Spatio-Temporal Anomaly Detection for Industrial Robots through Prediction in Unsupervised Feature Space

Asim Munawar, Phongtharin Vinayavekhin, Giovanni de Magistris

Spatio-Temporal Anomaly Detection for Industrial Robots through Prediction in Unsupervised Feature Space

Asim Munawar Phongtharin Vinayavekhin Giovanni De Magistris
IBM Research - Tokyo
{asim,pvmilk,giovadem}@jp.ibm.com

Abstract

Spatio-temporal anomaly detection by unsupervised learning have applications in a wide range of practical settings. In this paper we present a surveillance system for industrial robots using a monocular camera. We propose a new unsupervised learning method to train a deep feature extractor from unlabeled images. Without any data augmentation, the algorithm co-learns the network parameters on different pseudo-classes simultaneously to create unbiased feature representation. Combining the learned features with a prediction system, we can detect irregularities in high dimensional data feed (e.g. video of a robot performing pick and place task). The results show how the proposed approach can detect previously unseen anomalies in the robot surveillance video. Although the technique is not designed for classification, we show the use of the learned features in a more traditional classification application for CIFAR-10 dataset.

1. Introduction

Reliable working of an industrial pipeline is important to ensure the high performance of the system. As the sensors installed in the robot are not sufficient to reveal all kind of irregularities in the system, it is common for the technicians to have visual inspections of the machines and robots. This involves significant human labor and have a risk of human error and a delay in fault detection. To assist the human operator in this task, we present a video surveillance system for industrial robots and machinery using a monocular camera. The proposed system learns to perform spatio-temporal anomaly detection by using unlabeled surveillance video and warns the human operator in case of an unusual event.

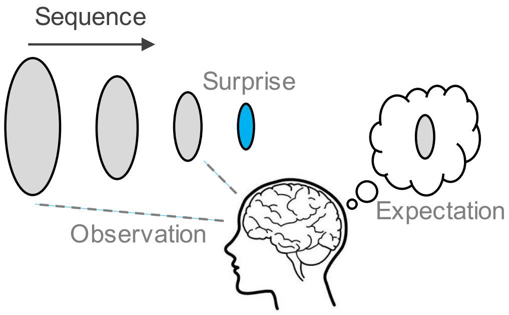

Figure 1. Expectation and surprise determine anomaly in human neural system.

The anomaly detection system we use is inspired by the prediction mechanism of human neural system. As shown in Figure 1, in human brain, anomalies are detected by comparing the expectation with the actual observation [9]. Instead of comparing the expectation and observation in image space the brain makes the comparison in an encoded feature space down the visual cortex.

In this paper, we create a feature representation of the images by training a deep convolutional neural network (CNN).Deep convolutional neural networks trained in a fully supervised manner have shown impressive results on datasets containing millions of images and thousands of classes [16]. The state-of-the-art results produced by such networks [16, 24] were only possible due to the availability of large labeled datasets. Creating these datasets is an expensive process and the amount of labeled data grows rapidly with the increasing number of model parameters. For many applications, like anomaly detection, the labeling of the data might be impossible. For these reasons, unsupervised learning - although underperforming in some applications - remains an appealing paradigm. Most of these unsupervised methods use data augmentation by exploiting both color space and geometric transforms. For our application of robotics surveillance, the data augmentation cannot be used. For example, applying geometric transformations (e.g. flip, scale, rotate etc.) have no advantage for surveillance of industrial robots from a fixed surveillance camera.

For the previous reasons, we develop a new unsupervised learning method of unlabeled data to train a deep con-

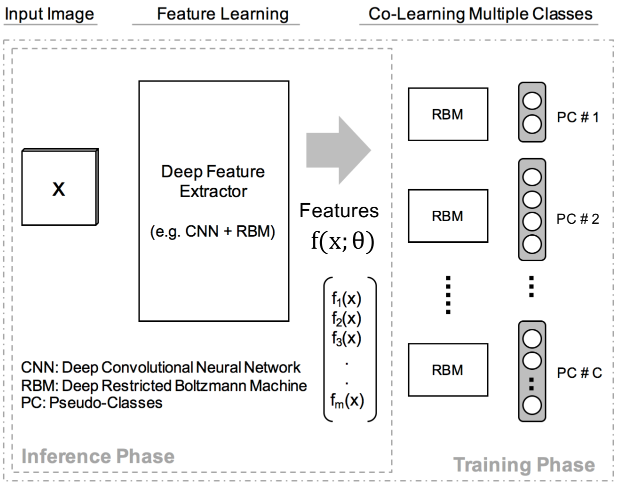

Figure 2. Co-learning on multiple pseudo-classes to learn unbiased feature representation.

volutional neural network based feature extraction system without data augmentation. The proposed method co-learns multiple classification objectives to generate unbiased feature vectors. The feature representation learned by the network is neutral and does not favor a single classification task. To detect anomalies in high dimensional input sequence (e.g. video), we train a deep prediction network in the feature domain and any event that cannot be predicted is treated as an anomaly. Although we demonstrate the proposed method on convolutional network, it is equally applicable to other kind of deep network configurations.

The rest of the paper is organized as follows: Section 2 explains the core idea of this research. Related work is described in Section 3. Details of the proposed method is described in Section 4. Quantitative analysis of the method is presented in Section 5. Finally, conclusion and future directions are given in Section 6.

2. Core Idea

The core idea behind this research is to convert the images in a sequence to a lower dimensional feature space; performing a prediction in this feature domain and comparing it with the actual observation can reveal both spatial and temporal anomalies. A good feature representation should effectively represent the input data. In supervised learning the features are learned to help perform a desired classification. As the network is trained to perform a single task the feature vectors may get biased towards that task [7]. In this paper we present a new method to generate unbiased features by unsupervised learning.

An assumption for unbiased features is that, if we can divide the input data in some clusters we should be able to divide the feature vectors into similar classes. As the data is not labeled, different image cues are used for clustering the images to generate pseudo-class labels. Instead of training on one set of pseudo-classes, we propose simultaneous training of the network on different pseudo-class labels. Figure 2 elaborates this process of co-learning. Co-learning on different labels at the same time forces the features to stay neutral and not get biased towards a single problem. The process is repeated many times with different clusters to generate more versatile features.

Figure 3 shows how the extracted features can be used to perform anomaly detection. We use a biologically plausible system for anomaly detection. According to T. Egner et al. [9], expectation and surprise determine the anomaly in human neural system. As it is not feasible to make prediction in a very high-dimensional space (e.g. videos), we use the compressed feature space to make a prediction. By predicting the next feature vector that should be observed and comparing it with the actual observation can find anomalies both in time and space.

3. Related Work

One of the main challenge in anomaly detection is that usually very few anomalous data is available. An intuitive solution is to learn to model or represent what is normal, and then detect any irregularities as anomalies. In a spatial domain, B. Saleh et al. [23] use visual attributes [10] based on the image appearance, i.e. color, texture and shape to model a normality of a particular class of object. The anomalous data is then detected by reasoning about the abnormalities using a generative model.

Anomaly detection in spatial-temporal domain becomes more complicated due to the temporal component in the data. Traditional approaches model the normalities by explicitly tracking object trajectories. Objects with unusual trajectories are identified as anomalies. A. Basharat et

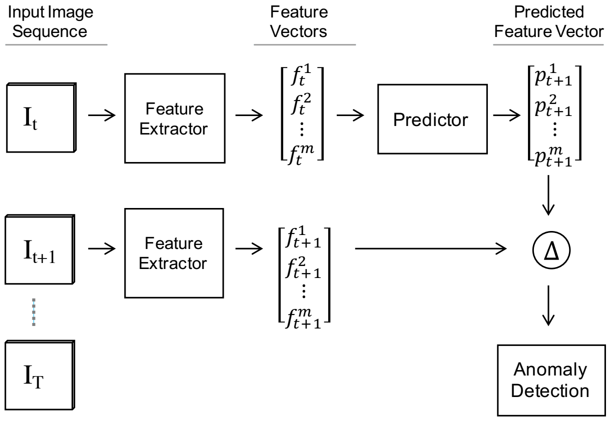

Figure 3. Anomaly detection by comparing the prediction and observation in feature domain.

al. [2] has provided an in-depth review on these approaches. W. Lawson et al. [17] define normalities of the different scenes using a dictionary of deep visual features obtained from AlexNet [16]. A context of each scene is learned and used to indicate which objects (deep visual feature clusters) should appear in the scene. Any image patches whose visual feature is matched to the cluster not associated to the current scene is detected as anomalies. D. Xu et al. [27] treated an anomaly detection as one-class classification problem. Autoencoders are used to unsurprisingly learn the feature for appearance and motion in the scene. Then a multiple one-class SVM classifiers are trained to separate between normalities and anomalies. This paper proposed a novel method to model normalities through prediction. Without explicitly detect objects or their motion, the method is capable of detecting anomalies by comparing the future prediction and the next scene appearance in the feature space.

Literature regarding an unsupervised feature learning are briefly given here as a reference to our novel feature learning method. Unsupervised feature learning has gained popularity in recent years due to the ease in data preparation. One category of an approach is based on using auto-encoders. Auto-encoder was originally considered as a dimensional reduction technique [11] similar to Principal Component Analysis (PCA). It soon became a technique to perform feature learning [21, 25] for various tasks. The main drawback is that it is hard to train the auto-encoder to learn the input distribution. A lot of research has been proposed to counter this issue [18, 20].

Another category of an approach is based on a K-mean clustering. Coates et al. [3, 4] proposed a unsupervised framework to extract features from K-mean. The proposed method is similar in to a single-layered feature mapping in a convolutional neural network. Input image is divided into a small patches and transformed into feature space using a provided encoding method based on K-mean. Dundar et al. [8] recently improved the method so that the learned filters are less redundant. They also extended the idea to allow K-mean to be trained in a deeper architecture with the intention to extract a higher-level representation.

A recent research from Dosovitskiy et al. [7] proposed a unsupervised feature learning method that is novel and does not belong the above categories. It is based on training a convolutional neural networks with a pseudo training data. Image patches are randomly sampled from input images and pseudo classes are created for each patch. To allow the networks to effectively learn discriminative features, a considerable amount of data in each class is required. This is achieved by applying a set of transformations to each patch and considering them as the data of the same class. The authors referred to the method as Exemplar-CNN.

Although Exemplar-CNN and the proposed method have some similarities, they use different techniques for creating/utilizing the pseudo-classes and training the network. Exemplar-CNN relies on data augmentation on each image to create pseudo-classes. We generate pseudo-classes based on various image cues (e.g. edge, color), without creating new image data (data augmentation). Data augmentation is suitable for learning the feature distribution for classification task, but not for the surveillance task where there are only a few variations in the appearance of the scenes. Moreover, Exemplar-CNN uses one set of pseudo-classes to train a network, while we use multiple sets of pseudo-classes to co-learn the network parameters using a single cost function.

Algorithm 1: Algorithm for creating pseudo-classes.

```
Data: $D_{train}$; Unlabeled training data
Data: $D_{test}$; Unlabeled testing data
Data: $N_{train}$; Size of training data
Data: $N_{dimension}$; Problem dimension
Input: $R$; Dimensionality reduction algorithms
Input: $C$; List of clustering algorithms
Input: $N_c$; Number of one type of clusters
Input: $N_C$; Number of clusters to generate
Result: $N_C$ labels are generated for both the training
        data and the testing data
———START———
/* Loop until $N_C$ generated */
while Total number of generated clusters $> N_C$ do
    /* Select algorithm from $R$ */
    $r$ = RandSelectFrom($R$)
    /* Compute reduced dimensional
       components for $D_{train}$ */
    $d_{train}$ = $r$.FIT(RANDINT($1$,$N_{dimension}$), $D_{train}$)
    /* Reduce $D_{test}$ dimensions */
    $d_{test}$ = $r$.TRANSFORM($D_{test}$)
    /* Select algorithm from $C$ */
    $c$ = RandSelectFrom($C$)
    while Number of generated clusters $>= N_c$ do
        /* Create clusters for $d_{train}$ */
        $C_{train}$ = $c$.FIT(RANDINT($1$,$N_{train}$), $d_{train}$)
        /* Create clusters for $d_{test}$ */
        $C_{test}$ = $c$.TRANSFORM($d_{test}$)
———END———
```

4. Details

In this section we separately discuss different parts of the proposed unsupervised feature learning and anomaly detection approach.

Algorithm 2: Proposed co-learning algorithm.

```
Data: $D_{train}$; Unlabeled training data
Data: $D_{test}$; Unlabeled testing data
Input: $N_C$; Total number of clusters (pseudo-classes) available for training
Result: Trained deep feature extractor that is not biased towards a single kind of problem
———START———
/* Create and initialize a deep feature extractor (FE) with random weights.   */
$F_{network}$=Init()
/* Loop until the overall termination criteria is satisfied.                 */
while Not Satisfying the Overall Termination Criteria do
    /* Randomly select some clusters from $N_C$ set of available pseudo-labels  */
    $N$=RANDINT(1,$N_C$)
    $C_{train}, C_{test}$ = RANDOMLYSELECTNCLUSTERS($N$)
    /* Join feature extractor with $N$ RBMs each terminating with a softmax    */
    $T_{network}$=JOIN($F_{network}$, $N$ RBM)
    while Not Satisfying the Termination Criteria do
        /* Update the network by sum of softmax cross entropy as cost function  */
        UPDATENETWORK($T_{network}$)
———END———
```

4.1. Unsupervised Feature Learning

The proposed unsupervised algorithm for training feature extractor depends on generating a set of pseudo-labels for training the network. Algorithm 1 elaborates the process of creating pseudo-labels. These pseudo-labels, when co-learned simultaneously, generate feature representation that is not biased towards single classification task.

4.1.1 Generating Pseudo-Classes

Pseudo-classes can be generated by clustering algorithms. In this paper, we have used commonly used k-means clustering algorithm [19] with randomly initialized centroids. For a set of data points $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, where each observation has d-dimensions, k-means clustering algorithm aims to partition the n observations into $k(\leq n)$ sets $\mathbf{S} = S_1, S_2, \ldots, S_k$. It does so by minimizing the sum of squares within each cluster. The objective of the k-means can be written as: where, $\boldsymbol{\mu}_i$ is the mean of points in set S_i.

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{x \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \qquad (1)$$

where $\boldsymbol{\mu}_i$ is the mean of points in set S_i.

k-means is an iterative algorithm but it converges quickly to a local optimum. Although k-means works well for low dimensional data, it makes little sense to apply the algorithm directly to a high dimension data like images. Due to the curse of dimensionality the Euclidean distance looses its meaning in very high dimensional spaces [1, 26].

In order to make useful clusters dimensionality reduction is applied based on simple image cues before clustering. In this paper we use two important features namely color and gradients. For clustering based on color information we create a 3D color histogram of the images in the dataset with different number of bins. The number of bins determine the ratio of dimensionality reduction. To cluster based on the edge information in the images, we have used histogram of oriented gradients (HOG) [5]. The dimensions of the reduced space depend on the image size, HOG cell size, number of orientation bins and the stride.

It is common to introduce variations in the input images by data augmentation. Data augmentation can involve both color space modifications and geometric transforms. Some techniques use extreme data augmentation of images to create pseudo classes for unsupervised learning [7]. Instead of using extreme data augmentation we introduce these variations by co-learning and changing the set of pseudo-classes that are learned simultaneously. Some kinds of data augmentation while heavily used in both supervised and unsupervised classification problems, does not help at all in other applications. For example, applying geometric transformations (e.g. horizontal flip, scale, rotate etc.) to increase training data have no advantage for surveillance of industrial robots from a fixed surveillance camera. Even though we don't use any data augmentation at all for the evaluation of the proposed approach, depending on the application some kind of augmentation might have a positive effect on the performance.

4.2. Co-Learning

The abstract concept of co-learning is shown in Figure 2. The idea behind co-learning is that learning to classify many pseudo-classes at the same time will help to generate more neutral feature representation. The feature extractor part of the deep network is followed by more then one softmax for different pseudo-labels that are co-learned. Each softmax is represented as:

$$P(y = j|\mathbf{x}) = \frac{e^{\mathbf{x}^T \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^T \mathbf{w}_k}} \tag{2}$$

where, **x** is the input and **W** is the weight matrix.

We use cross-entropy loss as the cost function for each softmax layer. Given, $p \in \{y, 1-y\}$ and $q \in \{\hat{y}, 1-\hat{y}\}$, cross entropy defines the similarity between p and q as:

$$\begin{aligned} L(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) \\ &= -\frac{1}{N}\sum_{n=1}^{N} [y_n \log\hat{y}_n + (1-y_n)\log(1-\hat{y}_n)] \end{aligned} \tag{3}$$

As the network have more than one softmax layers we use the sum of all the cross-entropy loss functions as our final cost function for back propagation.

Algorithm 2 shows the training method in detail. The process of co-learning is repeated with different set of pseudo-classes until the termination criteria is reached. As the features are not learned on a single class of labels and therefore, they are expected to be more general and unbiased. The parameters of the optimizer (e.g. learning rate etc.) may or may not be updated at the time of changing the set of pseudo-labels.

4.3. Spatio-temporal anomaly detection

It is difficult if not impossible to make a system that can predict the next expected input in the high dimensional image space. Instead of applying prediction directly to the images, we use a deep long short term memory (LSTM) [12] based recurrent neural network (RNN) to predict the next video frame in the learned feature space (Figure 3). As the feature space is an encoding of the original image space, the difference between the prediction and actual observation can reveal anomalies. The LSTM prediction network can encode the feature vectors in a video sequence both in space and time. Therefore, the proposed system is capable of detection spatial anomalies as well as temporal anomalies. Note that the prediction happens in feature space and not the original image space, as a consequence the proposed technique can generate a warning but cannot localize the position of anomaly.

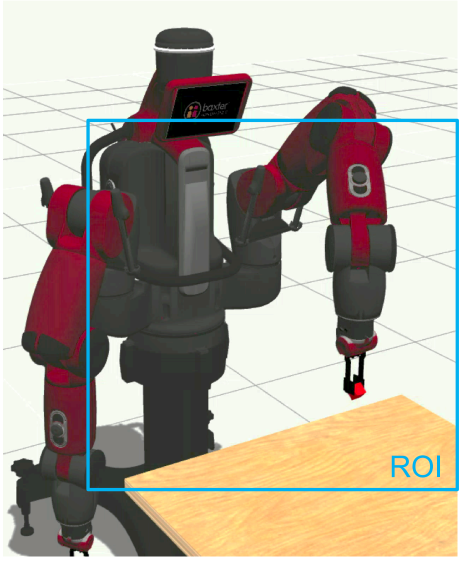

Figure 4. Robot surveillance view with the region of interest (ROI).

5. Results

In this section we show the use of the proposed technique for the surveillance of an industrial robot. We also show how the extracted features can be used in a more common classification task by using the well-known CIFAR-10 dataset [15].

5.1. Industrial robots surveillance

We used the proposed unsupervised feature learning for surveillance of industrial robots and machinery. As the industrial robots usually perform same task over and over again any irregularity in the normal operation can be detected as an anomaly. As shown in Figure 4, we perform anomaly detection using a Baxter robot [22] that is performing a pick and place operation repeatedly. Gazebo [14] was used to generate the surveillance video of the task at 10 frames per second. The object is placed at different locations and the task of the robot is to pick the object from that location and put it at another random location. Every

Table 1. List of anomalies for quantitative analysis of spatio-temporal anomaly detection. Anomalies that can be detected from a single frame are listed as spatial anomalies. The anomalies that require at least two observations are listed as temporal anomalies.

No.	Anomaly type	Category	Frames
0.	No anomaly	–	–
1.	Failed pick/Object fall	Spatial	26-62
2.	Person entering the view	Spatial	60-65
3.	Sudden changes in light condition	Spatial	20-22
4.	Sudden change in motion	Temporal	35-100
5.	Sudden stop	Temporal	96-100

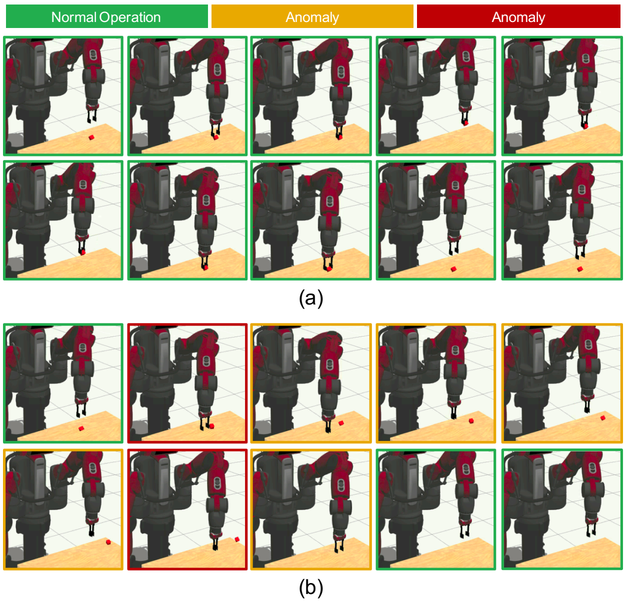

Figure 5. Robot surveillance video sequence (we show only 1 frame in 1 second). Outline color of each frames shows if the systems recognized it as an anomaly or not. (a) Normal pick and place operation with no anomaly. (b) Failed operation results in the ball rolling off the table.

10th frame of a normal pick and place operation is shown in Figure 5(a). Figure 5(b) shows a failed pick and place operation with object rolling off the table as a result of hitting the gripper. The proposed system informs the human operator if the operation is running normally or not. This is shown by the colored outline of the frame in Figure 5. Green suggests a normal operation, while red indicates an anomaly.

To verify the proposed anomaly detection technique, we have introduced some anomalies by simulation and synthetically. A list of the anomalies that we have introduced for testing our system is given in Table 1.

The training data consisted of $10,000$ image frames of the region of interest extracted from the video at the rate of 10 fps. Each image was of size 3x256x256. Instead of applying clustering directly to the images, HOG was used in different configurations to create cluster based on the edge information. In the first configuration the cell size was 16x16, gradients were discretized into 16 bins, and a stride of 16 was used to create 4096 dimension vectors for each image. In another configuration HOG was used with cell size 32x32, and a stride of 32 to make 1024 dimension output. Raw images were used for HOG based dimensionality reduction for clustering. K-means clustering with random initialization of centroids was done on the training data in the reduced dimension space. We created clusters of size $k = 10, 20, 30, 40, 50, 100, 150$ clusters for each configuration of HOG resulting in a set of 14 pseudo-labels.

The feature extraction network we use is a CNN with 3 convolutional layers and 1 fully connected layer in the end. To train the CNN we have used the raw images of the region of interest (shown in Figure 4) with size 3x256x256. First convolutional layer has 10 filters of size 11×11 with a stride of 2. The convolutional layer is followed by a max-pooling layer of size 3×3 and stride of 2. Second convolutional layer has 20 filters of size 7×7 with a stride of 2. Last convolutional layer has 40 filters of size 5×5 with a stride of 2. The last fully connected layer is of size 512. All the neurons in the convolutional and fully connected layer are rectified linear units (ReLUs). 50% dropout is applied to the output of last convolutional layer and the fully connected layer during training.

The co-learning was performed by randomly choosing any 2 of the generated pseudo-labels. They were trained using stochastic gradient descent (SGD) with initial learning rate of 0.01 and batch size was 64 for 5 epochs. After that a different set of 2 pseudo-labels was selected for learn. This process was repeated 10 times bringing the total number of epochs performed for co-learning on different labels to 50. Learning rate of SGD was decreased to 0.001 after first 10 epochs.

After converting our testing data to 512 dimensional feature space an LSTM based prediction system was used to learn the prediction of the next frame based on the past observations. To create testing data, we randomly created 700 sequences of 100 frames each from the long sequence of the $10,000$ feature vector frames. The prediction system consisted of a two layer LSTM of size $512 - 256 - 256 - 512$. The LSTM network is trained to predict the next 512 dimensional feature vector in the sequence. Adam [13] was used as optimizer algorithm for training. Mean square error was used as a cost function. The network was trained for 50 epochs with batches of size 50. LSTM cell state was reset after each training batch.

After training the prediction system we can input the current image in feature domain and generate a prediction for the next observation. This prediction is then compared with the actual observation. Instead of using means square error, we take the absolute difference of the two feature vectors and then compute mean of top 5% values in the difference vector. The reason for using the top 5% values is that if we use mean square error the values that are more or less similar will suppress the values that are significantly different. Therefore, we only consider the predicted values that are very different from the actual observation. This assumption is also consistent with the biological systems where we usually ignore small anomalies and just concentrate only on the strong anomalies.

The mean of the top 5% values give us a quantitative measure for the quality of our prediction. The results we

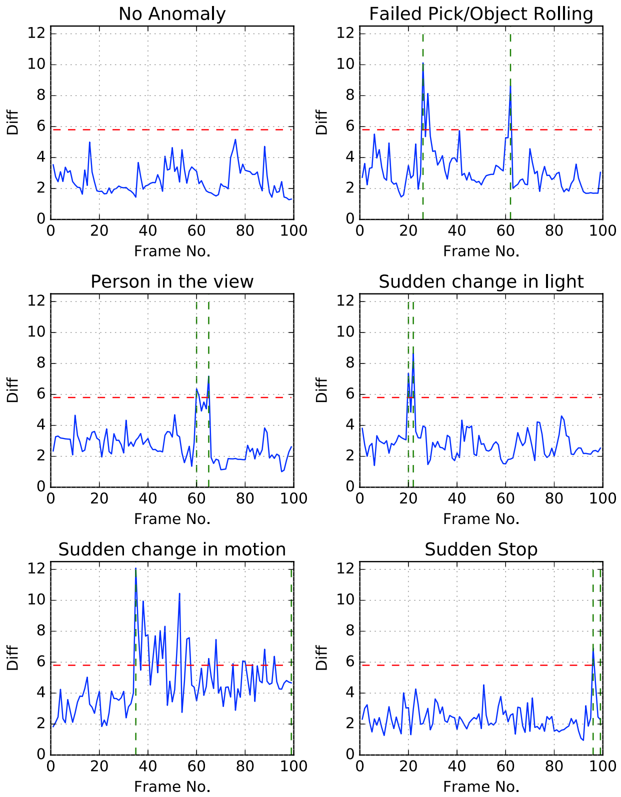

Figure 6. Anomaly detection results for the 6 test cases mentioned in Table 1. Red dotted line shows the threshold to decide anomaly. Green dotted lines define the boundaries of the frame containing anomaly.

achieve for the test cases presented in Table 1 are given in Figure 6. We can see that by deciding a threshold (shown by red dotted line), we can differentiate between a normal operation and anomaly. We selected a threshold such that no normal frames are detected as anomaly. As the threshold defines the tradeoff between false positive and false negatives, choosing an appropriate threshold is problem and data dependent.

We are using GTX TITAN equipped Intel Xeon 8 core 3.16GHz 10GB machine running Ubuntu 14.04. Training time includes the time taken by:

- Clustering - on CPU: 2694.2 s
- Feature extractor - on GPU: 1706.5 s
- Anomaly detector - on GPU: 182.3 s

At inference time, we can run the system at over 20fps on the CPU.

5.2. Using learned features for classification

Even though the proposed technique is not designed for classification task, we show how it can be used images to learn feature representation for CIFAR-10. The learned features were tested using the ground truth labels of the dataset.

CIFAR images have 3x32x32 dimensions. We applied k-mean clustering to color and edge based image cues. 3D color histogram with $8, 8, 8$ bins for red, green and blue color was used to produce 512 dimension vector for each image. HOG was also used in different configurations to cluster based on the edge information. In the first configuration the cell size was 16x16, gradients were discretized into 16 bins, and a stride of 16 was used to create 64 dimension vectors for each CIFAR image. In another configuration HOG was used with cell size 8x8, and a stride of 8 to make 256 dimension output. The dimensionality reduction was applied to both the training and the testing images of in raw form.

K-means clustering with random initialization of centroids was done only on the training data in the reduced dimension space. The testing data was assigned to the closest clusters. We created clusters of size $k = 5, 10, 20, 30, 40, 50, 100, 150, 200, 250$ clusters for each dimensionality reduction method resulting in 30 sets of pseudo-labels. The result of clustering by color histogram and HOG is shown in Figure 7. It is clearly visible that the clusters group the images based on color and edge orientation respectively.

The feature extractor network had three convolutional layers followed by a single fully connected layer. The first convolutional layer has 64 filters of size 5×5 with a padding of 2 and a stride of 1. This is followed by a max-pooling layer of filter size 3×3 and stride of 2. The next convo-

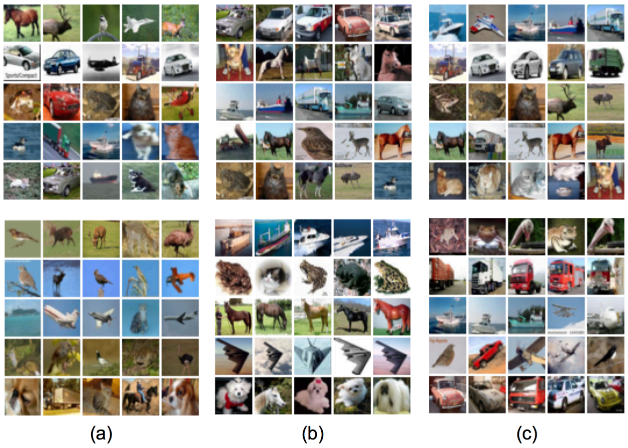

Figure 7. CIFAR-10 clusters used as pseudo-labels (Each row in an represent a different class). All clusters were generated on raw CIFAR-10 images by applying k-means with initial centroids randomly selected. For images in top row $k = 10$ and for the bottom row $k = 100$. Different algorithms were used for dimensionality reduction. (a) 3D color histogram with $8, 8, 8$ bins for red, green and blue (512 dimensions). (b) HOG is with cell size 16x16, 16 gradient bins, stride of 16 (64 dimensions). (c) HOG is with cell size 8x8, 16 gradient bins, stride of 8 (256 dimensions).

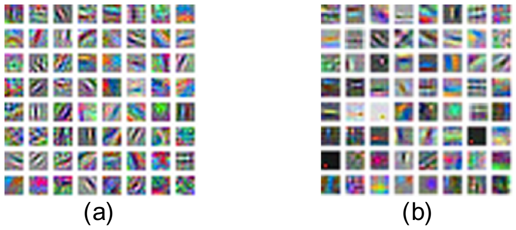

Figure 8. Filters of first convolutional layer. Different algorithms were used for dimensionality reduction. (a) Supervised learning. (b) Proposed unsupervised learning.

lutional layer had same stride and padding with 128 filters. The third layer again was similar to the previous layers with 256 filters. Both layer 2 and 3 of the network were followed by a max-pooling layer of size 3×3 and stride of 2. This is followed by a 50% dropout. The fully connected layer is 4096×2048. So our learned features will be of 2048 dimensions. The feature layer is fed to a 50% dropout before connecting them to the different softmax layers that are co-learned. All the neurons in the convolutional and fully connected layer are rectified linear units (ReLUs).

The co-learning was performed by randomly choosing any 5 of the generated pseudo-labels. They were trained using stochastic gradient descent (SGD) with initial learning rate of 0.01 and batch size was 128 for 5 epochs. After that a different set of 5 pseudo-labels was selected to learn. This process was repeated 10 times bringing the total number of epochs performed for co-learning on different labels to 50. The learning rate of SGD was changed to 0.001 after first 10 epochs. Even though the clustering was done on raw CIFAR-10 images, we used the normalized image (subtract mean and divide by standard deviation for each image independently) for training.

After the training is completed all the training and the testing data was passed through the deep CNN network to obtain the feature vectors. The feature vectors were used to train a 10-way linear SVM using the actual CIFAR-10 dataset labels. The classification accuracy reveals the effectiveness of the technique.

Without any data augmentation we can achieve 68.8% classification accuracy. The state of the art is 84.3% presented by Dosovitskiy et al. [6]. However, as discuss above our goal is not classification and while Dosovitskiy et al. rely on extreme data augmentation we do none at all. The filters learned by the first layer of convolutional neural network are shown in Figure 8.

6. Conclusions

In this paper we propose a new technique to detect anomalies in the operation of industrial robots. We do this by training a deep feature extractor in a fully unsupervised manner and combining it with a prediction system. In contrast to recent unsupervised learning techniques the proposed approach does not rely on data augmentation for learning. Due to the independence from data augmentation, the proposed technique can be used to solve the anomaly detection for industrial robots by unsupervised learning.

The proposed technique co-learns the parameters on multiple pseudo-classes at once to create unbiased feature vectors. Pseudo-classes are generated by simple clustering based on some image cues. We show how the feature extractor can be effectively combined with a prediction system for spatio-temporal anomaly detection. We demonstrated the use of such a system as an unsupervised anomaly detector for the industrial pipeline. Although not designed for classification task, we show the proposed approach can achieve reasonable performance for classification of CIFAR-10 dataset.

Looking for better image cue based clustering techniques for generation of pseudo-classes is one of the areas we want to explore in future. Furthermore, we would like to do theoretical analysis of the co-learning in unsupervised learning.

References

[1] C. C. Aggarwal. *High-Dimensional Outlier Detection: The Subspace Method*, pages 135–167. Springer New York, New York, NY, 2013.

[2] A. Basharat, A. Gritai, and M. Shah. Learning object motion patterns for anomaly detection and improved object detection. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[3] A. Coates, H. Lee, and A. Ng. An analysis of single-layer networks in unsupervised feature learning. In G. Gordon, D. Dunson, and M. Dudk, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR Workshop and Conference Proceedings*, pages 215–223. JMLR W&CP, 2011.

[4] A. Coates and A. Y. Ng. *Learning Feature Representations with K-Means*, pages 561–580. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[6] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1734–1747, Oct 2016. TPAMI-2015-05-0348.R1.

[7] A. Dosovitskiy, J. T. Springenberg, M. A. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. *CoRR*, abs/1406.6909, 2014.

[8] A. Dundar, J. Jin, and E. Culurciello. Convolutional clustering for unsupervised learning. *CoRR*, abs/1511.06241, 2015.

[9] T. Egner, J. M. Monti, and C. Summerfield. Expectation and surprise determine neural population responses in the ventral

visual stream. *The Journal of Neuroscience*, 30(49), December 2010.

[10] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1778–1785, June 2009.

[11] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[14] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154 vol.3, Sept 2004.

[15] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[17] W. Lawson, L. Hiatt, and K. Sullivan. Detecting anomalous objects on mobile platforms. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.

[18] Q. V. Le, M. Ranzato, R. Monga, M. Devin, G. Corrado, K. Chen, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

[19] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[20] T. L. Paine, P. Khorrami, W. Han, and T. S. Huang. An analysis of unsupervised pre-training in light of recent advances. *CoRR*, abs/1412.6597, 2014.

[21] M. Ranzato, C. Poultney, S. Chopra, and Y. L. Cun. Efficient learning of sparse representations with an energy-based model. In B. Schlkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1137–1144. MIT Press, Cambridge, MA, 2006.

[22] Rethink Robotics Inc. Baxter. http://velodynelidar.com/hdl-64e.html.

[23] B. Saleh, A. Farhadi, and A. Elgammal. Object-centric anomaly detection by attribute-based reasoning. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 787–794, Washington, DC, USA, 2013. IEEE Computer Society.

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[25] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, Dec. 2010.

[26] J. Xie, R. B. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. *CoRR*, abs/1511.06335, 2015.

[27] D. Xu, E. Ricci, Y. Yan, J. Song, and N. Sebe. Learning deep representations of appearance and motion for anomalous event detection. In X. Xie, M. W. Jones, and G. K. L. Tam, editors, *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*, pages 8.1–8.12. BMVA Press, 2015.