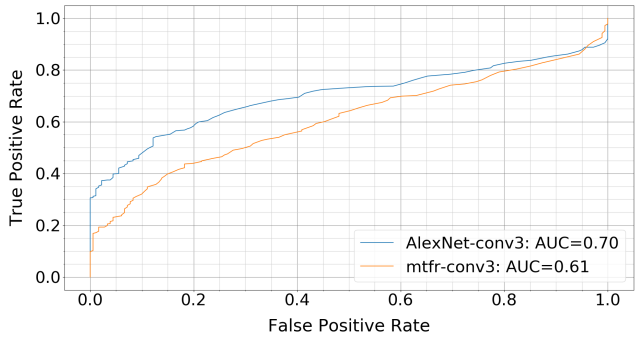

Fig. 5. A comparison of ROC curves obtained from the convolutional variant. The two curves are produced as a result of taking the third convolutional layer from the pretrained AlexNet model at TorchVision [27] and its multi-task Fast-RCNN adaptation.

We observe a significant increase in performance when the number of PCA components is raised above that used by Hinami et al. [9]. The method is also improved by scaling the collected features instead of normalising them. Given how closely related the methods are, it seems likely that the results obtained by Hinami et al. [9] could also benefit from these modifications.

References

[1] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon. Ganomaly: semi-supervised anomaly detection via adversarial training. In *Asian Conference on Computer Vision*, pages 622–637. Springer, 2018.

[2] S. Akçay, A. Atapour-Abarghouei, and T. P. Breckon. Skip-ganomaly: skip connected and adversarially trained encoder-decoder anomaly detection. *arXiv preprint arXiv:1901.08954*, 2019.

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

[4] Y. Cong, J. Yuan, and J. Liu. Sparse reconstruction cost for abnormal event detection. In *Computer Vision and Pattern Recognition*, pages 3449–3456. IEEE, 2011.

[5] K. Fragkiadaki, P. Arbelaez, P. Felsen, and J. Malik. Learning to segment moving objects in videos. In *Computer Vision and Pattern Recognition*, pages 4083–4090. IEEE, 2015.

[6] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[7] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis. Learning temporal regularity in video sequences. In *Computer Vision and Pattern Recognition*, pages 733–742. IEEE, 2016.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016.

[9] R. Hinami, T. Mei, and S. Satoh. Joint detection and recounting of abnormal events by learning deep generic knowledge. In *International Conference on Computer Vision*. IEEE, Oct. 2017.

[10] P. Krähenbühl and V. Koltun. Geodesic object proposals. In *European Conference on Computer Vision*, pages 725–739. Springer, 2014.

[11] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual genome: connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[13] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[14] W. Liu, W. Luo, D. Lian, and S. Gao. Future frame prediction for anomaly detection–a new baseline. *arXiv preprint arXiv:1712.09867*, 2017.

[15] C. Lu, J. Shi, and J. Jia. Abnormal event detection at 150 fps in matlab. In *International Conference on Computer Vision*, pages 2720–2727. IEEE, 2013.

[16] W. Luo, W. Liu, and S. Gao. A revisit of sparse coding based anomaly detection in stacked rnn framework. *International Conference on Computer Vision*, Oct. 2017.

[17] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In *Computer Vision and Pattern Recognition*, pages 1975–1981. IEEE, 2010.

[18] F. Massa and R. Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018. Accessed: 01/08/19.

[19] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[20] J. R. Medel. *Anomaly Detection Using Predictive Convolutional Long Short-Term Memory Units*. Rochester Institute of Technology, 2016.

[21] H. Ren, W. Liu, S. I. Olsen, S. Escalera, and T. B. Moeslund. Unsupervised behavior-specific dictionary learning for abnormal event detection. In *British Machine Vision Conference*, pages 28–1, 2015.

[22] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[23] M. Ribeiro, A. E. Lazzaretti, and H. S. Lopes. A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recognition Letters*, 2017.

[24] O. Ronneberger, P. Fischer, and T. Brox. U-net: convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[25] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017.

[26] D. W. Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[27] Torch contributors, torchvision. https://pytorch.org/docs/stable/torchvision/index.html, 2019. Accessed: 01/08/19.

[28] Y. Zhao, B. Deng, C. Shen, Y. Liu, H. Lu, and X.-S. Hua. Spatio-temporal autoencoder for video anomaly detection. In *Multimedia Conference*, pages 1933–1941. ACM, 2017.