This CVPR 2020 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Graph Embedded Pose Clustering for Anomaly Detection

Amir Markovitz[1], Gilad Sharir[2], Itamar Friedman[2], Lihi Zelnik-Manor[2], and Shai Avidan[1]

[1]Tel-Aviv University, Tel-Aviv, Israel [2]Alibaba Group

{markovitz2@mail, avidan@eng}.tau.ac.il, {first.last}@alibaba-inc.com

Abstract

We propose a new method for anomaly detection of human actions. Our method works directly on human pose graphs that can be computed from an input video sequence. This makes the analysis independent of nuisance parameters such as viewpoint or illumination. We map these graphs to a latent space and cluster them. Each action is then represented by its soft-assignment to each of the clusters. This gives a kind of "bag of words" representation to the data, where every action is represented by its similarity to a group of base action-words. Then, we use a Dirichlet process based mixture, that is useful for handling proportional data such as our soft-assignment vectors, to determine if an action is normal or not.

We evaluate our method on two types of data sets. The first is a fine-grained anomaly detection data set (e.g. ShanghaiTech) where we wish to detect unusual variations of some action. The second is a coarse-grained anomaly detection data set (e.g., a Kinetics-based data set) where few actions are considered normal, and every other action should be considered abnormal.

Extensive experiments on the benchmarks show that our method[1] performs considerably better than other state of the art methods.

1. Introduction

Anomaly detection in video has been investigated extensively over the years. This is because the amount of video captured far surpasses our ability to manually analyze it. Anomaly detection algorithms are designed to help human operators deal with this problem. The question is how to define anomalies and how to detect them.

The decision of whether an action is normal or not is nuanced. In some cases, we are interested in detecting abnormal variations of an action. For example, an abnormal type of walking. We term this fine-grained anomaly detection. In other cases, we might be interested in defining normal actions and regard any other action as abnormal. For example, we might be interested in determining that dancing is normal, while gymnastics are abnormal. We call this coarse-grained anomaly detection.

We desire an algorithm that can handle both types of anomaly detection in a single, unified fashion. Such an algorithm should take as input an unlabeled set of videos that capture normal actions *only* (fine- or coarse-grained) and use that to train a model that will distinguish normal from abnormal actions.

We take advantage of the recent progress in human pose estimation and assume our algorithm takes human pose graphs as input. This offers several advantages. First, it abstracts the problem and lets the algorithm focus on human pose and not on irrelevant features such as viewing direction, illumination, or background clutter. In addition, a human pose can be represented as a compact graph, which makes analyzing, training and testing much faster.

Given a sequence of video frames, we use a pose estimation method to extract the keypoints of every person in each frame. Every person in a clip is represented as a temporal pose graph. We use a combination of an autoencoder and a clustering branch to map the training samples into a latent space where samples are soft clustered. Each sample is then represented by its soft-assignment to each of the k clusters. This can be understood as learning a bag-of-words representation for actions. Each cluster corresponds to an action-word, and each action is represented by its similarity to each of the action-words. Figure 1 gives an overview of our method.

The soft-assignment vectors capture proportional data and the tool to measure their distribution is the Dirichlet Process Mixture Model. Once we fit the model to the data, we can obtain a normality score for each sample and determine if the action is to be classified as normal or not.

The algorithm thus consists of a series of abstractions. Using human pose graphs eliminates the need to deal with viewpoint and illumination changes. And the soft-assignment representation abstracts the type of data (fine-grained or coarse-grained) from the Dirichlet model.

[1]Code available at: https://github.com/amirmk89/gepc

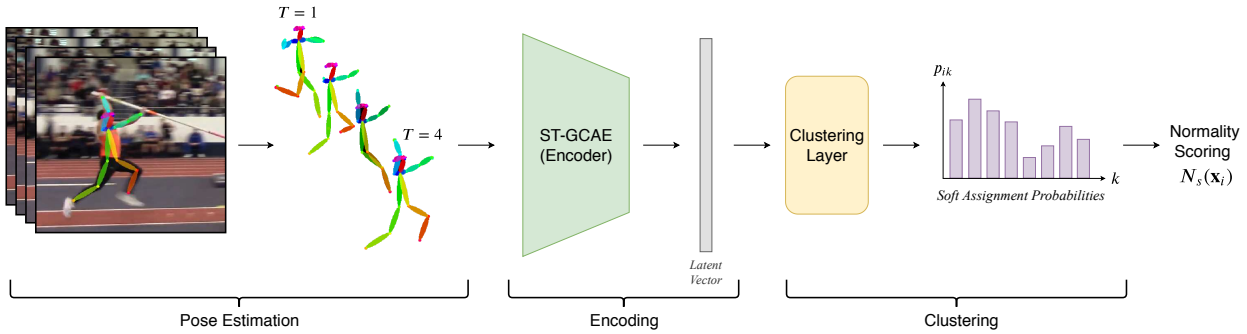


Figure 1. **Model Diagram (Inference Time):** To score a video, we first perform pose estimation. The extracted poses are encoded using the encoder part of a Spatio-temporal graph autoencoder (ST-GCAE), resulting in a latent vector. The latent vector is soft-assigned to clusters using a deep clustering layer, with p_{ik} denoting the probability of sample $\mathbf{x}_i$ being assigned to cluster k.

We evaluate our algorithm in two settings. The first is the *ShanghaiTech Campus* [16] dataset, a large and extensively evaluated anomaly detection benchmark. This is a typical (*fine-grained*) anomaly detection benchmark in which normal behavior is taken to be walking, and the goal is to detect abnormal events, such as people running, fighting, riding bicycles, throwing objects, etc.

The second is a new problem setting we propose, and denote *Coarse-grained* anomaly detection. Instead of focusing on a single action (i.e., walking), as in the ShanghaiTech dataset, we construct a training set consisting of a varying number of actions that are to be regarded as normal. For example, the training set may consist of video clips of different dancing styles. At test time, every dance video should be classified as normal, while any other action should be classified as abnormal.

We demonstrate this new, challenging, *Coarse-grained anomaly detection* setting on two action classification datasets. First is the *NTU-RGB+D* dataset, where 3D body joints are detected using Kinect. Second is a larger and more challenging dataset that consists of 250 out of the 400 actions in the *Kinetics400* dataset[2]. For both datasets, we use a subset of the actions to define a training set of normal actions and use the rest of the videos to test if the algorithm can correctly distinguish normal from abnormal videos.

We conduct extensive experiments, compare to a number of competing approaches and find that our algorithm outperforms all of them.

To summarize, we propose three key contributions:

- The use of embedded pose graphs and a Dirichlet process mixture for video anomaly detection;
- A new *coarse-grained* setting for exploring broader aspects of video anomaly detection;
- State-of-the-art AUC of 0.761 for the *ShanghaiTech Campus* anomaly detection benchmark.

[2]We only use a subset of the classes as not all classes can be detected using human pose detectors.

2. Background

2.1. Video Anomaly Detection

The field of anomaly detection is broad and has a large variation in setting and assumptions, as is evident by the different datasets proposed to evaluate methods in the field.

For our fine-grained experiment, we use the *ShanghaiTech Campus dataset* [16]. Containing 130 anomalous events in 13 different scenes, with various camera angles and lighting conditions, it is more diverse and significantly larger than all previous common datasets. It is presented in detail in section 4.1.

In recent years, numerous works tackled the problem of anomaly detection in video using deep learning based models. Those could be roughly categorized into reconstructive models, predictive models, and generative models.

Reconstructive models learn a feature representation for each sample and attempt to reconstruct a sample based on that embedding, often using Autoencoders [1, 6, 10]. Predictive model based methods aim to model the current frame based on a set of previous frames, often relying on recurrent neural networks [15, 16, 17] or 3D convolutions [21, 29]. In some cases, reconstruction-based models are combined with prediction based methods for improved accuracy [29]. In both cases, samples poorly reconstructed or predicted are considered anomalous.

Generative models were also used to reconstruct, predict or model the distribution of the data, often using Variational Autoencoders (VAEs) [3] or GANs [2, 14, 19, 20].

A method proposed by Liu *et al*. [13] uses a generative future frame prediction model and compares a prediction with its ground truth by evaluating differences in gradient-based features and optic flow. This method requires optic flow computation and generating a complete scene, which makes it costly and less robust to large scenery changes.

Recently, Morais *et al*. [18] proposed an anomaly detection method using a fully connected RNN to analyze pose

sequences. The method embeds a sequence, then uses reconstruction and prediction branches to generate past and future poses, respectively. Anomaly score is determined by the reconstruction and prediction errors of the model.

2.2. Graph Convolutional Networks

To represent human poses as graphs, the inner-graph relations are described using weighted adjacency matrices. Each matrix could be static or learnable and represent any kind of relation.

In recent years, many approaches were proposed for applying deep learning based methods to graph data. Kipf and Welling [12] proposed the notion of *Fast Approximate Convolutions On Graphs*. Following Kipf and Welling, both temporal and multiple adjacency extensions were proposed. Works by Yan *et al*. [27] and Yu *et al*. [28] proposed temporal extensions, with the former work proposing the use of separable spatial and temporal graph convolutions (ST-GCN), applied sequentially. We follow the basic ST-GCN block design, illustrated in Figure 2.

Veličković *et al*. [24] proposed Graph Attention Networks, a GCN extension in which the weighting of neighboring nodes are inferred using an attention mechanism, relying on a fixed adjacency matrix only to determine neighboring nodes.

Shi *et al*. [23] recently extended the concept of spatio-temporal graph convolutions by using several adjacency matrices, of which some are learned or inferred. Inferred adjacency is determined using an embedded similarity measure, optimized during training. Adjacency matrices are summed prior to applying the convolution.

2.3. Deep Clustering Models

Deep clustering methods aim to provide useful cluster assignments by optimizing a deep model under a cluster inducing objective. For example, several recent methods jointly embed and cluster data using unsupervised representation learning methods, such as autoencoders, with clustering modules [5, 8, 25, 26].

A method proposed by Xie *et al*. [26], denoted *Deep Embedded Clustering (DEC),* proposed an alternating two-step approach. In the first step, a target distribution is calculated using the current cluster assignments. In the next step, the model is optimized to provide cluster assignments similar to the target distribution. Recent extensions tackled DEC's susceptibility to degenerate solutions using regularization methods and various post-processing means [8, 9].

3. Method

We design an anomaly detection algorithm that can operate in a number of different scenarios. The algorithm consists of a sequence of abstractions that are designed to help each step of the algorithm work better. First, we use a human pose detector on the input data. This abstracts the problem and prevents the next steps from dealing with nuisance parameters such as viewpoint or illumination changes.

Human actions are represented as space-time graphs and we embed (sub-sections 3.1, 3.2) and cluster (sub-section 3.3) them in some latent space. Each action is now represented as a soft-assignment vector to a group of base actions. This abstracts the underlying type of actions (i.e., fine-grained or coarse-grained), leading to the final stage of learning their distribution. The tool we use for learning the distribution of soft-assignment vectors is the Dirichlet process mixture (sub-section 3.4), and we fit a model to the data. This model is then used to determine if an action is normal or not.

3.1. Feature Extraction

We wish to capture the relations between body joints, while at the same time provide robustness to external factors such as appearance, viewpoint and lighting. Therefore, we represent a person's pose with a graph.

Each node of the graph corresponds to a keypoint, a body joint, and each edge represents some relation between two nodes. Many keypoint relations exist, such as physical relations defined anatomically (e.g. the left wrist and elbow are connected) and action relations defined by movements that tend to be highly correlated in the context of a certain action (e.g. the left and right knees tend to move in opposite directions while running). The directions of the graph rise from the fact that some relations are learned during the optimization process and are not symmetric. A nice bonus with this representation is being compact, which is very important for efficient video analysis.

In order to extend this formulation temporally, pose keypoints extracted from a video sequence are represented as a temporal sequence of pose graphs. The temporal pose graph is a time series of human joint locations. Temporal domain adjacency could be similarly defined by connecting joints in successive frames, allowing us to perform graph convolution operations exploiting both spatial and temporal dimensions of our sequence of pose graphs.

We propose a deep temporal graph autoencoder based architecture for embedding the temporal pose graphs. Building on the basic block design of ST-GCN, presented in Figure 2, we substitute the basic GCN operator with a novel Spatial Attention Graph Convolution, presented next.

We use this building block to construct a Spatio-Temporal Graph Convolutional Auto-Encoder, or *ST-GCAE*. We use ST-GCAE to embed the spatio-temporal graph and take the embedding to be the starting point for our clustering branch.

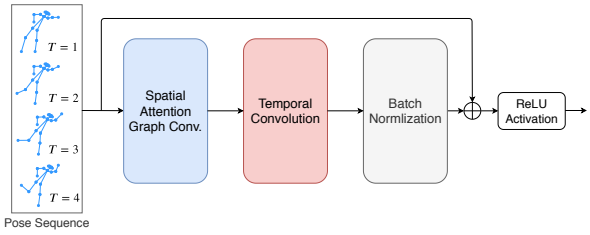


Figure 2. **Spatio-Temporal Graph Convolution Block:** The basic block used for constructing ST-GCAE. A spatial attention graph convolution (Figure 3) is followed by a temporal convolution and batch normalization. A residual connection is used.

3.2. Spatial Attention Graph Convolution

We propose a new graph operator, presented in Figure 3, that uses adjacency matrices of three types: Static, Globally-learned and Inferred (attention-based). Each adjacency type is applied with its own GCN, using separate weights. The outputs from the GCNs are stacked in the channel dimension. A 1×1 convolution is applied as a learnable reduction measure for weighting the stacked outputs, and provides the required output channel number.

The three adjacency matrices capture different aspects of the model: (i) The use of body-part connectivity as a prior over node relations, represented using the static adjacency matrix. (ii) Dataset level keypoint relations, captured by the global adjacency matrix, and (iii) Sample specific relations, captured by inferred adjacency matrices. Finally, the learnable reduction measure weights the different outputs.

The static adjacency A is fixed and shared by all layers. The globally-learnable matrix B is learned individually at each layer, and applied equally to all samples during the forward pass. The inferred adjacency matrices C are based on an attention mechanism that uses learned weights to calculate a sample specific adjacency matrix, a different one for every sample in a batch. For example, for a batch of size N of graphs with V nodes, the inferred adjacency size is $[N, V, V]$, while other adjacencies are $[V, V]$ matrices.

The globally-learned adjacency is learned by initializing a fully-connected graph, with a complete, uniform, adjacency matrix. The matrix is jointly optimized with the rest of the model's parameters during training. The computational overhead of this adjacency is small for graphs containing no more than a few dozen nodes.

An inferred adjacency matrix is constructed using a graph self-attention layer. After evaluating a few attention models we chose a simple multiplicative attention mechanism. First, we embed the input twice, using two sets of learned weights. We then transpose one of the embedded matrices and take the dot product between the two and normalize. We then get the inferred adjacency matrix. The attention mechanism chosen is modular and may be replaced with other common alternatives. Further details are provided in the supplementary material.

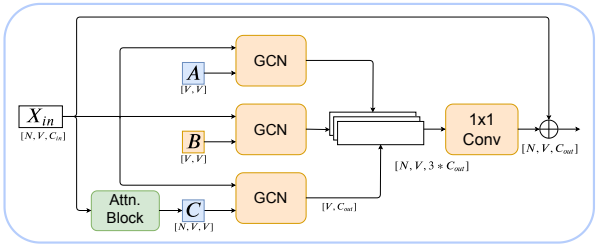


Figure 3. **Spatial-Attention Graph Convolution:** A zoom into our spatial graph convolving operator, comprised of three GCN [12] operators: One using a hard-coded physical adjacency matrix (A), the second using a global adjacency matrix learned during training (B), and the third using an adjacency matrix inferred using an attention submodule (C). A residual connection is used. GCN modules include batch normalization and ReLU activation, omitted for readability.

3.3. Deep Embedded Clustering

To build our dictionary of underlying actions, we take the training set samples and jointly embed and cluster them in some latent space. Each sample is then represented by its assignment probability to each of the underlying clusters. The objective is selected to provide distinct latent clusters, over which actions reside.

We adapt the notion of *Deep Embedded Clustering* [26] for clustering temporal graphs with our ST-GCAE architecture. The proposed clustering model consists of three parts, an encoder, a decoder, and a soft clustering layer.

Specifically, our ST-GCAE model maintains the graph's structure but uses large temporal strides with an increasing channel number to compress an input sequence to a latent vector. The decoder uses temporal up-sampling layers and additional graph convolutional blocks, for gradually restoring original channel count and temporal dimension.

The ST-GCAE's embedding is the starting point for clustering the data. The initial reconstruction based embedding is fine-tuned during our clustering optimization stage to reach the final clustering optimized embedding.

For each input sample $\mathbf{x}_i$, we denote the encoder's latent embedding by $\mathbf{z}_i$, and the soft cluster assignment calculated using the clustering layer by $\mathbf{y}_i$. We denote the clustering layer's parameters by Θ. The probability p_{ik} for the i-th sample to be assigned to the k-th cluster is:

$$p_{ik} = Pr(y_i = k|\mathbf{z}_i, \Theta) = \frac{exp(\boldsymbol{\theta}_k^T \mathbf{z}_i)}{\sum_{k'=1}^{K} exp(\boldsymbol{\theta}_{k'}^T \mathbf{z}_i)}. \quad (1)$$

We adopt the clustering objective and optimization algorithm proposed by [26]. The clustering objective is to minimize the KL divergence between the current model probabilistic clustering prediction P and a target distribution Q:

$$L_{cluster} = KL(Q||P) = \sum_i \sum_k q_{ik} log \frac{q_{ik}}{p_{ik}}. \quad (2)$$

The target distribution aims to strengthen current cluster assignments by normalizing and pushing each value closer to a value of either 0 or 1. Recurrent application of the function transforming P to Q will eventually result in a hard assignment vector. Each member of the target distribution is calculated using the following equation:

$$q_{ik} = \frac{p_{ik}/(\sum_{i'} p_{i'k})^{\frac{1}{2}}}{\sum_{k'} p_{ik'}/(\sum_{i'} p_{i'k'})^{\frac{1}{2}}}. \quad (3)$$

The clustering layer is initialized by the K-means centroids calculated for the encoded training set. Optimization is done in Expectation-Maximization (EM) like fashion. During the Expectation step, the entire model is fixed and, the target distribution Q is updated. During the Maximization stage, the model is optimized to minimize the clustering loss, $L_{cluster}$.

3.4. Normality Scoring

This model supports two types of multimodal distributions. One is at the cluster assignment level; the other is at the soft-assignment vector level. For example, an action may be assigned to more than one cluster (cluster-level assignment), leading to a multimodal soft-assignment vector. The soft-assignment vectors themselves (that capture actions) can be modeled by a multimodal distribution as well.

The Dirichlet process mixture model (DPMM) is a useful measure for evaluating the distribution of proportional data. It meets our required setup: (i) An estimation (fitting) phase, during which a set of distribution parameters is evaluated, and (ii) An inference stage, providing a score for each embedded sample using the fitted model. A thorough overview of the model is given by Blei and Jordan [4].

The DPMM is a common mixture extension to the unimodal Dirichlet distribution and uses the Dirichlet Process, an infinite-dimensional extension of the Dirichlet distribution. This model is multimodal and able to capture each mode as a mixture component. A fitted model has several modes, each representing a set of proportions that correspond to one normal behavior. At test time, each sample is scored by its log probability using the fitted model. Further explanations and discussion on the use of DPMM are available in [4, 7].

3.5. Training

The training phase of the model consists of two stages, a pre-training stage for the autoencoder, in which the clustering branch of the network remains unchanged, and a fine-tuning stage in which both embedding and clustering are optimized. In detail:

Pre-Training: the model learns to encode and reconstruct a sequence by minimizing a reconstruction loss, denoted L_{rec}, which is an L_2 loss between the original temporal pose graphs and those reconstructed by ST-GCAE.

Fine-Tuning: the model optimizes a combined loss function consisting of both the reconstruction loss and a clustering loss. Optimization is done such that the clustering layer is optimized w.r.t. $L_{cluster}$, the decoder is optimized w.r.t. L_{rec} and the encoder is optimized w.r.t. both. The initialization of the clustering layer is done via K-means. As shown by [8], while the encoder is optimized w.r.t. to both losses, the decoder is kept and acts as a regularizer for maintaining the embedding quality of the encoder. The combined loss for this stage is:

$$L_{combined} = L_{rec} + \lambda \cdot L_{cluster}. \quad (4)$$

4. Experiments

We evaluated our model in two different settings, using three datasets. The first setting is the common video anomaly detection setting, which we denote as the *Fine-grained* setting. In this setting, the normal sample consists of a single class and we seek to find fine-grained variations compared to it. For this setting, we use the *ShanghaiTech Campus* dataset. The second is our new problem setting, which we denote *Coarse-grained* anomaly detection, in which we seek to find abnormal actions that are different from those defined as normal.

4.1. ShanghaiTech Campus

Dataset The *ShanghaiTech Campus dataset* [16] is one of the largest and most diverse datasets available for video anomaly detection. Presenting mostly person-based anomalies, it contains 130 abnormal events captured in 13 different scenes with complex lighting conditions and camera angles. Clips contain any number of people, from no people at all to over 20 people. The dataset contains over 300 untrimmed training and 100 untrimmed testing clips ranging from 15 seconds to over a minute long.

Experimental Setting An experiment is comprised of two data splits, a training split containing *normal* examples only and a test split containing both *normal* and *abnormal* examples. Training is conducted solely using the training split. A score is calculated for each frame individually, and the combined score is the area under ROC curve for the concatenation of all frame scores in the test set.

We evaluate video streams of unknown length using a sliding-window approach. We split the input pose sequence to fixed-length, overlapping segments and score each individually. For clips with more than a single person, each person is scored individually. The maximal score over all the people in the frame is taken. As the *ShanghaiTech Campus* dataset is not annotated for pose, we use a 2D pose estimation model to extract human pose from every clip.

ShanghaiTech Campus	
Luo *et al*. [16]	0.680
Abati *et al*. [1]	0.725
Liu *et al*. [13]	0.728
Morais *et al*. [18]	0.734
Ours - Pose	**0.752**
Ours - Patches	**0.761**

Table 1. **Fine-Grained Anomaly Detection Results:** Scores represent frame level AUC. [18] uses keypoint coordinates as input.

We also evaluate our model using patch embeddings as input features instead of keypoint coordinates. Patches of pixel RGB data are cropped from around each keypoint. The patches are embedded using a CNN and patch feature vectors are used to embed each keypoint. All other aspects of the models are kept the same.

Given the use of a pose estimation model, the patch embedding may be taken from one of the pose estimation model's hidden layers, requiring no additional computation compared to the coordinate-based variant, other than increased dimension for the input layer. Further details regarding this variant of our model, implementation, and the pose estimation method used are available in the supplemental material.

Evaluation We follow the evaluation protocol of Luo *et al*. [16] and report the *Area under ROC Curve (AUC)* for our model in Table 1. *'Pose'* denotes the use of keypoint coordinates as the initial graph node embedding. *'Patch'* denotes the use of patch embeddings vectors, as discussed in this section. Our model outperforms previous state of the art methods, both pose and pixel based, by a large margin.

4.2. Coarse-Grained Anomaly Detection

4.2.1 Experimental Setting

For our second setting of Coarse-Grained Anomaly Detection, a model is trained using a sample of a few action classes considered normal. Training is done without labels, in an unsupervised manner. The model is evaluated by its ability to tell whether a new unseen clip belongs to any of the actions that make up the normal sample. For this setting, we adopt two action recognition datasets to our needs. This gives us great flexibility and control over the type of normal/abnormal actions that we want to detect. The datasets are *NTU-RGB+D* and *Kinetics-250* that are provided with clip level action labels.

In this setting, we first select 3-5 action classes and denote them our *split*. Classes are grouped into two sets of samples, *split* samples, and *non-split* samples. All labels are dropped. No labels are used beyond this point, except for the final evaluation phase.

We conduct two complementary experiments. *Few vs. Many* where there are few normal actions (say 3-5) in the training set and many (tens or even hundreds) actions that are denoted abnormal in the test set. We then repeat the experiment but switch roles of the train and test sets and denote this as *Many vs. Few*.

We repeat the above experiments for two types of splits. The first kind, termed *random splits*, is made of sets of 3-5 classes selected at random from each dataset. The second, which we call *meaningful splits*, is made of action splits that are subjectively grouped following some binding logic regarding the action's physical or environmental properties. A sample of meaningful and random splits is provided in Table 3. We use 10 random and 10 meaningful splits for evaluating each dataset.

4.2.2 Methods Evaluated

We compare our algorithm to several anomaly detection algorithms. All algorithms but the last one are unsupervised:

Autoencoder reconstruction loss We use the reconstruction loss of our ST-GCAE model. In all experiments, the ST-GCAE reached convergence prior to the deep clustering fine-tuning stage. Further optimization of the ST-GCAE yielded no consistent improvement in results.

Autoencoder based one-class SVM We fit a one-class SVM model using the encoded pose sequence representations (denoted $\mathbf{z}_i$ in section 3.3). During test time, the corresponding representation of each sample is scored using the fitted model.

Video anomaly detection methods We train the *Future Frame Prediction* model proposed by Liu *et al*. [13] and the *Skeleton Trajectory* model proposed by Morais *et al*. [18] using our various dataset splits. Anomaly scores for each video are obtained by averaging the per-frame scores provided by the model. As the method proposed by Morais *et al*. only handles 2D pose, it was not applied to the 3D annotated NTU dataset.

Classifier softmax scores The *supervised* baseline uses a classifier trained to classify each of the classes from the dataset split. The classifier architecture is based on the one proposed by [27]. To handle the significantly smaller number of samples, we use a shallower variant. For classifier architecture and implementation details, see suppl.

During the evaluation phase, a sample is passed through the classifier and its softmax output values are recorded. Anomaly score in this method is calculated by either using the softmax vector's max value or by using the Dirichlet normality score from section 3.4, using softmax probabilities as input. We found Dirichlet based scoring to perform better for most cases, and we report results based on it.

Method	NTU-RGB+D				Kinetics-250			
	Few vs. Many		Many vs. Few		Few vs. Many		Many vs. Few	
	Random	*Meaningful*	*Random*	*Meaningful*	*Random*	*Meaningful*	*Random*	*Meaningful*
Supervised	0.86	0.83	0.82	0.90	0.77	0.71	0.63	0.82
Rec. Loss	0.50	0.54	0.53	0.54	0.45	0.46	0.51	0.61
OC-SVM	0.60	0.67	0.60	0.69	0.56	0.56	0.52	0.47
Liu *et al*. [13]	0.57	0.64	0.56	0.63	0.55	0.60	0.55	0.58
Morais *et al*. [18]	-	-	-	-	0.57	0.59	0.56	0.58
Ours	**0.73**	**0.82**	**0.72**	**0.85**	**0.65**	**0.73**	**0.62**	**0.74**

Table 2. **Coarse-Grained Experiment Results:** Values represent area under the ROC curve (AUC). In bold are the results of the best performing *unsupervised* method. Underlined is the best method of all. For all experiments $K = 20$ clusters, see section 3.3 for details. It should be noted that AUC=0.50 in case of random choice.

It is important to note that this method is fundamentally different from our method and the other baselines. The classifier based method is a *supervised* method, relying on class action labels that were not used by other methods. It is thus not directly comparable and is here for reference only.

	Kinetics
Random 1	Arm wrestling (6), Crawling baby (77) Presenting weather forecast (254), Surfing crowd (336)
Dancing	Belly dancing (18), Capoeira (43), Line dancing (75), Salsa (283), Tango (348), Zumba (399)
Gym	Lunge (183), Pull Ups (255), Push Up (260), Situp (305), Squat (330)
	NTU-RGB+D
Office	Answer phone (28), Play with phone/tablet (29), Typing on a keyboard (30), Read watch (33)
Fighting	Punching (50), Kicking (51), Pushing (52), Patting on back (53)

Table 3. **Split Examples:** A subset of the random and meaningful splits used for evaluating *Kinetics* and *NTU-RGB+D* datasets. For each split we list the included classes. Numbers in parentheses are the numeric class labels. For a complete list, See suppl.

4.2.3 Datasets

NTU-RGB+D The *NTU-RGB+D* dataset by Shahroudy *et al*. [22] consists of clips showing one or two people performing one of 60 action classes. Classes include both actions of a single person and two-person interactions, captured using static cameras. It is provided with 3D joint measurements that are estimated using a Kinect depth sensor.

For this dataset, we use a model configuration similar to the one used for the *ShanghaiTech* experiments, with dimensions adapted for 3D pose.

Kinetics-250 The *Kinetics* dataset by Kay *et al*. [11] is a collection of 400 action classes, each with over 400 clips that are 10 seconds long. The clips were downloaded from YouTube and may contain any number of people that are not guaranteed to be fully visible.

Since Kinetics was not intended originally for pose estimation, some classes are unidentifiable by human pose extraction methods, e.g., the *hair braiding* class contains mostly clips focused on arms and heads. For such videos, a full-body pose estimation algorithm will yield zero keypoints for most cases.

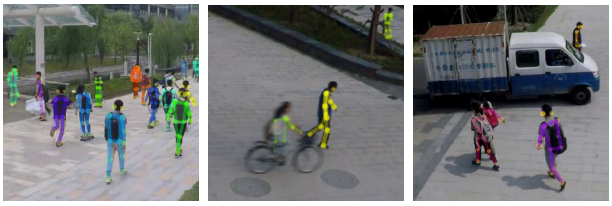
Therefore, we use a subset of *Kinetics-400* that is suitable for evaluation using pose sequences. To do that, we turn to the action classification results of [27]. Using their publicly available model we pick a subset of the 250 best-performing action classes, ranked by their *top-1* training classification accuracy. The accuracy of the class that had the lowest score is 18%. We denote our subset *Kinetics-250*.

Due to the vast size of Kinetics ($\sim$1000x larger than ShanghaiTech), we used a single GCN for the spatial convolution, using static A adjacency matrices only, and no pooling. This makes this block identical to the one proposed by [27], used for this specific setting *only*. We quantify the degradation of this variant in the suppl. *Kinetics* is not annotated for pose and we use a 2D pose estimation model.

4.2.4 Evaluation

We report *Area under ROC Curve (AUC)* results in Table 2. As these datasets require clip level annotations, the sliding window approach is not required for our method, and each temporal pose graph is evaluated in a single forward pass, with the highest scoring person taken.

As can be seen, our algorithm outperforms all four competing (unsupervised) methods, often by a large margin. The algorithm works well in both random and meaningful split modes, as well as in the Few vs. many and Many vs. few settings. Observe, however, that the algorithm works



(a) (b) (c)

Figure 4. **Failure Cases, *ShanghaiTech*:** Frames overlayed with extracted pose. In Column (a), the large crowd is occluding the abnormal skater and each other causing multiple misses. Column (b) depicts a cyclist, considered abnormal. Fast movement caused pose estimation failure, preventing detection. Column (c) depicts a vehicle in the frame, which is not addressed by our method.

better on the meaningful splits (compared to the random splits). We believe this is because meaningful splits share similar patterns.

The table also reveals the impact of the quality of pose estimation on results. That is, the *NTU-RGB+D* dataset is cleaner and the human pose is recovered using the Kinect depth sensor. As a result, the estimated poses are more accurate and the results are generally better than the *Kinetics-250* dataset.

4.3. Fail Cases

Figure 4 shows some failure cases. The recovered pose graph is superimposed on the image. As can be seen, there is significant variability in scenes, viewpoints and poses of the people in a single clip. Depicted in column (a), a highly crowded scene causes numerous occlusions and people being partially detected. The large number of partially extracted people causes a large variation in model provided scores, and misses the abnormal skater for multiple frames.

The two failures depicted in columns (b-c) show the weakness of relying on extracted pose for representing actions in a clip. Column (b) shows a cyclist very partially extracted by the pose estimation method and missed by the model. Column (c) shows a non-person related event, not handled by our model. Here, a vehicle crossing the frame.

4.4. Ablation Study

We conduct a number of experiments to evaluate the robustness of our model to noisy normal training sets, i.e., having some percentage of abnormal actions present in the training set, presented next. We also conduct experiments to evaluate the importance of key model components and the stages of our clustering approach, presented in the suppl.

Robustness to Noise In many scenarios, it is impossible to determine whether a dataset contains nothing but normal samples, and some robustness to noise is required. To evaluate the model's robustness to the presence of abnormal examples in the normal training sample, we introduce a varying number of abnormal samples chosen at random to the training set. These are taken from the unused abnormal portion of the dataset. Results are presented in Figure 5. Our model is robust and handles a large amount of abnormal data during training with little performance loss.

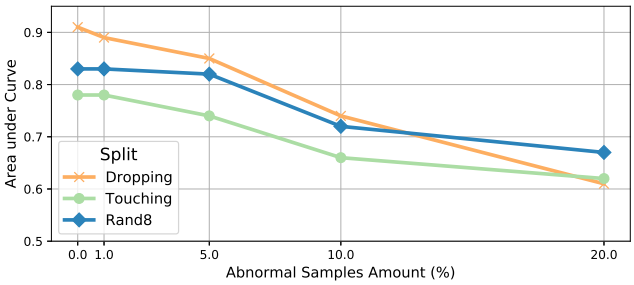


Figure 5. **AUC Loss for Training with Noisy Data:** Performance of models trained for NTU-RGB+D splits when a percentage of abnormal samples are added at random. The model is robust to significant amount of noise. At 20%, noise surpasses the amount of data for some of the underlying classes making up the split. Different curves denote different dataset splits.

For most anomaly detection settings, events occurring at a 5% rate are considered very frequent. Our model loses on average less than 10% of performance when trained with this amount of distractions. When trained with 20% abnormal noise, there is a considerable decline in performance. In this setting, the training set usually consists of 5 classes, so 20% distraction rate may be larger than an individual underlying class.

5. Conclusion

We propose an anomaly detection algorithm that relies on estimated human poses. The human poses are represented as temporal pose graphs and we jointly embed and cluster them in a latent space. As a result, each action is represented as a soft-assignment vector in latent space. We analyze the distribution of these vectors using the Dirichlet Process Mixture Model. The normality score provided by the model is used to determine if the action is normal or not.

The proposed algorithm works on both fine-grained anomaly detection, where the goal is to detect variations of a single action (e.g., walking), as well as a new coarse-grained anomaly detection setting, where the goal is to distinguish between normal and abnormal actions.

Extensive experiments show that we achieve state-of-the-art results on ShanghaiTech, one of the leading (fine-grained) anomaly detection data sets. We also outperform existing unsupervised methods on our new coarse-grained anomaly detection test.

References

[1] Davide Abati, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Latent space autoregression for novelty detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 6

[2] Samet Akçay, Amir Atapour-Abarghouei, and Toby P Breckon. Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection. *arXiv preprint arXiv:1901.08954*, 2019. 2

[3] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2:1–18, 2015. 2

[4] David M Blei and Michael I Jordan. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006. 5

[5] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, 2018. 3

[6] Yong Shean Chong and Yong Haur Tay. Abnormal event detection in videos using spatiotemporal autoencoder. *Lecture Notes in Computer Science*, page 189196, 2017. 2

[7] Or Dinari, Angel Yu, Oren Freifeld, and John W Fisher III. Distributed mcmc inference in dirichlet process mixture models using julia. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 518–525, 2019. 5

[8] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 3, 5

[9] Philip Haeusser, Johannes Plapp, Vladimir Golkov, Elie Aljalbout, and Daniel Cremers. Associative deep clustering: Training a classification network with no labels. In *German Conference on Pattern Recognition*. Springer, 2018. 3

[10] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K. Roy-Chowdhury, and Larry S. Davis. Learning temporal regularity in video sequences. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[11] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 7

[12] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017. 3, 4

[13] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection - a new baseline. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 6, 7

[14] William Lotter, Gabriel Kreiman, and David Cox. Unsupervised learning of visual structure using predictive generative networks. *arXiv preprint arXiv:1511.06380*, 2015. 2

[15] Weixin Luo, Wen Liu, and Shenghua Gao. Remembering history with convolutional lstm for anomaly detection. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 439–444. IEEE, 2017. 2

[16] Weixin Luo, Wen Liu, and Shenghua Gao. A revisit of sparse coding based anomaly detection in stacked rnn framework. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 5, 6

[17] Jefferson Ryan Medel and Andreas Savakis. Anomaly detection in video using predictive convolutional long short-term memory networks. *arXiv preprint arXiv:1612.0390*, 2016. 2

[18] Romero Morais, Vuong Le, Truyen Tran, Budhaditya Saha, Moussa Mansour, and Svetha Venkatesh. Learning regularity in skeleton trajectories for anomaly detection in videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 6, 7

[19] Mahdyar Ravanbakhsh, Moin Nabi, Enver Sangineto, Lucio Marcenaro, Carlo Regazzoni, and Nicu Sebe. Abnormal event detection in videos using generative adversarial nets. In *The IEEE International Conference on Image Processing (ICIP)*, 2017. 2

[20] Mahdyar Ravanbakhsh, Enver Sangineto, Moin Nabi, and Nicu Sebe. Training adversarial discriminators for cross-channel abnormal event detection in crowds, 2017. 2

[21] Mohammad Sabokrou, Mohsen Fayyaz, Mahmood Fathy, and Reinhard Klette. Deep-cascade: Cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes. *IEEE Transactions on Image Processing*, 26(4):1992–2004, 2017. 2

[22] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 7

[23] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *CVPR*, 2019. 3

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. 3

[25] Zhangyang Wang, Shiyu Chang, Jiayu Zhou, Meng Wang, and Thomas S. Huang. Learning a task-specific deep architecture for clustering. *Proceedings of the 2016 SIAM International Conference on Data Mining*, Jun 2016. 3

[26] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning (ICML)*, 2016. 3, 4

[27] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI Conference on Artificial Intelligence*, 2018. 3, 6, 7

[28] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Jul 2018. 3

[29] Yiru Zhao, Bing Deng, Chen Shen, Yao Liu, Hongtao Lu, and Xian-Sheng Hua. Spatio-temporal autoencoder for video anomaly detection. In *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 2017. 2