TEMPORAL ANOMALY LOCALIZATION IN VIDEO

VİDEODA ZAMANSAL ANOMALİ YERELLEŞTİRMESİ

HALİL İBRAHİM ÖZTÜRK

Assoc. Prof. Dr. Ahmet Burak CAN
Supervisor

Submitted to Institute of Sciences of Hacettepe University
as a Partial Fulfillment to the Requirements
for the Award of the Degree of Master of Science
in Computer Engineering

2021

ABSTRACT

TEMPORAL ANOMALY LOCALIZATION IN VIDEO

Halil İbrahim ÖZTÜRK

Master of Science, Computer Engineering Department
Supervisor: Assoc. Prof. Dr. Ahmet Burak CAN
June 2021, 70 pages

Detecting anomalies in surveillance videos is an important research problem in computer vision. In this thesis, we propose two deep network architectures for anomaly detection, Anomaly Detection Network (ADNet) and Anomaly Detection Network by Object Relations (ADOR). ADNet utilizes temporal convolutions to localize anomalies in videos. The model works online by accepting consecutive windows of video clips. Features extracted from video clips in a window are fed to ADNet, which allows to localize anomalies in videos effectively. We propose the AD Loss function to improve abnormal segment detection performance of ADNet. ADOR employs an object detector and spatio-temporal feature extractor to fuse object relations and action information. Fusion is achieved with cross attention layers which use attention memory from cross encoders. Additionally, we propose to use F1@k metric for temporal anomaly detection. Segment based F1@k is a better evaluation metric than frame based AUC in terms of not penalizing minor shifts in temporal segments and punishing short false positive temporal segment predictions. Furthermore, we extend UCF

Crime [1] dataset by adding two more anomaly classes and providing temporal anomaly annotations for all classes. Finally, we thoroughly evaluate our model on the extended UCF Crime dataset. ADNet and ADOR produce promising results according to the F1@k metric.

Keywords: Temporal Anomaly Detection, Temporal Anomaly Localization, Surveillance Videos, Video Anomaly Detection, Deep Learning

ÖZET

VİDEODA ZAMANSAL ANOMALİ YERELLEŞTİRMESİ

Halil İbrahim ÖZTÜRK

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Danışmanı: Doç. Dr. Ahmet Burak CAN
Eylül 2021, 70 sayfa

Gözetim videolarındaki anomalileri tespit etmek, bilgisayarlı görüde önemli bir araştırma problemidir. Bu çalışmada, Anomali Tespit Ağı (ADNet) ve Nesne İlişkileri ile Anomali Tespit Ağı (ADOR) olmak üzere iki ağ öneriyoruz. Önerdiğimiz ADNet, videolardaki anormallikleri lokalize etmek için zamansal konvolüsyonları kullanan bir ağdır. Model, ardışık video klip pencerelerini kabul ederek çevrim içi veya çevrim dışı çalışabilir. Bir penceredeki video kliplerden çıkarılan özellikler, videolardaki anormallikleri etkili bir şekilde yerelleştirmeyi mümkün kılan ADNet'e gönderilir. ADNet'in anormal segment algılama performansını iyileştirmek için "AD Loss" fonksiyonunu öneriyoruz. ADOR ise, nesne ilişkilerini ve eylem bilgisini birleştirmek için nesne algılayıcı ve uzamsal-zamansal özellik çıkarıcı kullanır. Füzyon, çapraz kodlayıcıdan gelen dikkat belleğini kullanan çapraz dikkat katmanlarıyla sağlanır. Ek olarak, zamansal anomali tespiti için F1@k metriğini kullanmayı öneriyoruz. Segment tabanlı F1@k, zamansal segmentlerdeki küçük kaymaları cezalandırmamak ve kısa yanlış pozitif zamansal segment tahminlerini cezalandırmak açısından çerçeve tabanlı AUC'den daha iyi bir değerlendirme metriğidir. Ayrıca, UCF Crime [1] veri setini, iki anomali sınıfı daha ekleyerek ve tüm sınıflar için zamansal anomali açıklamaları sağlayarak genişletiyoruz. Son olarak, genişletilmiş UCF Crime veri setinde modelimizi etraflıca değerlendiriyoruz. ADNet ve ADOR yöntemleri ile F1@k metriğine göre elde edilen sonuçlar ümit vericidir.

Anahtar Kelimeler: Zamansal Anomali Tespiti, Zamansal Anomali Lokalizasyonu, Gözetim Videoları, Video Anomali Tespiti, Derin Öğrenme

ACKNOWLEDGEMENTS

Firstly, I would like to thank to my supervisor Assoc. Prof. Dr. Ahmet Burak Can for his patience, time and guidance. He supported my ideas during thesis, spent time for me in challenging times. Writing this thesis would be impossible without his support, I am grateful for his help.

I would also like to thank my colleagues and former collegues for their precious ideas and supports for my research study.

Finally and most importantly, I am deeply indebted to my parents who were my first teachers, and to my siblings. They always motivated and encouraged me throughout my master education and thesis writing period. My beloved wife motivated me to work when my energy was depleted and my desire to work decreased, I am also deeply indebted to her.

This work was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. 114G028 and 119E098.

Contents

TABLES

FIGURES

1. INTRODUCTION

Surveillance cameras become widespread in today's physical world. Abnormal cases can occur in areas (e.g. public space, shopping malls, rail stations) watched with the cameras. Detecting abnormal cases in online stream or recorded video will provide better safety in our daily life. Early intervention for people injured in a traffic accident or explosion is very important in a place where there are no people to help. Rapid detection of an arson helps prevent the fire from growing and catch the culprit. However, analyzing video streams in real-time and detecting abnormal cases require excessive human resources. Since human observation is not an effective solution, automatic anomaly detection approaches that leverage artificial intelligence mechanisms are needed in surveillance systems.

Definition of anomaly becomes crucial to design automatic anomaly detection systems. Usually abnormal cases are defined as diverging from normal cases. Since normal events are determined environment specific, abnormal events depend on the environment. Therefore, each environment usually requires a special anomaly detector trained with data containing normal events in the environment. The studies [8, 10–15] follows the approach to detect anomalies in specific environments like in UCSD [7] and UMN [6] datasets. However, special anomaly detector approach is not applicable to a wide area surveillance system. Although most anomalies are environment specific, but some events are accepted as anomaly in most environments like explosion, shooting, road accident etc. Detecting these kind of anomalies enables a global anomaly detector which can be used every camera in cities. To model real world anomalies, we need to train anomaly detection models with large number of surveillance videos on different environments and anomaly cases.

In this thesis, we propose Anomaly Detection Network (ADNet) and Anomaly Detection Network by Object Relations (ADOR) approaches to detect and localize anomalies in temporal space of videos. ADNet and ADOR use temporal convolutions to localize anomalies in temporal space by using wide time span. In order to augment data in temporal space, we split videos as overlapping with specified ratio windows. Instead of processing entire video at once, window based video processing provides better performance in ADNet. In order to obtain a model that is more applicable to real world cases, we studied with the UCF Crime [1] dataset. This dataset contains large amount of real world surveillance data with common real world anomaly cases in 13 classes. The classes are Abuse, Arrest, Arson, Assault, Burglary, Explosion, Fighting, Road Accident, Robbery, Shooting, Shoplifting, Stealing and Vandalism. Scenes in the dataset are complex environments than action recognition, temporal action localization datasets. Complex environment makes context information, human-human interactions, human-object interactions essential to detect anomalies. Additionally, long term temporal data is required to detect anomalies like as stealing and robbery.

Our first model ADNet utilizes deep action recognition models with temporal convolution networks to localize anomalies in videos. The model uses extracted features from video clips by using a spatio-temporal deep action recognition network and then takes features of all clips in a window and produces a separate anomaly score for each clip. We further define AD Loss function to increase anomaly detection accuracy. When ADNet is trained with the proposed AD Loss, performance of detecting abnormal segments increases.

Second model ADOR utilizes object detection and object relationships in addition to action information in order to detect abnormal events in videos. Extracted features corresponding to objects from object detection network are processed by an object relationship encoder. At the same time, spatio-temporal features extracted from an action recognition model are handled by action encoder. Fusion of object relationships and action information is achieved by cross connections between encoders. The encoders process a clip of video to recognize short range anomaly in the clip, long range anomaly detection is accomplished with temporal convolutional network like in ADNet.

We also propose to use segment based F1@k metric to measure effectiveness of an anomaly detection model instead of clip based AUC metric. Clip based AUC is not a good metric for this purpose since it does not take into account temporal order of clips. It penalizes minor shifts in temporal segments and can not effectively punish short false positive segments in temporal space. Effectiveness of temporal F1@k metric is shown with experiment results in Section 5.5.3. F1@k is calculated on IoU with k percentage of temporal segments obtained by thresholding anomaly scores of clips with a specified value. It is better in terms of measuring how correctly predicted temporal segment matches with ground truth temporal segment.

As a last contribution we add two more anomaly classes to UCF Crime [1] dataset, which consists of 13 anomaly classes. UCF Crime [1] data set provides temporal anomaly annotations in the test set, but only provides video level annotations in the training set. We provide temporal annotations for all anomaly classes of the data set. Finally, we thoroughly evaluate our model on UCF Crime dataset and extended dataset version, UCF Crime V2. ADNet and ADOR produce promising abnormal segment detection scores with **28.32 F1@10** and **33.94 F1@10** respectively, while the baseline model's [1] score is **4.13 F1@10**. For the normal and abnormal segments, ADNet has 58.16 F1@10 score ADOR has 63.49 F1@10 score while baseline model's score is 45.20 F1@10.

In rest of the thesis, we give background information about deep convolutional neural networks and self-attention architectures in Chapter 2. Next, we review related works about anomaly detection, action recognition, temporal action localization, image captioning and object detection in Chapter 3. Then we present first method ADNet in Chapter 4. In same chapter, we introduce segment based F1@K metric and explain UCF-Crime dataset and extensions. Evaluation results on UCF-Crime dataset and ablation studies are in same chapter. Then, we present second method Anomaly Detection by Object Relations (ADOR) in Chapter 5. We evaluate the same dataset to compare ADOR with other methods, ablation studies are also included in the same chapter. Finally, we conclude the thesis with a brief summary and a discussion on future research opinions in Chapter 6.

2. BACKGROUND

In this chapter, we give brief summary about Deep Neural Networks and some image classification models. A summary about self-attention transformer networks follows deep neural networks. Then, we explain a pre-training strategy and BERT model used in pre-training and fine-tuning. In rest of the chapter, we cover action recognition, temporal action localization, image captioning and object detection.

2.1. Deep Neural Networks (CNN)

Convolutional neural networks (CNN) are an artificial neural network class consisting of sequential convolution layers, convolution filter kernels contain learnable weights. Learning convolution filters from data by backpropagation was first achieved by LeCun et al. [16]. Pioneering CNN LeNet-5 [17] recognizes handwritten numbers by gradient based learning. Learning lots of parameters from data requires big data and parallel computing resources. Deep learning algorithms were utilizes Central Processing Units (CPU) which causes long learning schedules. Graphics Processing Unit (GPU) makes available shorter training time. Also increased visual data enables learning deep architectures without memorizing. Other advantage of convolutional neural networks that reducing parameters by using shared parameters. State-of-the-art results achieved with deep learning-based models (e.g. image classification, object detection, instance segmentation, semantic segmentation, image captioning, action recognition, temporal action localization)

AlexNet [18] is winner of ImageNet ILSVRC (ImageNet Large-Scale Visual Recognition Challenge. Architecture of AlexNet involves two branches positioned between first layer and last layer, each branches works on different GPU to utilize two GPU at the same time. Overfitting is avoided by data augmentation and dropout [19] techniques. Non-linearity is achieved by ReLU [20] activation layers.

VGGNet [21] won the localization task in ILSVRC 2014 challenge, runner up in classification task. The proposed model uses small 3 x 3 convolution filters in contrast to AlexNet that consists of 11 x 11 and 5 x 5 convolution filters in first two layers. Small filters make deeper architecture possible with approximately same number of parameters. VGGNet consists of stacked 16-19 convolution layers. The authors underline importance of depth in CNNs.

Winner of ILSVRC 2015 challenge in image classification task is ResNet [2]. While maximum number of convolution layers in VGGNet is 19, ResNet consists of 152 convolution layers with less number of parameters and less complexity. The reason of that removing fully connected layers placed after convolution layers in VGGNet . Residual connections in ResNet makes training deeper networks possible. Residual connection which is shortcut connection between input and output of convolution layer, avoids gradient vanishing problem by flowing gradients from shortcut connections. An ensemble of residual connections ResNet with 152 convolution layers achieves 3.57% top-5 error rate on the ImageNet test set.

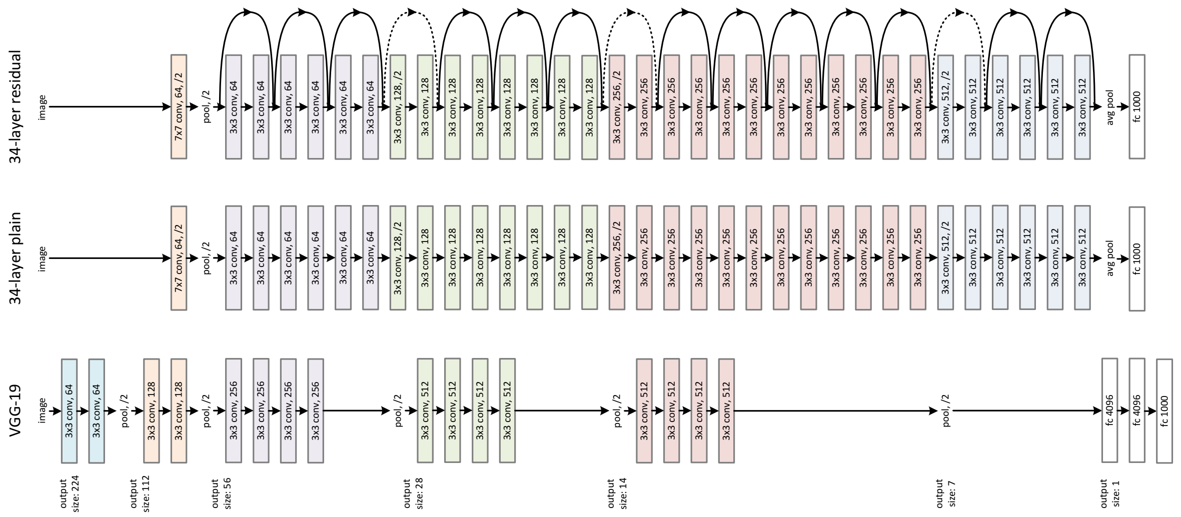


Figure 2.1. ResNet-34 is deeper than VGGNet-19, residual connections increase success of deep networks (Image is taken from [2])

MobileNet [22] introduces dept-wise separable convolutions to decrease required computation power. Depth-wise separable convolutions consists of two sequential convolution layers. Kernel size of first convolution is 1 x 1, the convolution mixes features in different channels. Second convolution kernel has 3 x 3 weights, the convolution operation occurs on only one channel of feature map. Computation decreases from H X W X C_{in} x C_{out} to H X W X C_{in} where $C_{in} = C_{out}$. Top-1 score of Mobilenet is 70.6% in ImageNet test set. y

MobileNet v2 [23] improves ImageNet image classification accuracy by inverted residual blocks. While MobileNet does not use residual connections, inverted residual blocks of Mobilenet v2 has residual connections between inputs and outputs. Forwarded feature map to inverted residual block is expanded in contrast to ResNet block that shrinks feature map. Activation function follows convolution layers of blocks in MobileNet and ResNet, but experiments in the study show that activation function in bottlenecks hurts performance. For that reason, activation functions in bottlenecks in blocks are removed. MobileNet v2 achieves 74.7% top-1 score in ImageNet test set.

Mobilenet v3 [24] has better architecture that is tuned by Neural Architecture Search algorithm. Inverted residual blocks are improved by inserting squeeze and excitation [25] in blocks. ReLU 6 activation function is replaced with h-swish activation function which is improvement of swish function. Top-1 score of the model is 75.2% in ImageNet test set.

EfficientNet [26] model gains efficiency by scaling model with compound coefficient in depth, width and input image resolution. The architecture consists of inverted residual blocks as in MobileNet v3. Top-1 score of smallest model efficientnet-b0 is 77.1% in ImageNet test set. Largest model EfficientNet-b7 achieves 84.3% top-1 score.

2.2. Action Recognition

Action recognition in untrimmed videos is important problem for video understanding tasks (e.g. temporal action localization, video anomaly detection etc.). Static image is not sufficient to recognize most actions, since action is combination of spatial and temporal information. For example fighting and dancing can not be distinguished from static image, but playing football can be understood from background information in static image. Temporal and spatial knowledge fusion happens by three methods late fusion, early fusion and slow fusion.

A slow fusion model C3D (Convolutional 3D) [27] input is fixed number consecutive RGB frames. 2D convolutions lose temporal information of the input signal right after every convolution operation. 3D convolution operation convolves on spatial and temporal space, while

2D convolution does not work on temporal space. 3D max pool decrease temporal and spatial dimension of feature to compute efficiently. Kernel sizes in all layers are 3 x 3 x 3, the architecture with the shape in all layers is best in experimented architectures.

Other slow fusion model I3D [28] recognizes action in clips by utilizing consecutive RGB and Optic-flow frames. Different type inputs are handled by two separete network branches. Generated probabilities for clips are summed together to get final probabilities. Also RGB and Optic-Flow branches are trained independently. I3D consists of 3D convolution layers and 3D max pool layers as [27]. Convolution kernels are initialized with pre-trained Inception V1 network [29] on ImageNet dataset. First two 3D convolution outputs are pooled in only spatial dimension, temporal dimension is not reduced. Each convolution and fully connected layer outputs are normalized with 3D Batch Normalization layer, C3D [27] does not consist of batch normalization layers. Optic flows are computed with TV-L1 [30] algorithm.

TSM (Temporal Shift Module) [31] uses 2D convolutional neural networks for each frame unlike I3D and C3D networks. Processing temporal knowledge is achieved by shifting part of channels to next adjacent frame's network. 1/4 of channels are shifted in forward and backward direction, 1/8 of channels forward, 1/8 channels backward. Each temporal shift in network doubles receptive field. Predictions of each 2D network are averaged to get final predictions for clip. TSM achieves better accuracy than I3D in Kinetics dataset [32].

CSN (Channel-Separated Convolutional Networks) [33] action recognition model reduces computation cost in training and testing phase. I3D and C3D use dense 3D convolution operations, all channels in input are used in generating output of dense convolutions. Mobilenet [22] reduces computation cost of convolution blocks by replacing dense convolutions with depth-wise separable convolutions, as we mentioned before. Similarly CSN uses 3D separated convolution operations which 3 x 3 x 3 channel-wise convolution follows 1 x 1 x 1 convolution filter which provides information shuffle between channels. Proposed two type channel separated bottleneck blocks differ in interaction strategy between channels. Interaction-reduced channel-separated bottleneck block removes 1 x 1 x 1 convolution filter positioned front of 3 x 3 x 3 depth-wise convolution. Data flow in Interaction-preserved channel-separated bottleneck block is 1 x 1 x 1 dense convolution then 3 x 3 x 3 depth-wise convolution. Interaction preserved CSN achieves better score than interaction reduced CSN. Pretrained model on IG-65M dataset [34] achieves 3% more accuracy in Kinetics-400 [32] dataset. IG-65M action recognition dataset consists of 65 million videos labeled in video-level. Depth-wise separable convolution gains better regularization opposed to dense 3D convolution, since CSN training accuracy is less than dense 3D convolution network but higher higher test accuracy

R(2+1)D [35] splits spatio-temporal convolution to spatial convolution and temporal convolution. 2D convolutions operate on static images in clip without using temporal knowledge. 1D convolutions follow 2D convolutions in each block to process temporal information. Since activation functions follows 1D and 2D convolutions, dividing spatio-temporal convolution into two parts doubles non-linearity. Unlike I3D, R(2+1)D use stride to down-sample features.

2.3. Temporal Action Localization

Localizing action segments in temporal space on untrimmed long videos provides meaningful information. Temporal action localization models accept consecutive frames and outputs action class or background class label for each frame. The models are build on action recognition models. Since long range temporal information processing is necessary, temporal convolutions or recurrent neural networks (RNN) are used to extend the receptive field of action recognition model.

TCN (Temporal Convolutional Network) [36] utilizes 1D convolutions over temporal space to segments and classify action in time. The proposed architecture processes all time entries together, unlike recurrent neural networks which processes time by time. Temporal convolutions operate on extracted spatio-temporal features extracted across time from a video. Encoder-Decoder TCN (ED-TCN) one of the proposed two architectures consists of temporal convolutions which output same size with input and maxpool layers. Maxpool layers halve time dimension to process efficiently. Upsampling time dimension in decoder part is achieved by repeating time entry twice. ED-TCN achieves better score than Bi-LSTM [37]. Second architecture Dilated TCN does not use max pool to down sample feature. Efficiency is accomplished by dilated temporal convolutions. Dilation rate of convolution layers are calculated with $2^{LayerOrder}$ formula. Increased dilation rate along layer order provides large receptive field. Some studies [38, 39] evaluates performance of the method with frame-wise metrics as in video anomaly detection methods. Segment-wise metric is proposed by the method.

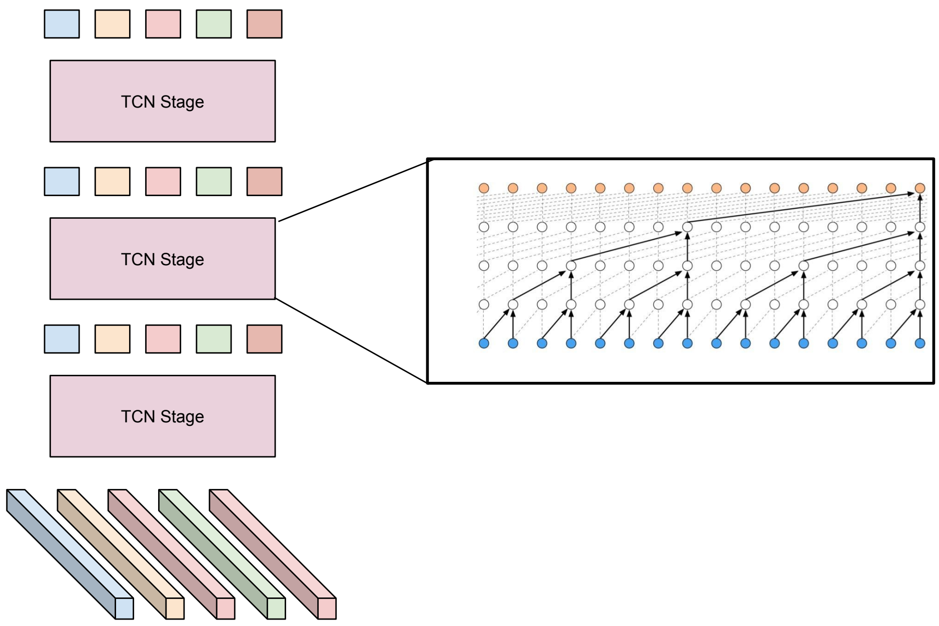


Figure 2.2. Multi-Stage Temporal Convolutional Network (Inside of stage image is taken from [3])

MS-TCN (Multi-Stage Temporal Convolutional Network) [40] consists of multiple stages which refine output of previous one as in Figure 2.2. Spatio-temporal features are feed forwarded to predict action segments in temporal space. Following stages refines by accepting output of previous stage. Each stage is criticised with ground-truth. Calculated stage errors are summed to back propagate together. Dilated temporal convolutions with increased dilation rate comprise stages of MS-TCN. Proposed loss function has Mean Squared Error (MSE) loss as classification loss and a smoothing loss. Smoothing loss forces to reduce difference between consecutive frames.

TAL-Net [41] follows two stage architecture in object detection similar to Faster R-CNN [42]. In first stage, segment proposals are generated, second stage classifies segment proposals. Segment proposal network outputs proposals for each anchors. Since predefined anchor widths operates on different temporal span, receptive fields of anchors are different. Segment proposal of TAL-Net benefits from anchor specific proposal networks names as multi-tower network. Each anchor specific proposal network has receptive field with similar width to anchor. Fixed size segment proposal features are obtained with 1D RoI Pool (SoI Pool). In proposal generation and proposal classification phase rgb features and optic flow feature are used in two branches. Outputs of branches are fused lately to get better result, like I3D [28].

2.4. Transformer

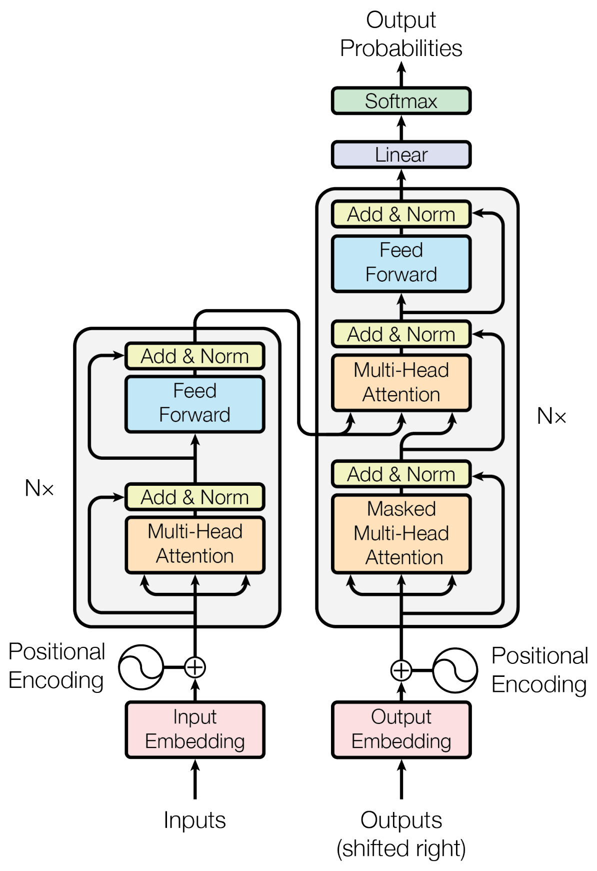


Figure 2.3. Transformer model architecture (Image is taken from [4])

A Sequence to sequence model Transformer [4] consists of encoder part and decoder part. Input sequence is accepted by encoder and the sequence is encoded in one pass. Memory encoded in encoder flows to decoder layers to generate output sequence. Transformer is proposed solution to a sequence to sequence problem machine translation. The problem is that translating a text from source language to target language. Encoder part of Transformer consists of stacked encoder layers which have self-attention layer and feed forward neural network. Words in input sentence are represented with word embedding vectors. In order to specify position of word embedding vectors are summed with positional vectors calculated with Formula 1.

$$PE_(pos, i) = \begin{cases} sin(pos/1000^{i/d_{model}}), & \text{if } i \mod 2 = 0 \\ sin(pos/1000^{(i-1)/d_{model}}), & \text{if } i \mod 2 = 1 \end{cases} \tag{1}$$

Query, Key and Value vectors comprise attention mechanism. Each query vector in the sequence is multiplied with all key k vectors in same sequence to compute weight of corresponding value v vectors. Obtained weights are multiplied with relevant value vector, then weighted value vectors are summed to get vector for the query. Query, key and value vectors are calculated from input vectors as in Formula 2.

$$\begin{aligned} Q &= W^Q * X \\ V &= W^V * X \\ K &= W^K * X \end{aligned} \tag{2}$$

Calculated weights from query and key vectors are normalized with softmax function so that their sum is one as in Formula 3 . Before normalization weights are divided to d_{model} which is length of input vector.

$$Y = Softmax(\frac{Q * K^T}{\sqrt{d_{model}})}) * V \tag{3}$$

Decoder generates target sequence by encoded memory and previous state of target sequence. After N encoder layers K key and V value vectors as memory are passed to each decoder layers. Decoder layers involves two attention layers and one feed forward neural network. First attention is self-attention layer, which applies attention on already predicted sequence words' embedding vectors. Second attention is encoder-decoder attention layer, Q vectors are calculated from already predicted sequence words' embedding vectors, K key and V value vectors are calculated from memory comes from encoder. After N decoder layers, output vector of last element in sequence is forwarded to linear layer to predict next word. Prediction starts with *[CLS]* token, after prediction each element in sequence, predicted sequence is accepted by decoder to predict next element until get *[EOS]* token.

2.5. Finetuning Pretrained Transformers

Before training model on a dataset, training on different and usually larger dataset from the dataset is named as pre-training. After pre-training the model can be fine-tuned for the dataset with less effort or little data. Also better score can be achieved by pre-training on large dataset and fine-tuning on the dataset as in [33, 34, 43–47]. BERT [5] using the strategy is pre-trained on a large unlabeled dataset to learn the language representation. Pre-trained model can be fine-tuned in other language tasks without model modification. BERT accepts two sentences and outputs two sentences, input and output sentences concatenated to get one sequence. *[SEP]* token is in between sentences to indicate the end of first sentence. Difference between transformer and BERT is that BERT consists of only encoder part.

Two pre-training methods are performed to teach learn language representation to BERT. First method is masked language modelling model that predicting masked WordPiece [48] tokens which selected randomly with 15% probability. The second strategy is guessing whether the next sentence is the actual or replaced sentence. Predictions are taken from the corresponding output from the [CLS] token in the input queue. BookCorpus [49] dataset and extracted texts from English Wikipedia is used during training.

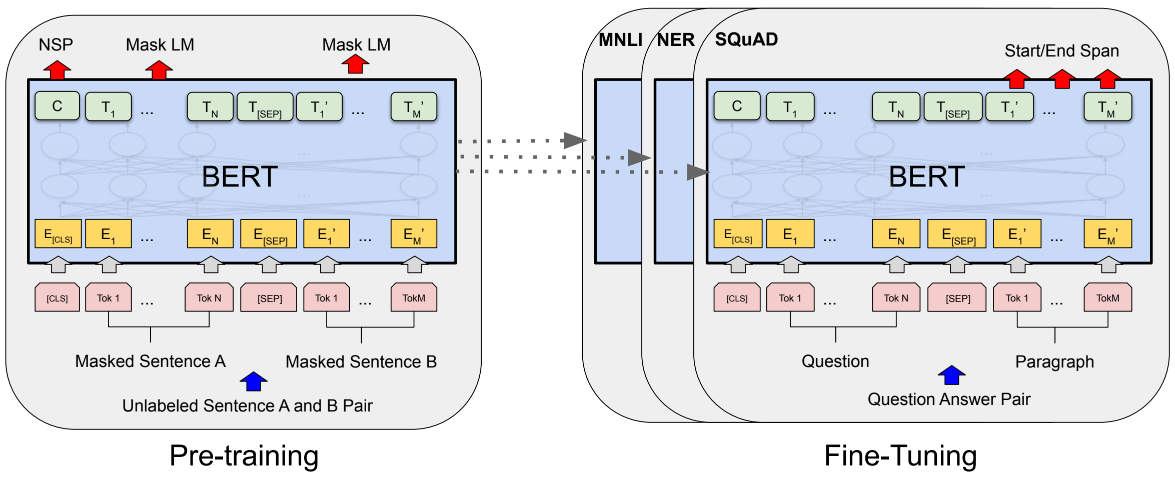


Figure 2.4. BERT is pre-trained on unlabelled dataset to learn language representation, then pre-trained model is fine-tuned on NLP tasks without modification on model (Image is taken from [5])

In order to prove success of the method BERT is fine-tuned on GLUE [50] Natural Language Processing Tasks. For each task pre-trained BERT is fine-tuned for 3 epochs. The tasks are Multi-Genre Natural Language Inference (MNLI) [51], Quora Question Pairs (QQP), Question Natural Language Inference (QNLI) [50], The Stanford Sentiment Treebank (SST-2) [52], The Corpus of Linguistic Acceptability (CoLA) [53], The Semantic Textual Similarity Benchmark (STS-B) [54], Microsoft Research Paraphrase Corpus (MRPC) [55], Recognizing Textual Entailment (RTE) [56], Winograd NLI (WNLI) [57]

Also BERT is finetuned on The Stanford Question Answering Dataset (SQuAD v1.1) [58] for 3 epochs wit 5e-5 learning rate and The Situations With Adversarial Generations(SWAG) [59] dataset for 3 epochs with 2e-5 learning rate.

2.6. Object Detection

Faster R-CNN [42] method outputs class and bounding box of objects that exists in image. The object detector has two stages, Region Proposal Network (RPN) is first stage of the method. Second stage is that predicting bounding box and class of object proposal. Fast R-CNN [60] generates object proposals with unshared network expensively. In contrast to Fast R-CNN, Region Proposal Network in Faster R-CNN shares backbone network (e.g. Resnet) with bounding box and class label prediction. For reason of that, the computation becomes cheaper than Fast R-CNN. Classification and bounding box regression layers accept fixed size features, but proposal sizes are not fixed. RoI Pool operation proposed by Girshick et al. [60] yields same size features for each proposal.

2.7. Image Captioning

Image captioning methods provides natural language description (caption) about given image. The description includes relations between objects, background definition, actions recognizable from static image. The task flows information extracted from computer vision area to natural language processing area. Herdade et al. [61] propose transformer [4] based method to generate captions. Before forwarding features to encoder, objects in image are detected by Faster R-CNN which pre-trained [62] on Visual Genome [63] object detection dataset. Object features flows over box attention layer and fully connected layer. Box attention layer is modified attention layer of transformer [4]. In order to consider spatial relations between objects in weighting features, relative position of an object to another object is calculated with Formula 4. The spatial relation feature is upsampled to 512d from 4d feature vector with linear embedding layer in Formula 5. Multiplication W_G matrix with produced embedding gives object relation weight. The final weight is obtained by summing the object relationship weight and the weight calculated from the attributes. Attention output of the vector is calculated with Formula 7.

$$\lambda(a,b) = (log(\frac{|x_a - x_b|}{w_a}), log(\frac{|y_a - y_b|}{h_a}), log(\frac{w_b}{w_a}), log(\frac{h_b}{h_a})) \tag{4}$$

$$obj\_rel(a,b) = ReLU(Emb(\lambda(a,b)W_G)) \tag{5}$$

$$w_{ab} = \frac{obj\_rel(a,b) + exp(\frac{q_a * k_b^T}{\sqrt{d_{model}}})}{\sum_{l=1}^{N} obj\_rel(a,l) + exp(\frac{q_a * k_b^T}{\sqrt{d_{model}}})} \tag{6}$$

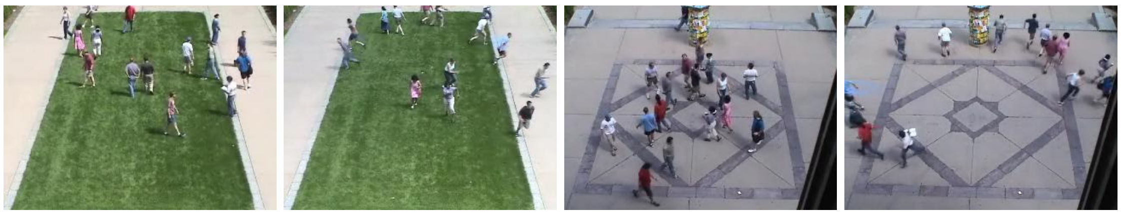
$$Y_a = \sum_{l=1}^{N} w_{al} * v_l \tag{7}$$

3. RELATED WORKS

In this chapter we review related works from anomaly detection. Video anomaly detection datasets are briefly reviewed in the chapter.

3.1. Video Anomaly Detection Datasets

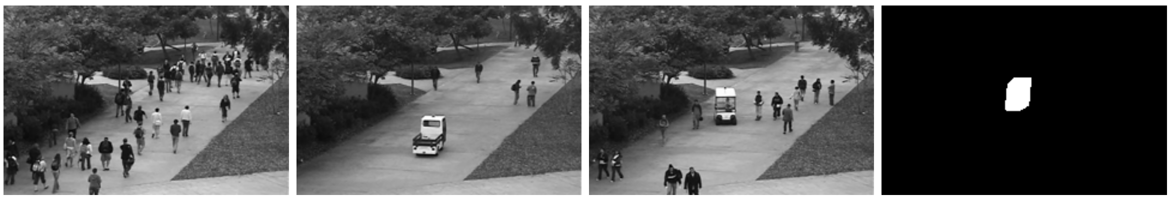
There are limited number of datasets on anomaly detection in the literature, (eg. UMN, UCSD etc.). Most of these datasets have small amounts of data and specific anomaly cases which are not applicable to real world scenarios. UCF crime is the first dataset to contain comprehensive real-life anomalies. In this section, we will briefly explain these dataset.



(a) Normal (b) Abnormal (c) Normal (d) Abnormal

Figure 3.1. UMN [6] Dataset sample images

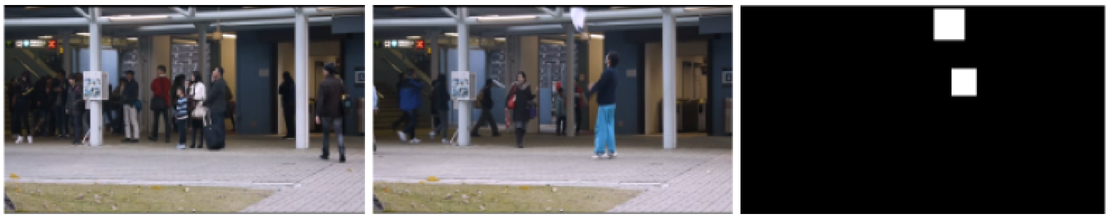
UMN [6] dataset consists of 11 videos and 7739 frames, including 1 indoor and 2 outdoor scenes. Images are 320×240 resolution. Each video starts with normal behavior and ends with panic abnormal behavior.



(a) Normal (b) Abnormal (c) Abnormal (d) Mask of Abnormal Case

Figure 3.2. UCSD [7] Dataset sample images

UCSD [7] dataset consists of two different scenarios, Pad1 and Pad2. Pad1 has 34 training and 36 test video samples, while Pad2 has 16 training and 12 test image samples. Abnormal events here are caused by cyclists, skaters or small cars circulating on the walking paths. Abnormal events in the videos are quite simple and do not reflect real life anomalous scenarios.



(a) Normal (b) Abnormal (c) Abnormal

Figure 3.3. Avenue [8] Dataset sample images

Avenue [8] dataset consists of 16 training and 21 test video examples. Video durations are approximately two minutes. While normal scenes consist of people walking on stairs and subway entrances, abnormal events consist of situations such as approaching the camera, people running/walking in the opposite direction.

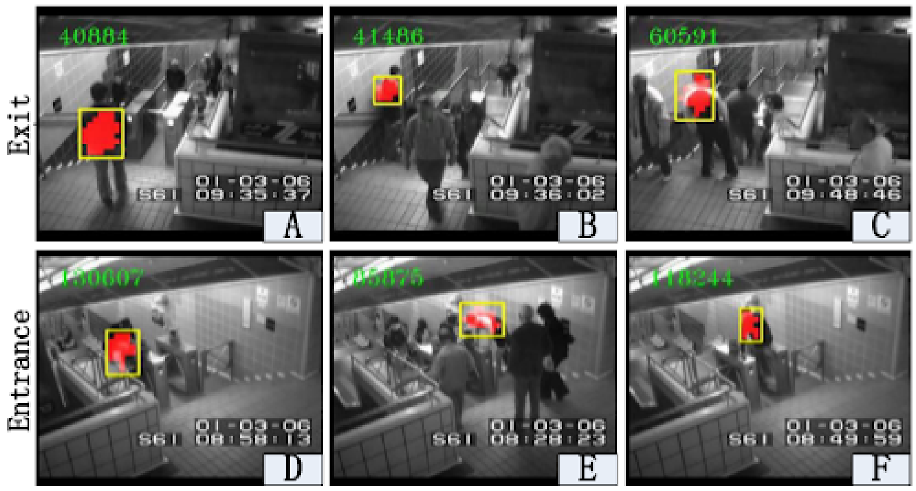


Figure 3.4. Subway [9] Dataset sample images

Subway [9] dataset consists of two subsets, input and output. The entrance door scenario is 96 minutes and the exit door is 43 minutes. Image resolutions are 512 x 384 pixels. In the entry scenario, there are abnormal events such as people moving in the wrong direction, stopping suddenly or running. In the exit scenario, there are some abnormal events such as people moving in the wrong direction and walking around near the exit door.

UCF-Crime [1] dataset contains 13 real world abnormal cases and normal cases. The covered anomalies are Abuse, Arrest, Arson, Assault, Accident, Burglary, Explosion, Fighting, Robbery, Shooting, Stealing, Shoplifting, and Vandalism. 800 normal and 810 abnormal videos constitutes training split of UCF-Crime. Test split consists of 150 normal and 140 abnormal videos. Training set is annotated video-level labels, but test set is annotated temporarily. Sample images of UCF Crime dataset can be seen in Figure 4.6.

3.2. Anomaly Detection in Computer Vision

Reconstruction methods learn a model or dictionary from normal videos to reconstruct given image with small errors. Abnormal inputs which contain unseen objects, actions or attributes can not be reconstructed with learned model successfully. The error between input and reconstruction reveals abnormal cases. An anomaly detection with reconstruction is proposed by Lu et al. [8]. Proposed model uses dictionary learned from normal videos. Firstly, scaled images to three sizes are divided to regions. For each region a relevant dictionary learned from normal videos to reconstruct region feature with dictionary elements. The region feature is extracted from ordered frame sequence in the region. L2 distance between reconstructed feature and extracted feature from region gives error rate. High error rate in a region of the image reveals anomaly in the scene. Proposed method decreases computation time during test.

Anomaly detection method by sparse coding model is proposed by Luo et al. [10]. The method aims to increase sparse code similarity between close frames in temporal space. Proposed Temporally-coherent Sparse Coding (TSC) objective is used in sparse coding and proposed stacked Recurrent Neural Network (sRNN) optimization. RNN works as auto-encoder to reconstruct input feature x_t at time t. Output of tth time is forwarded to hidden states of $t + 1$ th state. Also distance calculated by proposed formula between x_t and x_{t-1} feature vectors are passed to hidden state. In inference phase, after calculation reconstruction error for all frames in a video, errors are normalized with defined formula.

Alternative reconstruction method to sparse coding is deep neural networks. Hasan et al. [11] propose that reconstructing image and HOG+HOF features to detect reconstruction error. Regularity of reconstruction error in temporal space is provided by HOG+HOF features extracted along with trajectory information. Two branch of the network reconstruct HOG+HOF features and input image with encoder-decoder architecture. Local appearance and motion features are encoded to HOG and HOF descriptors. One branch of the network accepts concatenated HOG and HOF descriptors to reconstruct HOG+HOF feature vector. Due to the data scarcity, data augmentation is made on temporal dimension.

Nguyen et al. [12] propose encoder-decoder convolutional neural network to detect anomaly in scene and localize it. RGB Image and optic flow map is reconstructed from RGB input image by appearance convolutional auto-encoder. The model has one encoder that encode RGB image to lower dimension feature, two decoders which reconstruct RGB image and optic flow from encoded feature. Training is performed on normal videos for a scene. Reconstruction of abnormal input is divided to patches, comparison between corresponding input patch and output patch with proposed formula gives score of abnormality. Patch by patch comparison provides spatial localization. Proposed model can not be used as global anomaly detection in all scenes, since not seen objects or actions in scene will be classified as anomaly, e.g. change of lighting in train.

Discriminate predicted next frame by using generative model U-Net is another anomaly detection method proposed by Liu et al. [13]. U-Net [64] consists of convolution and transposed convolution layers to generate output with input shape. Proposed method predicts optic flow map from generated next frame and current frame. Predicted optic flows are criticised with ground truth next frame and current frame, in order to include motion knowledge. Anomaly is detected by discriminator which accepts predicted next frame and real next frame. Discriminator loss is calculated patch by patch by patch base loss in discriminator after prediction. Gradient, illumination difference in prediction, and discriminator output comprise generator objective. Generative adversarial learning schedule [65] is used by the proposed method. While generator is being trained, discriminator weights are fixed. Similarly generator weights are fixed during training discriminator.

Ravanbakhsh et al. [14] propose method to learn normal scenes with Generative Adversarial Network (GAN) architecture. The network consists of two generator sub-networks to generate optic flow map from image and generate image from real optic flow map. Difference between image output and corresponding input is calculated with feature extracted from AlexNet [18] network, in training phase. In test phase, Anomaly is detected by computing local differences between reconstructed outputs and inputs, computation between images is done directly from the image, not with features extracted from AlexNet as in training phase. Differences of optic flow maps and images come together to produce anomaly map.

Anomaly detection with using high level information (object, motion, attribute) is achieved by Hinami et al. [15]. Multi-task object detector has been trained on object detection datasets. The detector predicts object class, action and attribute from object proposal regions. Backbone network is AlexNet and RoI Pooling operation [60] is applied to feature map to get same size features for each object proposal. The distribution of classification scores is modeled for each category in target environment during training. In test stage, density of predicted category triplet is estimated by kernel density estimation (KDE). Estimated density is used as anomaly score.

Sultani et al. [1] propose anomaly detection method trained by multiple instance learning (MIL) method. In training stage, each video is split to fix number segments. If a video is annotated as abnormal, segments are put to positive bag, else they are put to negative bag. Spatio-temporal features of segments are extracted by C3D network. Maximum scored abnormal segment in positive bag is enforced to higher than maximum scored normal segment by modified hinge loss. Also modified hinge loss enforces difference in scores of neighbour segment to become less. The method is trained and tested on published UCF-Crime dataset.

4. ANOMALY LOCALIZATION WITH TEMPORAL CONVOLUTIONS

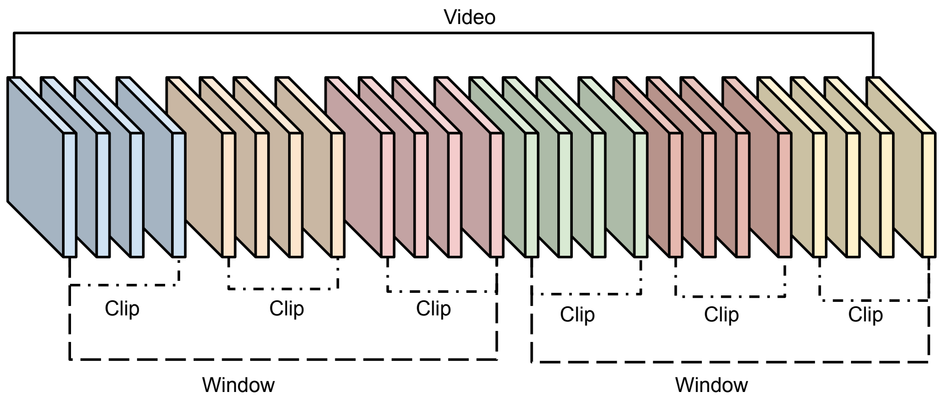


Figure 4.1. Video, video clip, window

In this section we introduce Anomaly Detection Network (ADNet) for anomaly detection and segmentation in temporal space. Input to ADNet is a set of features $x_{1:C} = (x_1, x_2, ..., x_C)$ extracted using a spatio-temporal CNN for each clip of given video, where C is number of clips for a video. A video clip consists of a number of consecutive frames of video, as in Figure 4.1. Number of frames in a video clip and frame shape are determined according to pre-trained spatio-temporal CNN. ADNet outputs anomaly probabilities $y_{1:C} = (y_1, y_2, ..., y_C)$ for each clip, where $y_t \in [0, 1]$. Target values are $a_{1:C} = (a_1, a_2, ..., a_C)$ where $a \in \{0, 1\}$. 0 and 1 indicates normal and abnormal classes, respectively. Labels are generated from output probabilities by applying 0.5 threshold value, which can be changed in inference time to adjust the model for different anomaly conditions.

We describe ADNet in Section 4.1. Then we explain Temporal Sliding Window in Section 4.2.. At the end of method section, we discuss the proposed loss in Section 4.3.

4.1. Temporal Anomaly Detection Network

To predict anomalies in a timeline by using temporal information, we adapt and improve MS-TCN [40] model which makes classification and temporal segmentation. We input all clip features of a video without splitting to small parts.

MS-TCN defines a sequence of stages, which has a series of blocks that consists of convolutional layers as in Figure 2.2. Each stage outputs anomaly scores, y_t, for each video clip features, x_t. Input of next stage is the output of previous stage, except the first stage. In the first stage, input video clip features are shrunk to D^H channel-size with 1x1 convolutions, where D^H is the number of channels of features used in hidden layers of stages. After downsampling, each video clip feature is represented with H dimensional feature vector. In rest of the stages, dilated 1D convolution operation is done on T downsampled clip features over temporal dimension. Except the first layer and last layer in the stages, input and output shapes of layers are same, this is achieved with padding. Dilation rate of convolutions is calculated with 2^l formula, where l is order of the layer in stage (i.e. $0, 1, 2, .., L$). Dilated convolutions increase receptive field of network with small kernel size. ReLU activation function which follows 1x1 convolution provides non-linearity. Residual connections between before and after convolutions in a block avoids gradient vanishing problem. Output of a stage is defined as follows:

$$Y^s = Sigmoid(W * V + b) \tag{8}$$

where $*$ is the convolution operation, W is filter kernel and b is bias value of kernel, $Y^s \in \mathbb{R}^{1xT}$ is the vector of anomaly scores outputted by s^{th} stage.

We input the output of previous stage, Y^{s-1}, to the next stage, Y^s. At the start of the stage, 1x1 convolution is applied to increase channel size from 1 to D^H.

4.2. Temporal Sliding Window

In the previous section, the model accepts the whole video clip features $(x_1, ..., x_T)$ of a video to the network. We split consecutive clip features to windows to augment data. Data augmentation provides better generalization. Each window contains W consecutive clip features, $(x_1, x_2, ..., x_W)$. If a video has less clip features than window width, $T < W$, we pad window with empty clip features x_0, which is filled with 0 values. Let $x = (x_1, x_2, x_0, ..., x_0)$

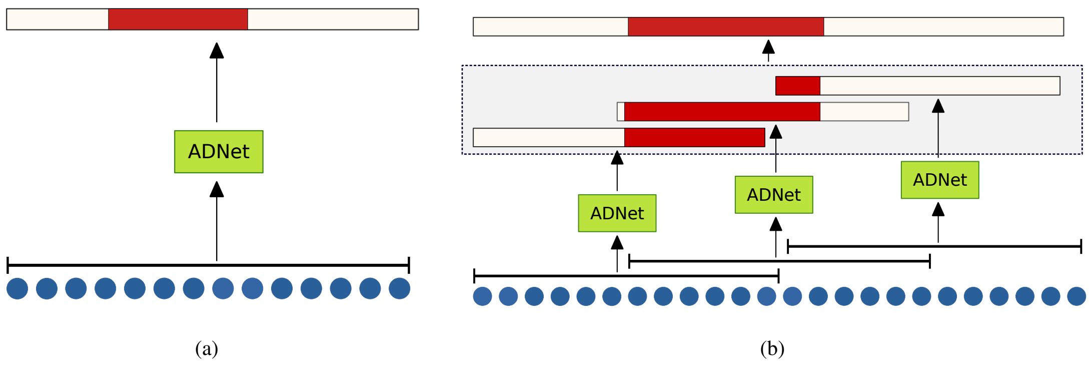


Figure 4.2. Temporal convolutional networks *(a)* accepts clip features x_i which are presented with blue circles. Instead of convolving over clip features of a video as in *(a)*, clip features are divided to equal windows to pass ADNet *(b)*. Windows intersect with previous windows, output scores in overlapped parts are averaged to get final output scores. Abnormal results are presented with red colour.

be an input and $m = (1, 1, 0, ...0)$ be the mask for x, where $x \in \mathbb{R}^{D_0 * W}$ and $m \in \mathbb{R}^{1*W}$. Information flow from padded empty clip features is blocked by masking outputs of convolutions as follows:

$$V^l = (W_2 * ReLU(W_1 * V^{l-1} + b_1) + b_2) \cdot m \tag{9}$$

where V^l is the output of l^{th} layer, (W_1, W_2, b_1, b_2) are parameters of convolution, $\cdot$ is dot product.

Temporal window approach makes also online anomaly detection and segmentation possible, while online detection is impossible in the network that accepts whole clip features as input. The window step size is set to half of the window size for smoother inference results, this increases the number of training clip windows as well. In other words, we augment data with by splitting videos to windows and using half stride windows during training. Start position of window i is calculated with $w_i^{start} = W/2 * (i - 1)$, end position formula is $w_i^{end} = W + W/2 * (i - 1)$. We use same window size for a network in both train and test stages. In the inference, we average anomaly scores of overlapping windows, as in Figure 4.2.(b)

Each layer in a stage except the last layer outputs $V^{T * D_H}$, where dimension of output is same for the convolutions. To match input dimension with output dimension, we pad input P, where $P = \lfloor K/2 \rfloor * 2^L$, $P < W$, K is kernel size. If padding P is equal to input size of window W, more than half of the inputs of convolution would be padding elements. Window width can not be more than the receptive field, where $ReceptiveField = 2^{l+1} - 1$. Therefore, to avoid information loss, we determine maximum number of layers for a window width W as follows:

$$L = \left\lceil log_2 \frac{W}{\lfloor K/2 \rfloor} \right\rceil \tag{10}$$

4.3. AD Loss

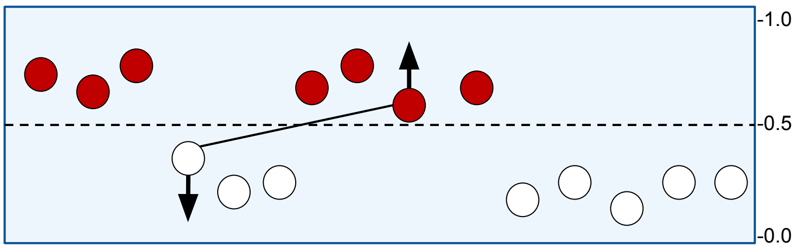


Figure 4.3. AD Loss increases distance between nearest opposite pair in manner of anomaly score for each clip. Red circles represent abnormal clips, white circles represent normal clips

Intuition behind ADLoss is similar to VSE++ [66], increasing distance between hard pairs. Hard pair of a normal video clip is abnormal video clip which has closest anomaly score to the normal video clip. Similarly hard pair of an abnormal video clip is normal video clip with closest anomaly score as in Figure 4.3. Since scores of the video clips are not same between iterations, hard pairs of the video clips change in each of step of training. ADLoss aims to increase distance between each hard pair to α value with Formula 12. Combination of L_{MSE} mean squared error loss and L_{AD} anomaly detection loss produces the final loss value for s^{th} stage as in Formula 11. Contribution of L_{AD} is controlled with $0 < \lambda < 1$ parameter in Formula 11. L_{AD} is calculated as in Formula 12, where y_A is score of abnormal input, y_{HN} hard normal of the abnormal input, y_N is score of normal input, and y_{HA} hard abnormal of the normal input.

$$L_s = L_{MSE} + \lambda * L_{AD} \tag{11}$$

$$L_{AD} = max(-\sum_{A}(y_A - y_{HN} - \alpha) - \sum_{N}(y_{HA} - y_N - \alpha), 0) \tag{12}$$

Final loss is summation of each stage losses which are produced from each stage anomaly probabilities. We calculate loss for each stage output as in Formula 13. Final loss is minimized during training.

$$L = \sum_{s} L_s \tag{13}$$

4.4. Implementation Details

We extract clip features from I3D[28] by applying average pooling to activations before classification layer. Video clips for I3D are generated with 16 frame temporal slide. We chose 16 temporal slide instead of 1 to decrease inference time. TSM[31] is the second feature extractor used in our study. We use Adam [67] optimizer with 5e-4 learning rate in all experiments. We trained and tested our models on PyTorch [68] framework. Kernel size and channel size of ADNet are 3 and 64 respectively in all settings. Labels of clips are determined by distribution of normal and abnormal classes.

4.5. Evaluation Metrics

The baseline method [1] and most other studies use AUC metric to measure anomaly detection performance. Abnormal cases happen in a segment of timeline. AUC metric evaluates performance of each clip independently, in other words, ignoring the temporal orders of the clips. For this reason AUC cannot do the necessary punishment when short false positives arises. An appropriate metric should penalize such false positives because they create excessive number of alarms in a smart surveillance system. A good metric also should not

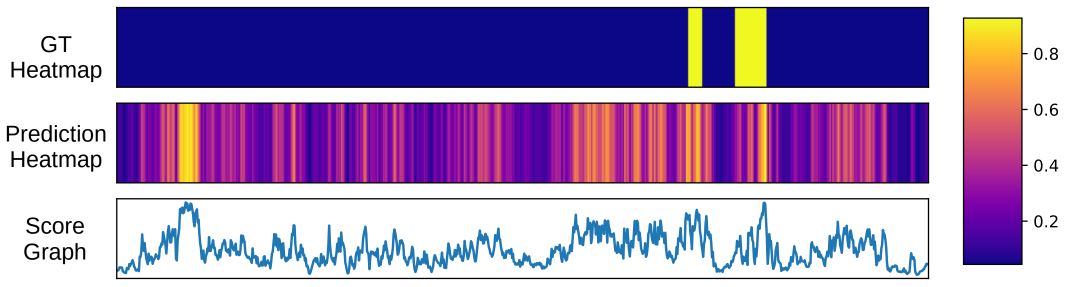


Figure 4.4. First two rows are heatmaps of ground truth, predicted anomaly scores for *Assault 10* video from the test set. Last row is graph of anomaly scores which is between 0 and 1. AUC score is 74.83, F1@25 score is 24.99 for this video

penalize should not penalize minor shifts between predicted segments and ground truth segments while penalizing over-segmentation errors. However, clip wise AUC metric penalizes minor shifts. Therefore, we approach to the problem as temporal action localization. We adapt F1@k metric proposed in [36] for evaluating anomaly detection performance, which handles the weaknesses of AUC metric better.

Abnormal segments form small part of ground truth in a test sample. For this reason, either wrong predictions in abnormal segments or small false positives in normal segments do not sufficiently affect AUC score. Figure 4.4. shows timelines of ground truth and prediction segmentation for a test video. While AUC score of the test video is **74.83** , F1@25 score is **24.99** . This example shows the robustness of F1@k metric comparing to AUC metric for this problem.

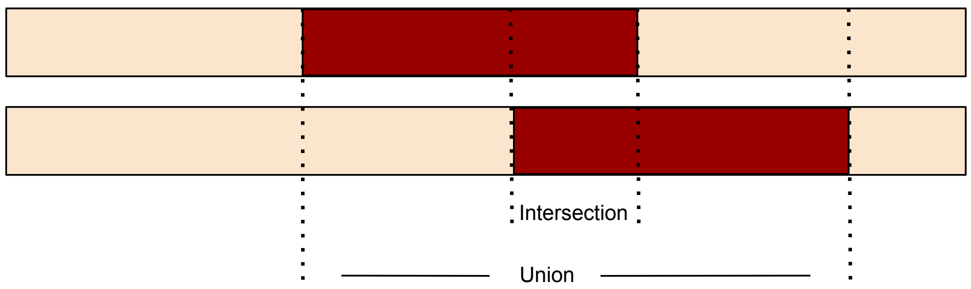


Figure 4.5. IoU between temporal segments

F1@k is calculated by k percentage intersection over union (IoU) between predicted temporal segments and ground truth temporal segments as in Figure 4.5. However, [36] does not include background segments to F1@k metric evaluation. In the anomaly detection problem, normal segments can be considered as background segments. In order to increase penalization for over-segmentation of abnormal segments, we do not consider segments with normal events as background segments, we include them to evaluation.

As mentioned before, we extract features for each clip from spatio-temporal networks. Let $C_i = (f_{n*i}, ..., f_{n*(i+1)-1})$ be a clip where f_j is j_{th} frame of given video. Since accepted number of frames from feature extractors can be different, number of clips can be different for given video with respect the selected feature extractor. To evaluate fairly, we produce frame level labels or scores from clip level results by copying clip scores y_i to frame scores $\hat{y}_j$ of the clip, $\hat{y}_{n*i,...,n*(i+1)} = y_i$. In other words, we make evaluation in frame-level instead of clip level.

4.6. Dataset

UCF Crime V2								
Class	Train	Test	Class	Train	Test	Class	Train	Test
Abuse	48	2	**Arrest**	45	5	**Arson**	41	9
Assault	47	3	**Protest***	96	8	**Burglary**	86	13
Explosion	29	21	**Fighting**	45/33	5	**Molotov Bomb***	91	9
Road Accidents	127	23	**Robbery**	145	5	**Shooting**	30	23
Stealing	94	5	**Vandalism**	45	5	**Normal**	798	159

Table 4.1. Number of videos in training split and testing split of each category. New categories in UCF Crime dataset are represented with * asterisk character. Since the 33 videos of the training set of the fighting class consisted of newly collected videos, we presented the training videos divided by the separator.

For the evaluation of the model, we use UCF Crime data set [1], which consists of 13 anomaly classes. We have added two different anomaly classes to the data set, which are "Molotov Bomb" and "Protest" classes. We also have added 33 videos to fighting class. Number of videos in train and test splits of UCF Crime dataset are in Table 4.1. In total, we have added 216 videos to the training set, 17 videos to the test set. Test set of UCF Crime data set

(a) abuse

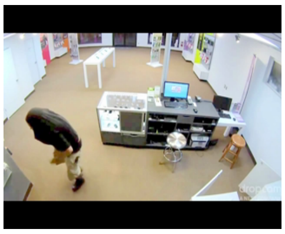
(b) arrest

(c) arson

(d) assault

WHERE'S THE CHANGE? MR. PRESIDENT
STOP SEPARATIONS OF FAMILIES
B-1070

(e) protest



(f) burglary

(g) explosion

(h) fighting

(i) molotov bomb

(j) road accidents

(k) robbery

(l) shooting

(m) shoplifting

(n) stealing

(o) vandalism

Figure 4.6. Snapshots from classes of UCF Crime v2 Dataset

has temporal annotations and classifications. However, training videos of UCF Crime data set are classified in video-level and temporal annotations are not provided for the training set. To train models with temporal information, we annotated anomalies of training videos in temporal domain. In order to annotate efficiently, annotators have used seconds as basis and assumed that the frames in a second all belong to the same class. Since we extend the dataset with new anomaly classes and temporal annotations for training videos, we name new version of the dataset as UCF Crime V2.

Since baseline model has been trained without new anomaly classes, we do not evaluate the baseline on UCF Crime v2 but we evaluate it on UCF Crime v1. We investigate effects of window size, number of layer, feature extractor and loss functions on UCF Crime v2 dataset.

4.7. Results

In this section we present and discuss results of evaluation experiments.

4.7.1. Effect of Temporal Sliding Window

Methods	UCF Crime V2		
	F1@10	F1@25	F1@50
ADNet W32-S3-L5	50.78	43.22	32.05
ADNet W64-S5-L6	**58.73**	**52.38**	**40.98**
ADNet W128-S8-L7	55.61	50.04	36.60
ADNet w/o Window	51.62	43.86	32.30

Table 4.2. Comparison of different window sizes (W: window, S: number of stages, L: number of layers in a stage). I3D is feature extractor in the experiments. Evaluation results on normal and abnormal segments together

We start evaluation by showing the effect of temporal sliding window method. We compare different window widths and without window of ADNet in Table 4.2. Experiments in the table have been made on UCF Crime V2 dataset. Number of layers in this experiment is set according to window width in Formula 10. Window size and number of stages might have different values, which are experimented in our ablation study. Table 4.2. presents three window sizes: 32, 64 and 128 clips in a window. Number of stages for each window width are selected by taking into account the best result of that window. Generally, number of stages increases in parallel to window width. As we mentioned before, temporal sliding window augments data during training. Decreasing window size provides better augmentation, but temporal information is lost. There is a trade-off between data augmentation and temporal knowledge. Table 4.2. shows that the best result is achieved with 64 temporal window width. All of the temporal sliding window results are better than straight forward ADNet (w/o Window) in terms of F1@k scores. This means that temporal sliding window improves performance of ADNet in UCF Crime v2 dataset.

4.7.2. Effect of Number of Layers

Methods	UCF Crime V2		
	F1@10	F1@25	F1@50
ADNet W64-S5-L8	53.03	47.03	33.95
ADNet W64-S5-L7	53.82	48.24	34.02
ADNet W64-S5-L6	**58.73**	**52.38**	**40.98**
ADNet W64-S5-L5	48.12	39.95	29.10
ADNet W64-S5-L4	51.35	42.19	28.17
ADNet W128-S8-L9	63.46*	61.78*	53.61*
ADNet W128-S8-L8	53.14	46.03	33.75
ADNet W128-S8-L7	**55.61**	**50.04**	**36.60**
ADNet W128-S8-L6	53.97	46.66	33.54
ADNet W128-S8-L5	63.46*	61.78*	53.61*

Table 4.3. Comparison of number of layers. I3D is feature extractor in the experiments. Evaluation results on normal and abnormal segments together. Asterisks (*) character present zero F1 score at abnormal segments.

We have compared window widths where number of layers is calculated with Formula 10. In this part we discuss effect of number of layers to performance. Table 4.3. shows results of experiments with fixed window widths (64 and 128) and number of stages (5 and 8).

Asterisks (*) character in Table 4.3. means that ADNet produced zero F1 score in abnormal segments for that setting. In other words, no temporal abnormal segments has detected by the method. For this reason the results with zero F1 scores are not meaningful and are not useful for temporal anomaly localization task in a surveillance system. Table 4.3. shows that ADNet with more layers achieves better results until a point. However, a layer number more than the layer number calculated by Formula 10 cause information loss as mentioned in Section 4.2. Thus, accuracy drops when layer number is greater than 6, which is the value for window size of 64 according to Formula 10. Similarly in experiments with window size of 128, accuracy drops when layer number is greater than 7 which is the value calculated with Formula 10.

4.7.3. Effect of Loss function

Methods	UCF Crime V2			
	Abnormal Segments		Normal Segments	
	F1@10	F1@25	F1@10	F1@25
ADNet (MSE)	29.00	19.33	**71.23**	**66.44**
ADNet (MSE+AD)	**32.16**	**20.70**	56.34	50.54

Table 4.4. Comparison of loss functions. I3D is feature extractor in the experiments.

We propose the AD loss function, which tries to maximize distance between hard pairs from opponent classes. Table 4.4. shows results for MSE loss and MSE and anomaly detection (AD) loss together. In these experiments, window width is 64, number of stages is 5, and number of layers is 6. The parameters are selected based on the previous experiments. λ parameter in Formula 12 controls contribution of AD loss to total loss, λ is set to 0.5 in these experiment. The results show that MSE+AD loss is more successful than MSE loss at abnormal segments, but MSE+AD loss is worse than MSE loss at normal segments. For this reason we use MSE loss only in other experiments.

4.7.4. Effect of Feature Extractor

Methods	UCF Crime V2			
	Abnormal Segments		Normal Segments	
	F1@10	F1@25	F1@10	F1@25
ADNet [I3D]	29.0	19.33	**71.23**	**66.44**
ADNet [TSM]	**33.50**	**22.78**	61.71	57.52

Table 4.5. Comparison of different spatio-temporal feature extractors

We have experimented on two different spatio-temporal feature extractors. In the previous experiments, we used I3D [28] as the feature extractor. Temporal Shift Module (TSM) [31] is more a efficient action recognition network than I3D network while performance on Kinetics [32] dataset is similar. To extract features from video clips, we have used TSM as an alternative to I3D. Table 4.5. shows the results for I3D and TSM networks, where parameters of ADNet are as follows, window width is 64, number of stages is 5 and number of layers is 6. Although I3D and TSM get different sized input clips and produce outputs in different formats, this experiment shows us that ADNet can utilize different feature extractor networks. As a result of this experiment, we observed that I3D features is more useful in ADNet network for segmenting normal events while TSM features is better for segmenting abnormal events.

4.7.5. Comparison with State-of-the-Arts

Temporal annotations of training set of UCF Crime data set has not been available until our study. Baseline model [1] has not been trained on temporarily annotated train set. In order to compare ADNet with models trained on temporarily annotated training set, we have trained two models. First model is Multi Layer Perceptron (MLP) with 3 layers as in baseline network [1] which generates predictions clip-wise. MLP Model accepts features extracted from C3D network as in baseline study. Second model is Encoder Decoder Temporal Convolutional Network (ED-TCN) proposed in [36] with temporal sliding window method as

Methods	UCF Crime v1								
	Abnormal Segments			Normal Segments			All Segments		
	F1@10	F1@25	F1@50	F1@10	F1@25	F1@50	F1@10	F1@25	F1@50
Baseline Network [C3D] [1]	4.13	1.65	0	63.27	56.36	46.54	45.20	39.64	32.32
MLP [C3D]	7.34	1.86	0.65	65.96	63.15	54.75	49.40	44.83	38.15
ED-TCN [I3D] [36]	21.18	12.63	4.88	61.60	53.59	36.71	47.81	39.61	25.85
ADNet [I3D] (Ours)	**28.32**	**18.71**	**9.44**	**71.23**	**66.44**	**55.48**	**58.16**	**51.85**	**41.29**

Table 4.6. Performance comparison of state-of-the art methods

in Section 4.2. ED-TCN model accepts features extracted from I3D network. According to the results presented in Table 4.6., our proposed model achieves better scores than baseline network and other models in all categories on UCF Crime v1 test set. While baseline model achieves **4.13 F1@10** score at abnormal segments, our model achieves **28.32 F1@10** score. Window width, number of stages and number of layers of ADNet in Table 4.6. are 64, 5, and 6, respectively.

5. ANOMALY DETECTION BY OBJECT RELATIONS

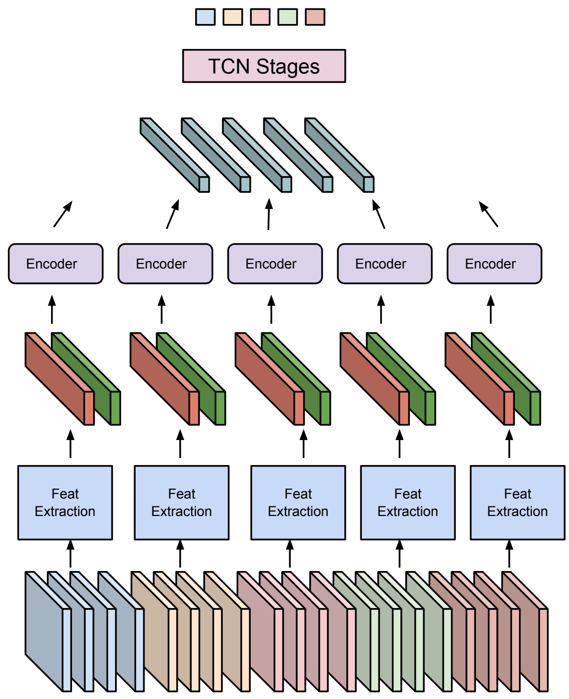


Figure 5.1. ADOR Architecture

UCF-Crime dataset anomalies are complex circumstances which can not be identified with only action knowledge. Most of the anomalies are more complex than actions in action recognition datasets. Additionally, temporal action localization videos comprise simpler circumstances than anomalies in UCF-Crime dataset. As an anomaly example from UCF-Crime, robbery of a motorcycle can not be distinguished easily from driving motorcycle by owner. Objects and object relations in scene are helpful to find some anomalies defined in UCF-Crime dataset. Also static images are not enough to detect video anomalies in UCF-Crime dataset. Object relations are obtained by using an image captioning method [61] in which a transformer network accepts object features extracted from Faster R-CNN. Therefore, in our proposed ADOR model, information from object relations and spatio-temporal features is fused with cross-attention layers. ADOR model consists of three parts to localize anomalies in temporal space, as in Figure 5.1. Spatio-temporal feature extraction from video clips and object features extraction from center frame of the clip is the first part. Encoding and fusing features is the second part. The last is for localizing anomalies from encoded features by staged temporal convolutions.

We explain parts of ADOR model in rest of the chapter. Feature Extraction is explained in Section 5.1., Encoding action features and object features with cross connections is explained in Section 5.2. and Temporal Anomaly Detection is reviewed in Section 5.3. Then we investigate ADOR model by some ablation studies and share evaluation results on UCF Crime dataset in Section 5.5.

5.1. Feature Extraction

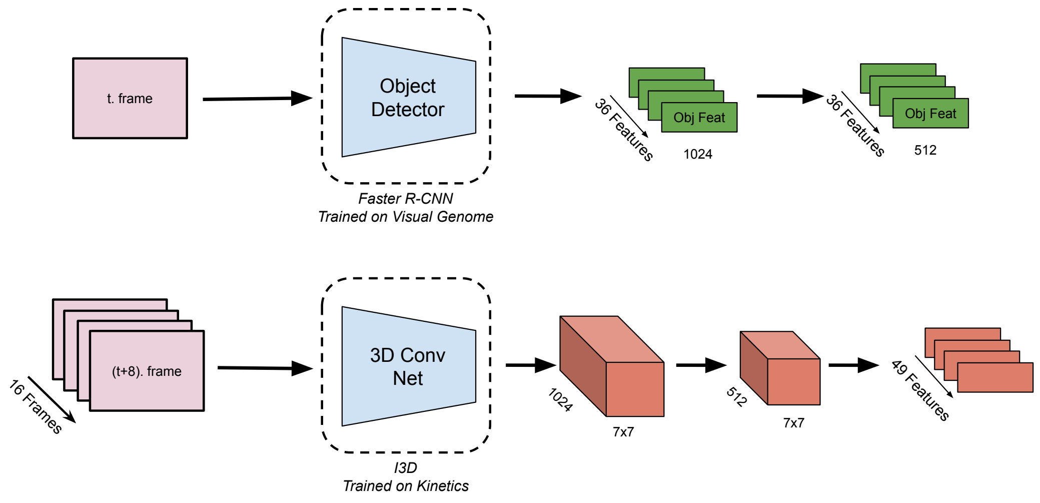


Figure 5.2. ADOR Input Preperation

In the first step, action features and object features are extracted from a video clip and center frame of the video clip, respectively. The feature extraction networks are pre-trained on task specific datasets, these datasets are Visual Genome object detection dataset and Kinetics-400 action recognition dataset. Pre-trained Spatio-temporal CNN I3D [28] accepts clips of 16 consecutive frames to extract the feature. Object detection and object feature extraction network Faster R-CNN works on static images. In order to process object relations and action features together we extract object features from center frame of the clip as in Figure 5.2.

Action features are extracted from video frames $f_{1:T} = (f_1, f_2, ..., f_T)$ where T is number of frames of a video. A video clip $c_i = (f_{i*s-7}, ..., f_{i*s+8})$ where i is clip order, s is temporal stride between video clips, is forwarded to spatio-temporal CNN. Also, center frame of the clip f_{i*s} is forwarded to object detection network. The features of objects with an object classification score higher than 0.2 are pooled with the RoI Pool layer from feature map of object detector's backbone. Extracted features, that have same size because of RoI Pooling layer, are average pooled to generate 2048 dimensional vectors from 7 x 7 x 2048. If number of objects are higher than 36, low scored objects are eliminated until 36 objects left.

Spatio-temporal features X_{st} extracted from video clips using I3D have 7 x 7 x 1024 dimensions. These features are down-scaled to X^*_{st} with 7 x 7 x 512 shape by learned 1 x 1 convolution. Flattened and transposed video clip features $X_{action} = X^{*T}_{st}$ with 49 x 512 shape are ready to forward encoder.

5.2. Object Relations and Scene Action Encoding with Cross Attention

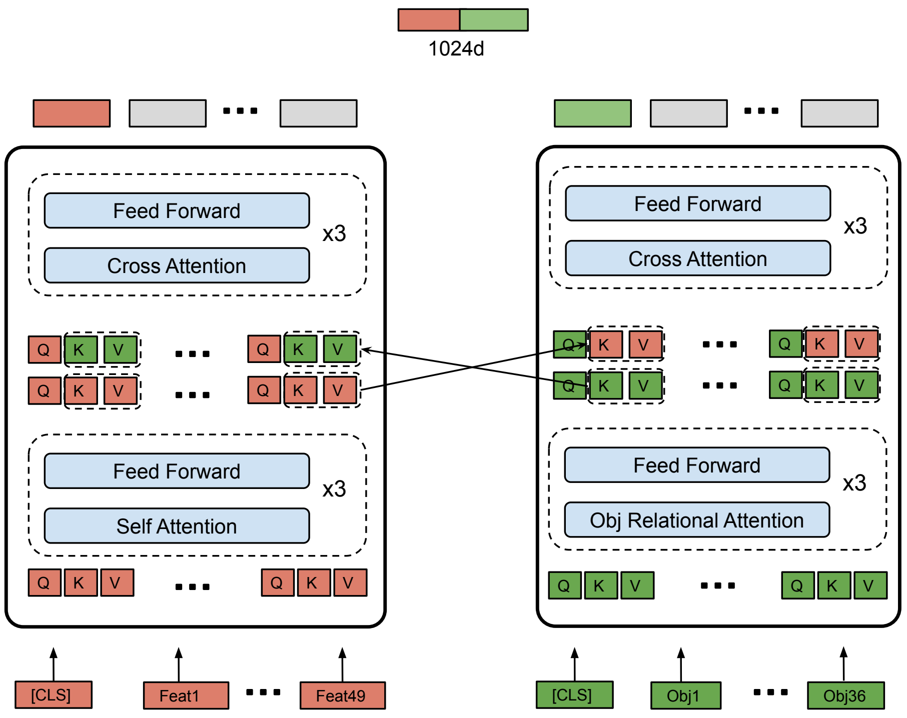


Figure 5.3. ADOR Encoder Architecture

In the second step, anomaly encoder network encodes extracted action and object features. Each video clip is represented with 1024d feature obtained with ADOR encoder. Action encoder and object relation encoder constitutes ADOR encoder. The encoders process the feature sets. Cross-attention layers in the encoders achieve action and object relation knowledge fusion. As we mentioned before, encoders generate output vector for each input vector in input sequence. To classify video clips we append classification embedding vectors to front of the extracted feature sets for video clips. As a result, action encoder accepts $(x^t_{[cls_a]}, x^t_{action_0}, ..., x^t_{action_{49}})$ sequence and outputs $(e^t_{[cls_a]}, e^t_{action_0}, ..., e^t_{action_{49}})$ sequence. Also input sequence of object relation encoder is $(x^t_{[cls_o]}, x^t_{obj_0}, ..., x^t_{obj_{36}})$, output sequence of the encoder is $(x^t_{[cls_o]}, x^t_{obj_0}, ..., x^t_{obj_{36}})$. To obtain 1024d clip feature, we concatenate 512d feature vectors associated with *[CLS]* tokens as in Formula 14.

$$z^t = concat(e^t_{[cls_a]}, e^t_{[cls_o]}) \tag{14}$$

Positions of action features in sequence are specified with positional encodings as in Formula 1. Positions of object features are stated with relative positions between other objects in the frame with Formula 4. Obtained embedding with relative position to reference object is utilized to get attention weight to relative object as we mentioned in section 2.7. Since *[CLS]* token is not object like others in the input sequence of object relation encoder, the bounding box of the token does not exist. In order to use Formula 4 without modification, we can use imaginary box for *[CLS]* like as centered on [0, 0] with zero width and height. If we accept the bounding box of *[CLS]* token as an artificial box, relations between token and objects would be differ by Formula 4. Also it would hurts learned embedding matrix. Since spatial distance between the token and objects do not exist, we do not use relational distance for *[CLS]* token in Formula 15.

$$w_{ab} = \begin{cases} \frac{exp(\frac{q_a * k_b^T}{\sqrt{d_{model}}})}{\sum_{l=1}^{N} exp(\frac{q_a * k_b^T}{\sqrt{d_{model}}})}, & \text{if } [CLS] token \\ \frac{obj\_rel(a,b) + exp(\frac{q_a * k_b^T}{\sqrt{d_{model}}})}{\sum_{l=1}^{N} obj\_rel(a,l) + exp(\frac{q_a * k_b^T}{\sqrt{d_{model}}})}, & \text{if } otherwise \end{cases} \tag{15}$$

Before fusing features from different sources, encoding features by attention layers in themselves are achieved with the first half part of the encoders. Information exchange between encoders starts at second half of the encoders. Key K and value V vectors are passed from one encoder to another encoder to apply cross attention as in Formula 16. Cross attention layers do not use spatial relation between objects as in object relation attention layer, due to lack of boxes of action features.

$$\begin{aligned} Y_{action}^{i+1} &= Softmax(\frac{Q_{action}^{i} * (K_{obj}^{N/2})^T}{\sqrt{d_{model}})}) * V_{obj}^{N/2} \\ Y_{obj}^{i+1} &= Softmax(\frac{Q_{obj}^{i} * (K_{action}^{N/2})^T}{\sqrt{d_{model}})}) * V_{action}^{N/2} \end{aligned} \tag{16}$$

5.3. Temporal Anomaly Localization

Consecutive encoder outputs of a video $(z^0, z^1, ..., z^T)$ are converted to anomaly score by Temporal Convolutional Network which is adapted from MS-TCN as in 4.1. To obtain divergence and data augmentation in temporal space videos are split to fixed size intersected windows. In case of less insufficient outputs in a window, missing clip features are filled with pad features. Information flow from pad features are avoided with masking encoder outputs of padded inputs.

5.4. Implementation Details

We implemented ADOR on PyTorch [68] framework. ADOR was trained by Adam optimizer [67] with 5e-5 learning rate. We used Binary Cross Entropy (BCE) loss to criticize the model. To produce same results with the hyper parameters random seed of PyTorch was set to 123. ADOR is trained with video clips windows that size is 128. Total number of stages in ADOR is 7, we consider the encoder as the first stage, the remaining 6 stages are temporal convolutional network. Each stage output is criticized with the ground truth except the first stage encoder output. We trained and tested model on Nvidia GeForce GTX 1080 Ti GPU.

5.5. Results

5.5.1. Effect of Cross Connections

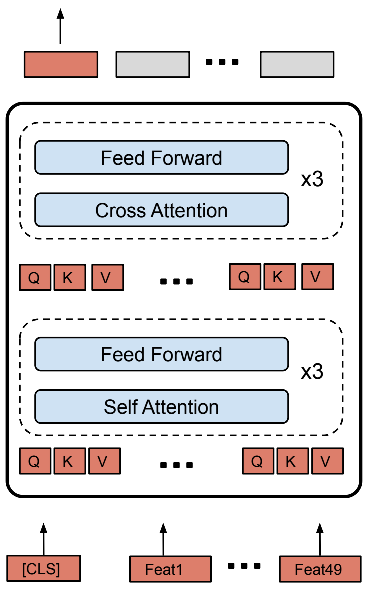


Figure 5.4. Action encoder

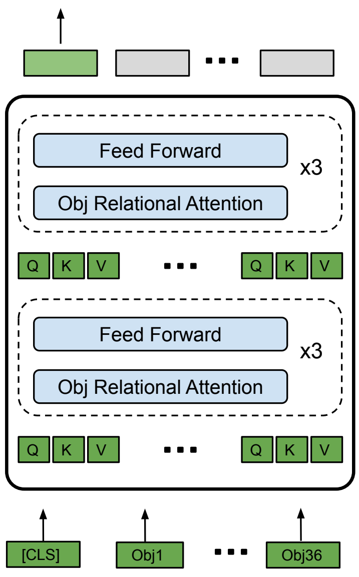


Figure 5.5. Object relation encoder

We investigate effect of cross connections between action and object relation encoders. For this purpose, encoder part of ADOR is replaced with action encoder as in Figure 5.4. and object relation encoder as in Figure 5.5. separetely. Table 5.1. reveals that fusion of action and object relation encoders with cross connections provides better temporal segmentation. The rest of the architecture is the same except for the encoder to compare results fairly.

Models	UCF Crime V1					
	Abnormal			Normal		
	F1@25	F1@50	F1@75	F1@25	F1@50	F1@75
Action Encoder	9.52	3.03	0.43	**71.04**	47.96	18.91
Object Encoder	7.98	0.61	0.61	65.13	51.32	34.52
ADOR	**22.99**	**10.34**	**3.45**	66.43	**52.63**	**37.89**

Table 5.1. Experimental results of the model using only action encoder instead of ADOR encoder, model using only object relation encoder and model with ADOR encoder consisting of cross-connections

5.5.2. Effect of Window Size

Models	UCF Crime V1		
	F1@25	F1@50	F1@75
ADOR (64 window width)	50.32	36.83	27.71
ADOR (128 window width)	53.03	39.77	26.84
ADOR (512 window width)	**56.68**	**42.21**	**29.98**

Table 5.2. Scores of ADOR with different window sizes

ADOR utilizes sliding window approach similar to ADNet. Since windows larger than 128 elements do not fit in GPU memory, we cannot train models with windows larger than 128. However models can infer on different sized windows than training window. In order to evaluate ADOR with larger windows, we employed the model which is trained with 128 size windows. The results in Table 5.2. show that ADOR produces better results with larger windows in contrast to ADNet.

Models	UCF Crime V1			
	F1@25	F1@50	F1@75	AUC
ADOR (2 stages)	34.69	24.93	16.81	**78.95**
ADOR (7 stages)	**56.68**	**42.21**	**29.98**	70.57

Table 5.3. Experiment results on UCF Crime V1 with models which consists of different number of stages. AUC scores shared to show the impact of clip-by-clip metric inefficiency.

5.5.3. Effect of Temporal Metric

As we mentioned in section 4.5. frame-wise score calculation does not reveal temporal efficiency of videos. Output that contains disconnected parts of a segment as in Figure 5.6. is not useful for video understanding. Table 5.3. shows that higher AUC score does not provides better segment based F1 score. Temporal consistencies of ADOR with 2 stages and ADOR with 7 stages are shown in Figure 5.6. As we mentioned before, we consider ADOR encoder as the first stage, the remaining stages are temporal convolutional network stages.

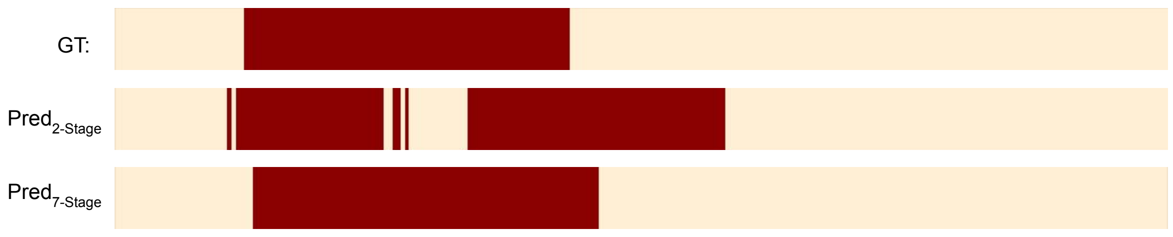


Figure 5.6. Temporal results of 2-stages ADOR and 7-stages ADOR for Explosion 10 in UCF Crime Dataset

5.5.4. Class Scores

Performance of models on classes differs, for the reason of that we compare class performance by training models on each isolated class data.Since F1 scores in test set are greater than ADNet experiments in Section 5.5., we calculated F1 scores with higher Intersection over Union (IoU) percentages to compare results more effectively. In order to evaluate fairly the experiments was made with same spatio-temporal feature extractor (I3D), same temporal resolution and same clip length. Also seed number of random in PyTorch is set to fixed number in class based model comparison experiments to avoid injustice. ADOR achieves better score than ADNET in 9 classes in F1@25, 8 classes in F1@50, 7 classes in F1@75 metric as in Table 5.4.

Classes	UCF Crime V2					
	ADOR			ADNET		
	F1@25	F1@50	F1@75	F1@25	F1@50	F1@75
Abuse	**40.00**	**20.00**	**13.33**	0.00	0.00	0.00
Arrest	**35.71**	**28.57**	**7.14**	25.29	17.65	0.00
Arson	**52.00**	26.00	16.00	47.76	**29.85**	**17.91**
Assault	47.06	23.53	11.76	**58.82**	**35.29**	11.76
Protest	**50**	**40**	0	48.28	27.59	**6.90**
Burglary	**42.67**	**24.00**	**21.33**	38.64	22.73	15.91
Explosion	**61.33**	**49.33**	24.00	58.27	47.24	**26.77**
Fighting	**60.00**	**40.00**	**33.33**	47.06	17.65	11.76
Molotov Bomb	52.17	31.88	**20.29**	**62.50**	**46.87**	18.75
Road Accidents	**73.44**	**57.81**	**29.69**	66.67	54.70	29.06
Robbery	61.11	38.89	16.67	**76.92**	**53.85**	**30.77**
Shooting	52.38	38.10	15.87	**57.32**	**40.24**	**23.17**
Shoplifting	36.97	16.81	6.72	**41.03**	**17.95**	**7.69**
Stealing	**57.14**	**42.86**	**14.29**	40.00	40.00	5.71
Vandalism	45.16	**19.35**	6.45	**48.48**	18.18	**12.12**

Table 5.4. F1 scores of models trained with only that class's data for each class.

5.5.5. Comparison with State-of-the-Arts

We compare ADOR with ADNet, baseline study [1] and models trained with temporal annotations. As we mentioned in Section 4.7.5., baseline network was not trained with temporal annotations. For this reason we trained MLP model with C3D features, ED-TCN model with I3D features to compare our models. The results in Table 5.5. shows that while ADNet achieves **28.32 F1@10** score on abnormal segments ADOR achieves **33.94 F1@10** score.

Methods	UCF Crime v1								
	Abnormal Segments			Normal Segments			All Segments		
	F1@10	F1@25	F1@50	F1@10	F1@25	F1@50	F1@10	F1@25	F1@50
Baseline Network [C3D] [1]	4.13	1.65	0	63.27	56.36	46.54	45.20	39.64	32.32
MLP [C3D]	7.34	1.86	0.65	65.96	63.15	54.75	49.40	44.83	38.15
ED-TCN [I3D] [36]	21.18	12.63	4.88	61.60	53.59	36.71	47.81	39.61	25.85
ADNet [I3D] (Ours)	28.32	18.71	9.44	71.23	66.44	**55.48**	58.16	51.85	41.29
ADOR (Ours)	**33.94**	**26.06**	**11.52**	**75.03**	**68.04**	54.20	**63.49**	**56.68**	**42.21**

Table 5.5. Performance comparison of state-of-the art methods

6. CONCLUSION

We proposed a temporal Anomaly Detection Network (ADNet) and Anomaly Detection Network by Object Relations (ADOR) , which enables to localize anomalies in videos with temporal convolutions. ADNet utilizes spatio-temporal features extracted from pre-trained action recognition model, ADOR employs both spatio-temporal features and object features extracted from pre-trained object detection network. In our knowledge, ADNet is the first to formulate the video anomaly detection problem as similar to the action localization problem, is the first approach that utilizes both object relations and action information in video anomaly detection. We also introduced AD loss function, which enabled to have better detection performance in ADNet for abnormal classes. We evaluated and discussed the effects of model parameters, which are window width, number of layers, number of stages, feature extractor, and loss functions for ADNet.

We also extended the UCF Crime anomaly dataset with two additional anomaly classes and temporal annotations of training videos. Extensive evaluations of the model shows that the model has promising results on real world anomaly videos. Window based operation of the model allows processing of online video streams. We also investigated the evaluation metrics in terms of measuring anomaly detection performance. Since F1@k does not penalize minor shifts and does punish short false positive temporal segment predictions, we concluded that F1@k metric is better than AUC metric for measuring anomaly detection performance.

In the future works, scene context information and relations between objects can be utilized to improve anomaly detection performance. Additionally, associations between action and object information might improve anomaly detection performance especially in human involved anomalies, such vandalism, fighting, stealing, etc. ADOR use action information extracted from the whole scene but does not have information about relations between objects and action information of the objects. ADNet can be studied with different action recognition methods or with more detailed action information from scenes. Similarly, ADOR can be studied with different object detectors, action detectors and its transformer architecture can be extended further to fuse object and action information better.

REFERENCES

[1] Waqas Sultani, Chen Chen, and Mubarak Shah. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6479–6488. **2018**.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. **2016**.

[3] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, **2016**.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008. **2017**.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, **2018**.

[6] Ramin Mehran, Alexis Oyama, and Mubarak Shah. Abnormal crowd behavior detection using social force model. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 935–942. IEEE, **2009**.

[7] Vijay Mahadevan, Weixin Li, Viral Bhalodia, and Nuno Vasconcelos. Anomaly detection in crowded scenes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1975–1981. IEEE, **2010**.

[8] Cewu Lu, Jianping Shi, and Jiaya Jia. Abnormal event detection at 150 fps in matlab. In *Proceedings of the IEEE international conference on computer vision*, pages 2720–2727. **2013**.

[9] Amit Adam, Ehud Rivlin, Ilan Shimshoni, and Daviv Reinitz. Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE transactions on pattern analysis and machine intelligence*, 30(3):555–560, **2008**.

[10] Weixin Luo, Wen Liu, and Shenghua Gao. A revisit of sparse coding based anomaly detection in stacked rnn framework. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 341–349. **2017**.

[11] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K Roy-Chowdhury, and Larry S Davis. Learning temporal regularity in video sequences. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 733–742. **2016**.

[12] Trong-Nguyen Nguyen and Jean Meunier. Anomaly detection in video sequence with appearance-motion correspondence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1273–1283. **2019**.

[13] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection–a new baseline. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6536–6545. **2018**.

[14] Mahdyar Ravanbakhsh, Moin Nabi, Enver Sangineto, Lucio Marcenaro, Carlo Regazzoni, and Nicu Sebe. Abnormal event detection in videos using generative adversarial nets. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1577–1581. IEEE, **2017**.

[15] Ryota Hinami, Tao Mei, and Shin'ichi Satoh. Joint detection and recounting of abnormal events by learning deep generic knowledge. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3619–3627. **2017**.

[16] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, **1989**.

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, **1998**.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, **2012**.

[19] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, **2012**.

[20] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*. **2010**.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, **2014**.

[22] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, **2017**.

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520. **2018**.

[24] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324. **2019**.

[25] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141. **2018**.

[26] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, **2019**.

[27] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497. **2015**.

[28] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308. **2017**.

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9. **2015**.

[30] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint pattern recognition symposium*, pages 214–223. Springer, **2007**.

[31] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7083–7093. **2019**.

[32] 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**.

[33] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5552–5561. **2019**.

[34] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12046–12055. **2019**.

[35] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459. **2018**.

[36] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165. **2017**.

[37] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *International conference on artificial neural networks*, pages 799–804. Springer, **2005**.

[38] Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738. **2013**.

[39] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 780–787. **2014**.

[40] Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3575–3584. **2019**.

[41] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1130–1139. **2018**.

[42] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, **2015**.

[43] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, **2020**.

[44] Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11336–11344. **2020**.

[45] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, **2019**.

[46] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, **2019**.

[47] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, **2019**.

[48] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, **2016**.

[49] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27. **2015**.

[50] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, **2018**.

[51] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, **2017**.

[52] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642. **2013**.

[53] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, **2019**.

[54] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14. Association for Computational Linguistics, Vancouver, Canada, **2017**. doi:10.18653/v1/S17-2001.

[55] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*. **2005**.

[56] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*. **2009**.

[57] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. **2012**.

[58] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, **2016**.

[59] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*, **2018**.

[60] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448. **2015**.

[61] Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares. Image captioning: Transforming objects into words. *arXiv preprint arXiv:1906.05963*, **2019**.

[62] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086. **2018**.

[63] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. **2016**.

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, **2015**.

[65] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, **2014**.

[66] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. *arXiv preprint arXiv:1707.05612*, **2017**.

[67] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, **2014**.

[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035. **2019**.