Weakly Supervised Anomaly Detection Inexact Supervision on Video Frames

Anastasia Drozdova 202202975, Andreas Jørgensen 202006409 & Marc Jensen 201907421

***Abstract*—Video anomaly detection (VAD) aims to locate unusual events or activities in videos. Weakly supervised video anomaly detection is difficult since the two classes are similar. Experiments with three models, MGFN, RTFM and S3R, were performed. All models use a feature extractor to represent the video before inputting it into the backbone of the model. Furthermore, all models use a separation loss on the feature magnitudes that the backbones output.**

In this report, the models were trained and tested on UCF-crime and XD-violence datasets. Additionally, a validation set was implemented for model selection and to find the generalization error. The models with model selection on the validation set yielded worse results, but have a better generalization ability.

***Index Terms*—Video anomaly detection, weakly supervised, anomaly detection, inexact supervision, magnitude-contrastive glance-and-focus network (MGFN), Self supervised sparse representation (S3R), robust temporal feature magnitude learning (RTFM)**

I. Introduction

ANOMALY DETECTION (AD) is the task of detecting abnormalities from normalities. This report performs Video anomaly detection VAD, where the goal is to classify frames that stand out from the normality. This is a difficult task because the setting determines if the event is abnormal or normal. An example of this could be that a fire in a fireplace is normal, but a fire in the bedroom is abnormal. The abnormalities investigated in this report are Burglary, Car Accident, Explosion, Arson, Shoplifting, etc. This task has rising importance because of the increasing amount of surveillance videos and the potential to be used in autonomous surveillance systems [1].

A simple approach to solving AD is using Unsupervised Learning, where vast numbers of normal frames are used to learn a representation of what is considered normal. Therefore this is also called one-class video anomaly detection oVAD [2]. However, these methods have shown a lacking ability to learn the discrimination between normal and abnormal. If the training data is not diverse, the model overfits because the training sample does not cover well the space of normal videos sufficiently. This results in new normal frames being predicted as abnormal. On the other hand, they often predict abnormal frames as normal because they have limitations in discriminating edge cases that are abnormalities that are similar to normal frames [3].

Supervised learning on frame-level predictions has enormous labeling costs. The reason is that standard cameras record 30 frames each second, resulting in a lot of data that must be labeled. Secondly, they are more complex to label than most computer vision tasks because of the similarities between normal and abnormal frames. An example of an easy annotation task is labeling if a picture contains a dog. A more challenging example that needs multiple frames is labeling if a person is abusing a dog (abnormal) or if a person is playing with a dog (normal).

A combination of unsupervised and supervised learning is weakly supervised learning. This is when some labels are used for learning, but the labels are either insufficient. The three sub-types of weakly supervised learning:

- **Incomplete supervision** where only part of the data is labeled. An example is financial transactions, where only part of the transactions is labeled due to the high labeling cost.
- **Inexact supervision** where the labels are not precise enough. An example is when coarse-grained labels are available, but the task is detecting frame-level anomalies.
- **Inaccurate supervision** where the labels are noisy. Example of this is computer-generated labels, which are less precise than human labels but still yield better performance than the fully unsupervised approach [4].

This report uses inexact supervision because the data sets used only have video-level labels of anomaly or not, and not the frame-level the goal is to predict. The S3R(2022) model uses an oVAD approach because it solves both task simulations [2]. The two other models used was MGFN(2022) [5] and RTFM(2022) [1]. All three models are state-of-the-art, achieving 2nd, 1st 5th on Paperswithcode for UCF-crime and 3rd, 4th, and 6th place for XD-violence. UCF-Crime consists of surveillance videos, whereas XD-violence consists of both Youtube videos and videos from movies.

The aim of this report is to make a fair comparison of the results of the three different models: RTFM [1], MGFN [5] and S3R [2].

The Robust Temporal Feature Magnitude learning (RTFM) [1] trains a feature magnitude learning function to recognize the positive instances effectively. RTFM adapts dilated convolutions and self-attention mechanisms to capture long- and short-range temporal dependencies. The model presented in [6] employs temporal context to increase the current time feature and enforces the samples of various categories to be more separable in the feature space to learn the temporal connection and feature discrimination better.

The Magnitude-Contrastive Glance-and-Focus Network (MGFN) [5] is built on top of the RTFM model [1]. However, it uses a Feature Amplification Mechanism and a Magnitude Contrastive Loss to enhance the distinctiveness of feature

magnitudes for detecting anomalies. The MGFN model achieves state-of-the-art on WSAD benchmarks UCF-Crime and XD-Violence. A more detailed description of the model is in the Model Framework.

The Self-Supervised Sparse Representation for Video Anomaly Detection (S3R) [2] is built on top of the RTFM model [1]. However, it also uses Dictionary Learning. This a self-training method to generate pseudo labels, and the model solves both oVAD and wVAD simultaneously. A more detailed description of the model is in the Model Framework.

II. Related work

Analyzing every frame of the video requires many hours of hard work because of the large quantity of training data in surveillance videos. Consequently, researchers choose to focus on settings where manual labeling is reduced or avoided entirely, which includes the one-class [7] or unsupervised setting to learn anomaly detection without annotation, as well as the weakly supervised, where just video-level annotations are provided.

Various works focus on the one-class setting. ConvLSTM network [8] is designed to learn the normal segments. Also, Auto-Encoder (AE) networks [9]–[11] are used to reconstruct features of normal frames.

Others employ the memory mechanism to learn the normal patterns and use meta-learning [12] to improve the model's generalization capability to the unseen normal scenarios [13]–[16].

Earlier work in the weakly supervised learning setting [17] uses a multiple instance learning strategy to locate video anomalies. Graph convolution networks were also used [18]; however, the generalization capability was insufficient.

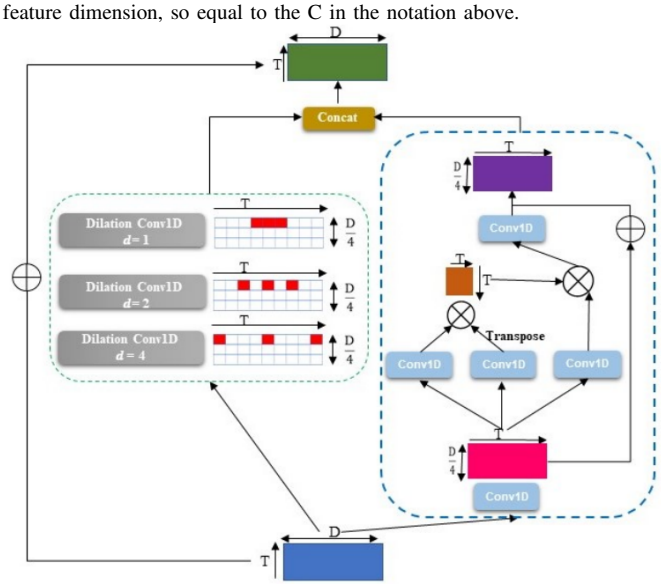
Other models include a Siamese network to learn the typical feature simulation [19] and clustering-based frameworks [20], [21].

More recent works introduce various network architectures to learn spatial-temporal features [1], [6], [22]–[24]. To enhance task-specific discriminative representations using video-level annotations, a multiple instance self-training framework (MIST) [23] was developed. MIST comprises a multiple instance pseudo label generator and a self-guided attention-boosted feature encoder to automatically focus on abnormal segments and extract task-specific representations.

Multi-Sequence Learning (MSL) method [24] introduces a hinge-based MSL ranking loss that uses a sequence composed of multiple clips as an optimization unit to reduce errors at the beginning of the training. A Transformer-based network learns video-level anomaly probability and clip-level anomaly scores. The authors also propose gradually reducing selected segments' lengths as a self-training strategy.

Weakly Supervised Anomaly Localization (WSAL) method [25] analyzes the temporal context and extracts semantic representations via high-order context encoding. The immediate semantics are derived directly from the segment representations and combined with the dynamic variations to get the anomaly scores. An approach for dealing with noise interference and the lack of localization guidance in anomaly detection is also presented.

Fig. 1. Backbone of the RTFM model from Supplemental pdf. The D is the feature dimension, so equal to the C in the notation above.



III. Model Framework

The implemented models are Robust Temporal Feature Magnitude learning (RTFM) [1], The Magnitude-Contrastive Glance-and-Focus Network (MGFN) [5] and Self-Supervised Sparse Representation for Video Anomaly Detection (S3R) [2].

Both the MGFN and S3R model is built on top of the RTFM and all three has the following framework[A-D]:

A. The Feature Extractor
B. The backbone of the model
C. The calculation of Feature Magnitude & Predictions
D. The losses that are used to learning.

A. *Feature Extractor*

The feature extractor used for all the training is the I3D RGB. I3D RGB has shown considerable benefits by pre-training on a large video dataset to extract and compress useful features from the data [26]. The feature extractor takes B untrimmed videos V, each video is denoted

$$V_i \in \mathbb{R}^{N_i \times H \times W \times 3} \tag{1}$$

where N_i denote the number of frames in video i and H, W is height and width. Each video is split evenly into T clips. The feature f of the t^{th} video clip in V_i is denoted $f^{i,t} \in P, C$. The feature map,

$$F = \{f^{i,t} \text{ where } i \in [1, B], t \in [1, T]\} \in \mathbb{R}^{B \times P \times T \times C} \tag{2}$$

and P is the number of crops in each clip and C is the feature dimension. [5] T = 32 for all models during training.

B. *Backbones*

1) RTFM Backbone: The backbone of the RTFM is a Multiscale Temporal Network (MTN). The MTN is built using two modules. The first module is a Pyramid of Dilated Convolutions (PDC), which is displayed in the left (green

Fig. 2. Backbone of the MGFN model [5]

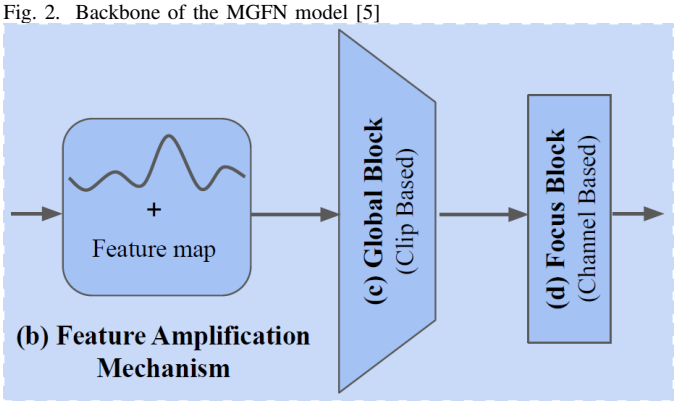
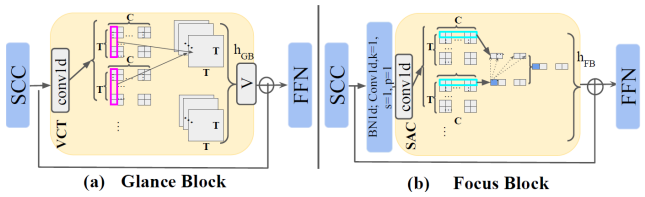


Fig. 3. Blocks of the MGFN model [5]



dashed) box on Fig. 1. PDC tries to learn local dependencies over different scales.

The second module is a self-attention network and is displayed in the right (blue dashed) box in Fig. 1. It tries to learn global dependencies between clips. The MTN takes the I3D features of the B videos each with P crops as input. The MTN outputs the temporal feature representation. This is the concatenation of the output from the PDC and self-attention module. [1] Furthermore, there is a skip connection, which ensures the original features are not forgotten.

2) MGFN Backbone: Following Fig. 2, the backbone of MGFN consist of a Feature Amplification Mechanism, Global Block and Focus Block.

The **Feature Amplification Mechanism (FAM)** is implemented to enhance the feature map F, by calculating the feature norm $M^{i,t}$ of $f^{i,t}$

$$M^{i,t} = \left(\sum_{c=1}^{C} \left| f^{i,t,c} \right|^2 \right)^{\frac{1}{2}} \in \mathbb{R}^{1\times1\times P\times1} \tag{3}$$

where c denotes the feature dimension index. Then the FAM derives the enhanced features $f_{FAM} = f^{i,t}_{FAM}$ by adding the 1D convolution modulated feature norm, $Conv1D(M^{i,t})$,

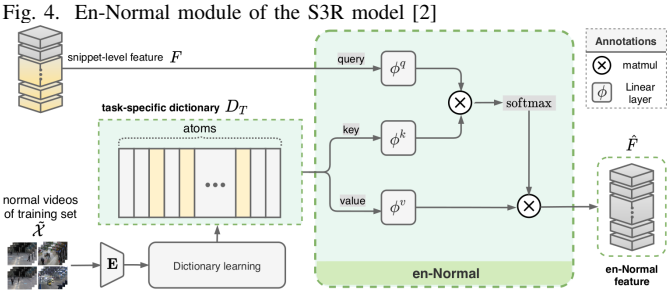
$$f^{i,t}_{FAM} = f^{i,t} + \alpha Conv1D(M^{i,t}) \in \mathbb{R}^{1\times1\times P\times C} \tag{4}$$

where α is a hyperparameter that controls the norm term [5].

There are two different blocks using the FAM, see fig. 3. The first step of the **Glance Block (GB)** is a Short-Cut Convolution (SCC) that reduces the dimension of the feature map to the output, $F_{SSC\_GB} \in \mathbb{R}^{B\times T\times P\times \frac{C}{32}}$ Then the reduced feature map is fed through the Video Clip-level Transformer (VCT) to learn the global correlation among clips. Specifically an attention map $A \in \mathbb{R}^{1\times T\times T\times P}$ to correlate the temporal clips,

$$A^{i,t_1,t_2} = \sum_{c=1}^{C} Q\left(F^{i,t_1,c}_{scc\_GB}\right) K\left(F^{i,t_2,c}_{scc\_GB}\right) \tag{5}$$

Fig. 4. En-Normal module of the S3R model [2]



where $t_1, t_2 \in [1, T]$ and Q, K are 1D Query and key convolutions respectively. Then softmax normalization is used to generate $a \in \mathbb{R}^{1\times T\times T\times P}$

$$a^{i,t_1,t_2} = \frac{e^{A^{i,t_1,t_2}}}{\sum_{t_2=1}^{T} e^{A^{i,t_1,t_2}}} \tag{6}$$

The output is the weighted average of all clips in the video

$$F^{i,t_1,c}_{att\_GB} = \sum_{t_2=1}^{T} a^{i,t_1,t_2} V(F^{i,t_2,c}_{scc\_GB}) \tag{7}$$

where V is the 1D value convolution of the transformer and $F^{i,t_1,c}_{att\_GB} \in \mathbb{R}^{1\times T\times P\times \frac{C}{32}}$. A Feed-Forward Network (FFN) finishes off the Glance Block and outputs a feature map F_{GB}. [5]

The first step of the **Focus Block (FB)** is to increase the input dimensions F_{GB} to $\frac{C}{16}$ dimensions, which are used in the SSC, to make a feature map F_{scc_FB}. The second step is a Self-Attentional Convolution (SAC),

$$F_{sac\_FB} = F_{scc\_FB} \circledast F_{scc\_FB} \in \mathbb{R}^{B\times T\times P\times \frac{C}{16}} \tag{8}$$

where

$$F^{i,t,k_1}_{sac\_FB} = \sum_{k_1,k_2=0}^{\frac{C}{16}} F^{i,t,k_1}_{scc\_FB} F^{i,t,k_2}_{scc\_FB} \in \mathbb{R}^{1\times1\times P\times1} \tag{9}$$

Finally, after a Feed Forward Network the feature map F_{FB} is outputted [5].

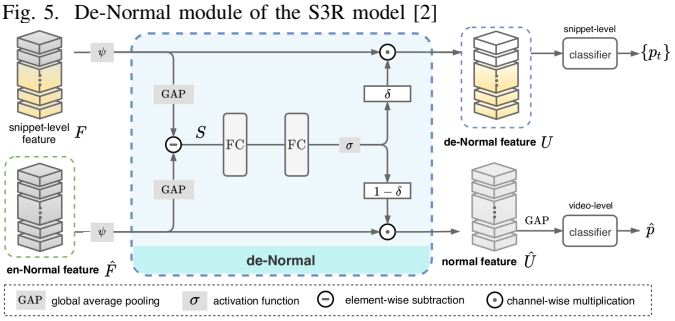
3) S3R Backbone: S3R [2] utilizes dictionary learning: taking the normal videos of the train set to learn its corresponding dictionary D of N atoms. To do so, the following expression is optimized:

$$\text{argmin}_{D,w_t} \sum_{x\in X} \sum_{t=1}^{T} (\|f_t - Dw_t\|^2 + \lambda \|w_t\|_0),$$

where X is the train set, f_t clip representation, $D \in \mathbb{R}^{C\times N}$ is the resulting VAD dictionary, and $w_t \in \mathbb{R}^N$ is the coefficient vector constrained by the sparsity prior [2].

En-Normal module (Figure 4 is a dictionary-based attention module. It better correlates the clip-level feature F and the learned task-specific dictionary D_T, leading to the corresponding normal-event feature $\hat{F}$ Since DT is assumed to span the feature space of all normal-event patterns, the attention mechanism is used to reweigh clip-level input feature F with respect to D_T to obtain its reconstructed normal-event feature

Fig. 5. De-Normal module of the S3R model [2]



$\hat{F}$. Linear embeddings ϕ to project F and D_T. The attention is computed in the embedding space and defined as

$$\hat{F} = \text{softmax}\left(\phi^q(F)\phi^k(D_T)^\intercal\right)\phi^v(D_T)\,,$$

where phi ϕ^q, ϕ^k, and ϕ^v separately represent linear functions to derive query, key, and value embeddings [2].

De-Normal module (Figure 5) aims to filter out normal-event patterns, so the remaining features can indicate anomalous regions. It takes the initial and reconstructed features, F and $\hat{F}$.

First, MTN is applied to both F and $\hat{F}$ to explore the temporal dependency and retrieve the enhanced features as $\psi(F) \in \mathbb{R}^{T\times C}$ and $\psi(\hat{F}) \in \mathbb{R}^{T\times C}$, where ψ denotes the MTN operation. Next, global average pooling (denoted as $g()$) is applied collect global events, where each channel includes normal or anomalous semantics. Get the channel-wise difference:

$$S = g(\psi(F)) - g(\psi(\hat{F}))\,,$$

To further keep the anomalous-event channels of cross-video semantics and simultaneous depress normal-event channels, SENet-style operations are employed to explore the channel-wise relationship and derive the corresponding channel scales for depressing normal event within the input video representation as

$$\delta = \sigma(\text{MLP}(S))\,,$$

$$U = \delta \odot F, \quad \hat{U} = (1-\delta)\odot\hat{F}\,.$$

where MLP consists of two fully connected layers, σ denotes the sigmoid activation, and $\odot$ means the channel-wise multiplication. The scale vector $\delta \in \mathbb{R}^c$ indicates the channel-level weights to keep anomalous events, while $1`-\delta$ denotes channel weights for focusing on normal events [2].

C. Feature Magnitudes & predictions

The output features from a model are denoted as $f^{i,t,c}_{model} \in \mathbb{R}^{BP\times T\times C}$.

1) Scores from the feature predictions: The features are fed through a fully connected network. This network use a sigmoid function (at the end) to get an anomaly score for each clip in all of feature crops. Then a mean function is taken over the crops to get the average anomaly score for a clip in each video. This output is denoted $s^{i,t} \in \mathbb{R}^{B\times T\times 1}$ [5].

2) Feature magnitudes & Top-k: The l_2 norm is used to calculate the features-magnitudes

$$M^{i,t} = \left(\sum_{c=1}^{C}(f^{i,t,c}_{model})^2\right)^{\frac{1}{2}} \in \mathbb{R}^{BP\times T}. \tag{10}$$

The biggest top-k feature magnitudes are extracted in the subset $M^{i.t}_k \in \mathbb{R}^{BP\times k}$. This set is split into one for the normal feature magnitudes and one for the abnormal:

$$M^{i,j}_n \in \mathbb{R}^{BP/2\times k}, \quad M^{i,j}_a \in \mathbb{R}^{BP/2\times k}. \tag{11}$$

The corresponding index of $M^{i,t}_k$ is found. Firstly, these indexes are used to extract a subset $\Omega \in \mathbb{R}^{BP\times k\times C}$ of the k features from the $f^{i,j,c}_{model}$. This set of features is then split into two sets, one for the normals features and one for the abnormal features:

$$\Omega_N \in \mathbb{R}^{BP/2\times k\times C} \tag{12}$$

$$\Omega_A \in \mathbb{R}^{BP/2\times k\times C} \tag{13}$$

Secondly, these indexes are used to subset the top-k predicted scores from $s^{i,t}$ and this subset is denoted $s^{i,t}_k \in \mathbb{R}^{B\times 3\times 1}$. These are used to calculate the video level predictions $\hat{p}_i = \frac{1}{k}\sum_{t=1}^{K} s^{i,t}_k \in \mathbb{R}^B$. The reason behind this is, that through training the network is optimized to minimize the 3 most abnormal clips from the normal videos and maximize the three most abnormal clips in the abnormal video [5].

D. Losses

The losses used for our learning are taken from the respective papers Github repos. All three models use end-to-end training with a multi-task objective/loss [2]:

$$L = L_{cls} + \alpha L_{sep} + \lambda_1 L_{ts} + \lambda_2 L_{sp} \tag{14}$$

where L_{cls} denoted the classification loss and L_{sep} separates the two classes of videos. All models used a sparsity loss $L_{sp} = \sum_{j=1}^{T} s^{i,j}_a$ to impose a prior that abnormal events are rare in each abnormal video. The models also use a temporal smoothness loss $L_{ts} = \sum_{j=1}^{T}\left(s^{i,j}_a - s^{i,j+1}_a\right)^2$ that enforces similar predictions for neighbouring clips [5]. The binary cross entropy is used as the classification loss $L_{cls} = BCE(\hat{p}_i, y_i)$. This loss measures the difference between the predicted video class (abnormal/normal) and the known targets y_i.

1) RTFM Losses: The separation loss L_{sep} of the RTFM model is split between a loss on the abnormal features and a loss on the normal features,

$$L_{sep} = \frac{2}{BP}\sum_{b=1}^{BP/2}\left(L_{abn} + L_{nor}\right)^2 \in \mathbb{R} \tag{15}$$

$$\text{where } L_{abn} = \left|Margin - \left(\sum_{c=1}^{C}\left(\frac{1}{k}\sum_{i=1}^{k}\Omega_A\right)^2\right)^{\frac{1}{2}}\right| \in \mathbb{R}^{BP/2}, \tag{16}$$

$$Margin = 100 \text{ and } L_{nor} = \sum_{c=1}^{C}\left(\frac{1}{k}\sum_{i=1}^{k}\Omega_A\right)^2\Big)^{\frac{1}{2}} \in \mathbb{R}^{BP/2}. \tag{17}$$

2) MGFN Losses: The MGFN model uses a Magnitude Contrastive loss to better encourage feature separability

$$L_{sep} = \frac{4}{BP}\sum_{i,j=0}^{\frac{BP}{4}}(1-l)\big(D(M_n^{i,t}, M_n^{i,t})\big)^2 \tag{18}$$

$$+ \frac{4}{BP}\sum_{i,j=0}^{\frac{BP}{4}}(1-l)\big(D(M_a^{i,t}, M_a^{i,t})\big)^2 \tag{19}$$

$$+ \frac{2}{BP}\sum_{i,j=0}^{\frac{BP}{2}} l\Big(max\big(0, Margin - D(M_n^{i,t}, M_a^{j,t})\big)\Big)^2 \tag{20}$$

where l is an indicator function and $l = 0$ when the two clips come from the same class, and $l = 1$ if one clip is normal and the other one abnormal. $M_n^{i,t}$ & $M_a^{i,t}$ are defined in formula (11). For the inter-class losses, the features are split in half. The indexes i and j iterate through each of these halves respectively. D is the Euclidean distance function. The weight balancing this loss-term is denoted λ_3 instead of α from the formula (14). L_{sep} groups the normal classes together in formula (18) and the abnormal classes together in (19). Formula (20) separates the feature magnitude distance of the abnormal and normal classes.

3) S3R Lossses: The S3R model uses the same loss L_{sep}as the RTFM model (15) to separate the abnormal and normal feature magnitudes. The classification loss is different than the other two L_{cls} as BCEs are used for both classifying the video predictions and clip-level predictions

$$L_{cls} = BCE\big(s_k^{i,t}, y_i\big) + BCE\big(\hat{p}, y_i\big), \tag{21}$$

where $s_k^{i,t}$ is the top-k clip level probabilities and $\hat{p}$ is the video level [2].

IV. Implementation

A. Parameter settings for the networks

1) MGFN: The default parameters for this network are: margin = 200 (in the MGFN loss), k = 3 (amount of top-k features). To train the network the Adam optimizer was used. This was with learning rate 0.001 and a weight decay of 0.0005. The batch_size used was 8 random normal and 8 random anomalous videos. Other parameters such as lambda_1 = 1 (weight of smoothness loss), lambda_2 = 1 (weight of the sparsity loss), lambda_3 = 0.001 (weight of the MGFN_loss) and dropout of 70% are used.

2) RTFM: The default parameters for this network are: margin ("m") is set to 100, k = 3 (amount of top-k features). The Adam optimizer has been used with learning rate 0.0001 and a weight decay of 0.0005. The batch_size used was 32 random normal and 32 random anomalous videos. Other parameters such as lambda_1 = 8e-4 (weight of smoothness loss), lambda_2 = 8e-3 (weight of the sparsity loss), alpha = 0.0001 (weight of the RTFM_loss) and dropout of 70% are used.

3) S3R: The default parameters for this network are: The Adam optimizer has been used with learning rate 0.001 and a weight decay of 0.005. The batch_size used was 32 random normal and 32 random anomalous videos. The model is RTFM-based, so inherited parameters are the same: lambda_1 = 8e-4, lambda_2 = 8e-3, alpha = 0.0001. Dropout rate of 70%. Dictionary size is $N/2$, where N is the number of normal train videos, "omp" algorithm with 100 iterations from sklearn.decomposition.MiniBatchDictionaryLearning is used.

B. Improvements of the code

In the code a change to the mini-batch sampling process was implemented. This was to ensure that the mini-batches are taken at random which is important for the learning of the network [27]. An update to the smoothness loss function was made, so that the smoothness loss was calculated per video in the mini-batch.

Working with this UCF-crime data set it was found that the ground truth (gt) file uploaded on the MGFN repo was wrong. This mistake was brought to the authors attention in a github issue. Using the wrong gt file an AUC-score of 86.97% could be reached with the pretrained MGFN model (the result reported in their paper [5]). In this work a new correct gt-file was created. This file was then used for testing the different models.

V. Experiments

A. Test, Train and Validation split of the data

All three papers investigated use a test/train split of the data. The authors use the test data to validate and select the best model. The papers use supervised metrics (such as AUC) in the training of the networks. These are calculated using the frame-level annotation that is only available in the test set. In a usual machine learning setting a validation data set is created from the train data. The model is then trained on the train set and the generalization error/loss is reported using the validation set [27]. The loss from the validation data set is used as an early stopping criterion while training the model. This returns the parameter of the model with the lowest validation loss [27]. Using the validation data set, it is ensured that a weakly supervision setting is satisfied.

B. Data

Only video-level labels are given for each video in the training-process. E.g. this video contains an anomaly. In the test sets, the start and the end frame of each anomaly are given. The extracted I3D features can be downloaded from either the data provider or from one of the paper's GitHub pages. The type of anomalies is given in the datasets but unused in the models tested. Since the models only do binary classification and not a multi-classification of the anomalies.

1) UCF-crime: This data set contains 1900 untrimmed real-world surveillance videos which has a total duration of 128 hours. An abnormal video contains one of the following anomalies, Abuse, Arrest, Arson, Assault, Road Accident, Burglary, Explosion, Fighting, Robbery, Shooting, Stealing,

Shoplifting, and Vandalism. The data set is split into a train- and a test set of 1610 and 290 videos respectively. In the train set, a total of 810 videos contain an anomaly, and 800 videos contain normal scenarios. A normal scenario could be a surveillance video from inside a mall. The test set is split into 140 anomaly and 150 normal videos [28]. The test set is highly unbalanced, since 7.5% of the frames are anomalous. In this report the features for the UCF-crime data set was downloaded from this link. In this report, a 10% validation data set was created from the UCF train set.

2) XD-violence: The XD-violence data set contains 4754 untrimmed videos which has a total duration of 217 hours. The videos were collected from movie clips or YouTube videos. The different anomalies present are: Abuse, car accident, explosion, fighting, riot and shooting. In this data set multiple anomalies can happen in the same video. The data set is split into a train set of 3954 videos and a test set of 800 videos. The train set contain 2049 normal (non-violence) videos and 1905 videos where violence (anomalies) occur. In the test set the split is 500 videos with violence and 300 videos with non-violence [29]. The test set is unbalanced since 23.1% of the frames are anomalous. The features of the XD-violence data set was downloaded from the official data-set site [29]. In this report a 20% validation data set was created from the XD train set.

C. Hyper-parameter turning

Firstly identical training setups as the papers were used to recreate similar results. The setups trained the model using all train data and validated the model on the test set. It was found that all networks had a tendency to over-fit to the train set. This was clear since the validation/test loss did not decrease while the train loss decreased. Therefore, it was investigated if different types of regularization could minimize the over-fitting behavior. The different regularization parameters tested were: Weight decay, dropout rate, number of modules/blocks and different weight to the loss functions [27], [30]. A grid search for hyper-parameters was performed around the parameters from the papers.

A minor amount of epochs was used to optimize the hyper-parameters, because only a few are sufficient to determine if the parameters are good or not [30]. About 100 epochs were used in the optimization for each combination of parameters. In the papers, 1000 epochs were used when training (though with an early stopping scheme).

D. Performance measures used for testing

The goal of the different models is to predict high anomaly scores for frames where an anomaly is present. But a good model should at the same time predict low anomaly scores for normal frames. It should be noted that there is generally a great imbalance between the number of anomalous frames and normal frames in the different test sets (see subsection V-B). Generally in this domain of WSAD the following metrics are used AUC, precision recall (PR) and average precision (AP). AUC is the only metric used on the UCF-crime data set and AP is the only metric used on the XD-violence when comparing models. However, these metrics can be influenced by imbalance. Therefore, other measures such as precision, recall and accuracy are implemented. In this case, an anomaly is flagged when the predicted anomaly score gets above a threshold of 0.5. The accuracy measures do not handle imbalance that well. The measures F1 score that weight precision and recall equally is used. The macro F1 (F1M) score that weights the F1 score of the different classes is also used.

VI. Results

The following sections summarize the results of the models on the different data sets. The models with the extension _pre is the highest performing model trained by the authors of the papers. These models are only used for testing and no training. The _cheat tag indicates that our models have been trained similarly to the authors. This is done by validating during training on the test set and selecting the model with the highest AUC or PR overall. The _val indicates that a subset of the train set has been used for validating the model during training. The model selection criterion is the lowest validation loss obtained. In the training it was observed, that the model would tend to over-fit the train data before the 100 epochs (please refer to this: link as an example). Therefore, the hyper-parameter search of 100 epochs where used as actual training of the networks (but still with an early stopping scheme implemented).

A. Results on UCF-Crime

Table I summarizes the results of the best-performing models.

1) Results of the MGFN model: From the hyper-parameter search the best _cheat model found was with the parameters from the section IV except with a weight decay of 0.005. The model was found at epoch 95, and the training can be seen in this link. These hyper-parameters were unfortunately not good in the sense that the test-loss was very unstable. It was observed that different hyper-parameter could result in good performing models. Though with bad loss curves - the loss on the test set seemed very unstable. For run details [MGFN-38 - MGFN-58].

The validation model was trained with the parameters from the implementation section. The training can be seen in this link. It was found that the model with the lowest validation loss occurred at epoch 50.

From the tables, it can be seen that the pretrained MGFN model achieves the best results accordingly to the AUC score. With this the model is able to recall 91.4% of the frames containing anomalies. This is though with the low precision of 16.6%. The MGFN_cheat achieved 81.6% in the AUC score but where able to get a higher average precision and precision recall of 25.1%. The validation model scores 80.7%, which is actually only 2.8 percent-points from the pretrained model in terms of AUC score. This model could though only recall 75.8% of the anomalies but with a higher precision of 18%. Looking at the F1 scores - the validation model does better than the other models. Comparing the loss of the models, it can be seen that the _val model achieved a far lower loss - both

TABLE I
RESULTS ON UCF-CRIME
AP IS AN ABBREVIATION OF "AVERAGE PRECISION", F1M IS THE "F1 MACRO" SCORE AND PR IS THE AREA UNDER THE "PRECISION RECALL" CURVE.

Network		AP(%)	AUC(%)	F1(%)	F1M(%)	PR(%)	Accuracy(%)	Recall(%)	Precision(%)	Test-loss	Train-loss
MGFN	pre	23.5	83.5	28.1	52.3	23.5	64.0	**91.4**	16.6	5.501	5.339
	cheat	25.1	81.6	28.5	55.0	25.1	70.6	77.4	17.4	2.353	.898
	val	20.2	80.7	29.1	55.8	20.2	72.0	75.8	18.0	1.252	.249
RTFM	cheat	21.2	82.0	30.9	56.9	**31.6**	72.5	81.1	**19.1**	.430	.021
	val	21.6	80.3	29.8	57.1	29.3	**74.6**	71.3	18.9	.042	.014
S3R	pre	**31.5**	**85.9**	**31.3**	56.2	31.5	70.4	89.2	18.9	-	-
	cheat	27.4	83.7	29.7	54.9	27.4	69.2	85.7	17.9	-	-
	val	20.3	80.5	29.7	**56.3**	32.8	72.6	76.4	18.4	-	-

on the test- and train set. This is logical, since the validation model is selected for its low loss, whereas the MGFN_pre and _cheat are chosen based on the best AUC.

2) Results of the RTFM model: The pretrained RTFM model was not available, and therefore only a cheat and validation model is trained on the data set. The best cheat model was found using the hyper-parameters from the section IV. For run details [RTFMUC-22 - RTFMUC-37]. The training details of the validation run with the original parameters is available here.

According to the authors of the RTFM paper, the RTFM model could achieve an AUC score of 84.30 on the UCF-Crime data with I3D RGB features [1]. Our best model achieved 82.0%. The cheat model beats the validation model in all metrics except for AP, F1M and accuracy. The validation model achieves way lower loss.

3) Results of the S3R model: The authors provide a checkpoint file for their pretrained model, which achieves AUC of 85.99. Training the model using parameters from the provided config file achieved slightly worse results.

There was an error while generating lists, so the hyperparameter search achieved nothing, as the model could not train. After fixing the lists, there was not enough time to perform extensive hyperparameter optimization.

Both cheat and val models show similar results across all metrics, then the pretrained model. The parameters provided by the authors did not produce the same model.

B. Results on XD-Violence

Table II summarizes the results of the different models trained on the XD-violence data set. There are some minor differences between the reported results in the table and the results achieved during training (on Neptune). This is because minor corrections in the ground truth file were conducted after training. The results in the table are calculated using the correct ground truth file.

1) Results of the MGFN model: Three different cheat models stood out during the hyper-parameter optimization. These models were of special interest since they achieved high AP scores. The parameters of the three models have not converged since the models peaked early in the training. The runs and epochs are respectively: MGFN_cheat_A at epoch 8, MGFN_cheat_B at epoch 1 and MGFN_cheat_C at epoch 7. These models have been trained with very different hyper-parameters, but still seem to get high AP scores. The model "MGFN_cheat_A" actually gets an AP score corresponding to a third place on "Papers with code" scoreboard [XD-violence]. The highest anomalous score on a frame that the model MGFN_cheat_A predicted was a score of 0.138. This means that all predicted anomaly scores are below the threshold of 0.5 resulting in a recall and precision of 0. It is also clear that these trained models are not on par with the pre-trained MGFN model. This can be seen by the lower F-scores and recall.

All model run details [MGFNXD-1 - MGFNXD-122]. It is generally a tendency in these model-runs that the AP score peaks in the begining and the decay as the network progress in its training. The _cheat model has been selected from this run: MGFNXD-59. It is noticeable that this model has more or less been chosen by us. This is because the early high AP-score would make the early stopping criterion select that model. The issue is that the early model does not perform as well on other measures compared to the one selected (at epoch 60). The changes to the hyper-parameteres are: Drop_out of 40 %, learning rate of 0.001 and a smaller network configuration is used.

2) Results of the RTFM model: The hyper-parameter optimization is available here in: [RTFM-17 - RTFM-31]. The best hyper-parameters where achieved from run 18, RTFM-18. This was with the hyper-parameters from the section IV except that a weight decay of 0.005 was used.

According to the RTFM paper, the RTFM model gets an average precision of 77.81 [1]. The best model found during our training achieved an average precision of 77.4. This model has a recall of 80.3% and a precision of 54.7%.

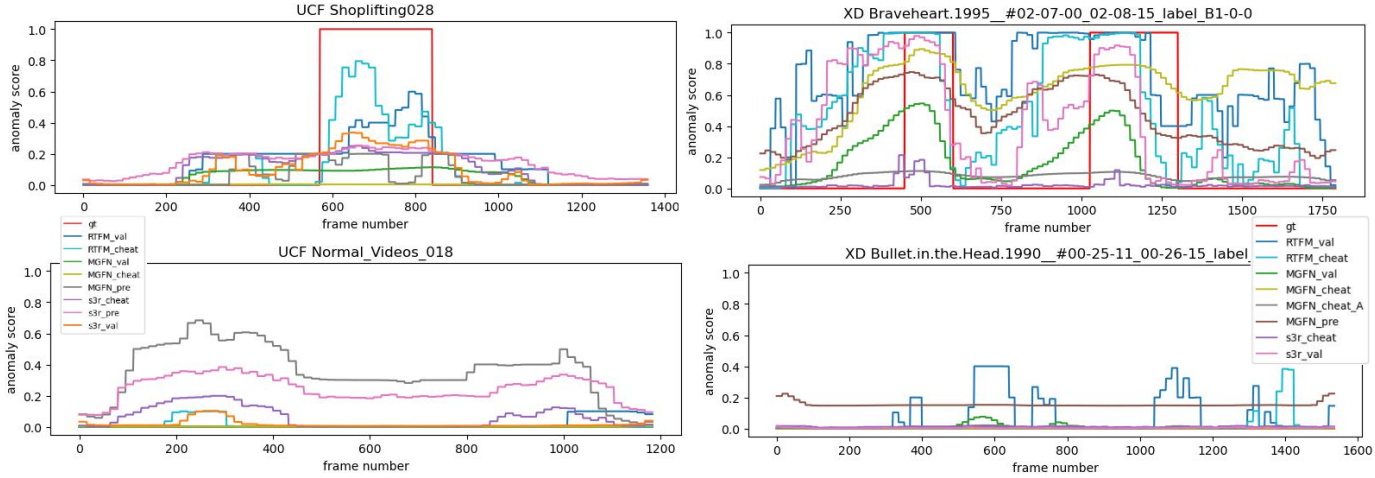
The validation model was trained with the parameters from section IV. The validation model got worse results compared to the cheat model and the results of the RTFM paper. The validation model got a smaller AP score of 65.4% and a smaller precision score of 53.2%, but a better recall of 88.9% on the anomalous frames. Check the run of this model in this link.

3) Results of the S3R model: The authors have not provided pretrained model or training configs. From the results in the paper, the model achieved 80.26 AP. Both cheat and val have

TABLE II
RESULTS ON XD-VIOLENCE.

Network		AP(%)	AUC(%)	F1(%)	F1M(%)	PR(%)	Accuracy(%)	Recall(%)	Precision(%)	Test-loss	Train-loss
MGFN	pre	80.1	**92.9**	71.9	81.3	80.1	**86.1**	77.0	67.4	4.033	5.249
	cheat	71.5	90.2	65.9	74.7	71.5	77.7	**93.5**	50.9	1.807	1.074
	cheat_A	**80.6**	**92.9**	0	43.4	**80.6**	76.9	0	0	0.833	1.029
	cheat_B	78.9	92.3	25.4	56.9	78.9	80.1	14.7	94.5	1.552	1.968
	cheat_C	79.0	92.4	22.0	55.2	79.0	79.6	12.4	**96.5**	13.139	14.008
	val	72.5	89.0	27.7	58.2	72.5	80.3	16.4	91.2	1.367	1.543
RTFM	cheat	77.4	90.8	70.7	80.2	77.5	84.6	80.3	63.2	.022	.026
	val	65.4	88.9	66.6	75.8	67.4	79.3	88.9	53.2	.016	.009
S3R	cheat	77.7	91.2	**72.1**	**81.4**	77.7	86.0	77.9	66.9	-	-
	val	74.5	92.4	71.9	80.2	74.5	83.7	89.8	59.9	-	-

Fig. 6. Qualitative figure showing the frame level predictions for each model, on a normal and an abnormal video on each data set.



a lower score. Metrics are similar, but cheat model has higher precision, while val model got higher recall.

4) Prediction of the different models: In the figure 6 the different predicted anomaly scores from the models are displayed. In the left column, the predictions on UCF-crime videos are visualized. The top (left) figure shows a shoplifting anomaly, and the bottom (left) figure shows a video containing no anomaly. In the right column, the predictions on XD-violence videos are visualized. In the top (right) row a video with two anomalies are plotted. In the bottom (right) row anomaly scores are displayed for a video containing no anomalies. These plots show that all models make inaccurate predictions at times. These plots also display that the models trained on XD-violence are more confident, by predicting an anomaly score close to 1 or 0.

C. Discussion

The objective of a good anomaly detection model is finding as many anomaly frames simultaneously with getting a high precision (low false alarm rate). It was found that the different variations of the models were all capable of recalling over 70% of the anomalies in the UCF-data. Contrary, the precision is minimal in the range [16.6 - 19.1] (please refer to the table I). None of the implemented methods seems to stand out. In this case, finding the anomalies might have a too large weight in the objective. A lower k-value in the top-k function or more weight to the sparsity loss might decrease the false alarm rate. However, these parameters are difficult to adjust in a fully weakly supervised setting, because the ground truth file is needed to calculate the precision and recall.

MGFN_cheat[A-C] achieved state-of-the-art AP performance on the XD-violence data set. At the same time, MGFN_cheat[A-C] perform poorly on the other measures when compared to the other models in the table II. This shows that relying on one measure to judge the performance of a model is inadequate. With the exception of the MGFN_cheat[A-C] and MGFN_val, a good combination of recall and precision is achieved. This is with a recall in the 77-90% area and precision of 53-68%. No model outperforms the remaining for all measures. Generally, the MGFN model was difficult to train on the XD-violence data set. Often the models trained would get high AP score in the beginning of the training. The AP score would decrease as other scores increased. The validation loss would not seem to decreasing much during the training, while the training loss would decrease. This kind of behavior indicates that the MGFN model does some overfitting. The RTFM model would not show this type of behavior to the same degree. A reason for this difference might be the sizes of the models. The MGFN model contains nearly 28.7 million parameters, while the RTFM model contains about

24.7 million parameters (when trained on the UCF-crime data set).

The models can be compared between the UCF-Crime and XD-violence. The models trained on XD-violence perform better with regards to the metrics AP, PR, precision, accuracy and F1-scores. This is likely because of the differences in the data sets. Firstly, there is more data in the XD-violence (more videos and longer duration) and that is the reason for better learning. Secondly, video sources can make a difference. Thirdly, fewer anomaly classes exist in the XD-violence data set. This means that the models need to generalize to less anomalous scenarios. Fourthly, the similarities between the anomalies in the train and test set. For instance, the UCF-crime data anomaly class: "abuse" contain only abuses of animals in the test set. In the UCF-crime training set the abuse videos contain only abuse on humans. If the models tested do not understand this behavior, or haven't seen normal behavior around animals, then the model will perform less well in this anomaly class. These differences indicate that if more data was gathered a better-performing model could be achieved. More data also helps the models avoid over-fitting as well, which all the trained models tend to do.

The idea behind having different representations of anomalies (in the same class) between train and test e.g. human abuse and animal abuse is to test a model's generalization ability. A good anomaly detection model would be able to detect the new abnormalities, even though the model has not seen this type of anomaly in the training phase. The model should learn the behavior using the separation loss. So that even though the model has not seen animal abuse directly, it should know how humans behave with animals in a normal setting. From this knowledge, it should then be able to detect that something is wrong with the behavior of the human and/or animal, and therefore classify the frames as an anomaly. The _cheat models do not perform as well compared to the models from the papers(_pre). Furthermore, _val models perform worst of all. The most prominent reason is likely the selection criteria. The _pre and _cheat models are selected based directly on the performance on the test set. Another reason that might cause a slight decrease in performance is that the _val models are trained on fewer videos. This is because a part of the train set was taken out for the validation set. Another reason might also be, that the loss is not necessarily directly linked to good AUC/AP performance.

VII. Conclusion

In this report, a comparison between the models MGFN, RTFM, and S3R has been conducted. This has been done in the most fair way possible, by calculating the same measures for the different models on the same data sets. In this report, the data sets UCF-crime and XD-violence have been implemented for training and testing the models. Additional validation sets have been created from the data sets (please refer to the section IV). From the results of the models, no model performed the best for all measured (please refer to section VI). It was found that there was a payoff between the measures used, so if a model had a high performance on one measure, it would lack on another.

It was also found that using multiple metrics for model comparison is more informative. In this paper, a very extreme case has been provided. In the hyper-optimization, the MGFN_cheat_A model (which was trained on the XD-violence) was found. This model would perform similarly to the SOTA models on the "Papers with code" scoreboard [XD-violence]. It was evident, that this model lacked performance in other areas (please refer to section VI for more information). During the model comparisons, it is essential to consider the model's generalization abilities. The (_val) models trained with validation performed worse than the ones with validation on the test set. The issue found with using the test set for validation and model selection is that this removes the reliability of the general results. It is doubtful that the (_pre and _cheat) models will perform as well in the application of anomaly detection scenarios.

To further develop these models, multiclass predictions could be implemented. This would make sense in a real-world scenario so that the correct emergency service can be alerted. E.g., if the model identifies a fire, it could call the fire department. Another scenario would be if the model detected violence, then it could alert the police department.

References

[1] Y. Tian, G. Pang, Y. Chen, R. Singh, J. W. Verjans, and G. Carneiro, "Weakly-supervised video anomaly detection with robust temporal feature magnitude learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 4975–4986.

[2] J.-C. Wu, H.-Y. Hsieh, D.-J. Chen, C.-S. Fuh, and T.-L. Liu, "Self-supervised sparse representation for video anomaly detection," in *ECCV*, 2022.

[3] M. Z. Z. et. al., "Cvpr2020 talk - cleaning labels noise with clusters for minimally supervised anomaly detection," 2020. [Online]. Available: https://www.youtube.com/watch?v=nJJbueHVot8

[4] M. Jiang, C. Hou, A. Zheng, X. Hu, S. Han, H. Huang, X. He, P. S. Yu, and Y. Zhao, "Weakly supervised anomaly detection: A survey," 2023. [Online]. Available: https://arxiv.org/abs/2302.04549

[5] Y. Chen, Z. Liu, B. Zhang, W. Fok, X. Qi, and Y.-C. Wu, "Mgfn: Magnitude-contrastive glance-and-focus network for weakly-supervised video anomaly detection," *arXiv preprint arXiv:2211.15098*, 2022.

[6] P. Wu and J. Liu, "Learning causal temporal relation and feature discrimination for anomaly detection," *IEEE Transactions on Image Processing*, vol. 30, pp. 3513–3527, 2021.

[7] M. Z. Zaheer, A. Mahmood, M. H. Khan, M. Segu, F. Yu, and S.-I. Lee, "Generative cooperative learning for unsupervised video anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 744–14 754.

[8] W. Luo, W. Liu, and S. Gao, "Remembering history with convolutional lstm for anomaly detection," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, 2017, pp. 439–444.

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

[10] R. T. Ionescu, F. S. Khan, M.-I. Georgescu, and L. Shao, "Object-centric auto-encoders and dummy anomalies for abnormal event detection in video," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7834–7843, 2018.

[11] Y. Chang, Z. Tu, W. Xie, and J. Yuan, "Clustering driven deep autoencoder for video anomaly detection," in *European Conference on Computer Vision*, 2020.

[12] Y. Lu, F. Yu, M. K. K. Reddy, and Y. Wang, "Few-shot scene-adaptive anomaly detection," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer, 2020, pp. 125–141.

[13] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1705–1714.

[14] H. Park, J. Noh, and B. Ham, "Learning memory-guided normality for anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 14 372–14 381.

[15] Z. Liu, Y. Nie, C. Long, Q. Zhang, and G. Li, "A hybrid video anomaly detection framework via memory-augmented flow reconstruction and flow-guided frame prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 588–13 597.

[16] R. Cai, H. Zhang, W. Liu, S. Gao, and Z. Hao, "Appearance-motion memory consistency network for video anomaly detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 2, 2021, pp. 938–946.

[17] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6479–6488.

[18] J.-X. Zhong, N. Li, W. Kong, S. Liu, T. H. Li, and G. Li, "Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1237–1246.

[19] B. Ramachandra, M. Jones, and R. Vatsavai, "Learning a distance function with a siamese network to localize anomalies in videos," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 2598–2607.

[20] B. Wan, Y. Fang, X. Xia, and J. Mei, "Weakly supervised video anomaly detection via center-guided discriminative learning," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2020, pp. 1–6.

[21] M. Z. Zaheer, A. Mahmood, M. Astrid, and S.-I. Lee, "Claws: Clustering assisted weakly supervised learning with normalcy suppression for anomalous event detection," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*. Springer, 2020, pp. 358–376.

[22] J. Wu, W. Zhang, G. Li, W. Wu, X. Tan, Y. Li, E. Ding, and L. Lin, "Weakly-supervised spatio-temporal anomaly detection in surveillance video," *arXiv preprint arXiv:2108.03825*, 2021.

[23] J.-C. Feng, F.-T. Hong, and W.-S. Zheng, "Mist: Multiple instance self-training framework for video anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 14 009–14 018.

[24] S. Li, F. Liu, and L. Jiao, "Self-training multi-sequence learning with transformer for weakly supervised video anomaly detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, 2022, pp. 1395–1403.

[25] H. Lv, C. Zhou, Z. Cui, C. Xu, Y. Li, and J. Yang, "Localizing anomalies from weakly-labeled videos," *IEEE transactions on image processing*, vol. 30, pp. 4505–4515, 2021.

[26] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," *CoRR*, vol. abs/1705.07750, 2017. [Online]. Available: http://arxiv.org/abs/1705.07750

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[28] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. [Online]. Available: https://www.crcv.ucf.edu/research/real-world-anomaly-detection-in-surveillance-videos/

[29] P. Wu, j. Liu, Y. Shi, Y. Sun, F. Shao, Z. Wu, and Z. Yang, "Not only look, but also listen: Learning multimodal violence detection under weak supervision," in *European Conference on Computer Vision (ECCV)*, 2020. [Online]. Available: https://roc-ng.github.io/XD-Violence/

[30] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay," *CoRR*, vol. abs/1803.09820, 2018. [Online]. Available: http://arxiv.org/abs/1803.09820