Context-aware Video Anomaly Detection in Long-Term Datasets

Supplementary Material

7. Architecture and training details

7.1. Architecture details

Context Branch. In the context branch, the MLP consists of two 'Linear-LayerNorm-GELU' blocks and one 'Linear' block for projecting concatenated one-hot encoded context information into a 128-dimensional embedding. The number of heads in the Temporal/Spatial Context Attention Block (TSCAB) is set to 8.

Motion Branch. To calculate the background ratio of each non-overlapping patch for extracting HOF features, we set the flow magnitude thresholds to 0.5 and 1.0 for the benchmark datasets and the WF dataset respectively. We use $k_{mot} = 3$ transformer blocks to extract information from the quantized motion and the error code. For the motion transformer block, the number of heads is set to 8.

Appearance Branch. The spatial downsampling rate of the U-net is 8 and the 'Conv2d' embedding layer uses a size 2 kernel with stride 2 to embed 16×16 spatial patches as tokens for the following transformer blocks. We use $k_{app} = 3$ transformer blocks to extract appearance information.

All three branches generate output of the same size of $\mathbf{h} \in \mathbb{R}^{B \times N \times D}$, where the batch size B is set to 12, the token length N is set to 257 and 193 for the benchmark datasets and the WF dataset respectively, and the feature dimension D is set to 512.

7.2. Details about local level alignment

The pseudo-code for patch-wise local alignment is shown in Fig. 7. For patch-wise local alignment, the representation of each patch treats the rest of patches within the frame as negative samples. During inference, we utilize the patch-wise local alignment quality as the indicator of anomalies. As shown in Fig. 8, a normal frame during inference will have an almost perfect diagonal matrix, indicating most patches have good alignment between the appearance and motion representations. On the contrary, when an anomaly appears in the frame, either unseen motion or unseen appearance will make the local alignment quality decrease drastically. Through evaluating the patch-wise local alignment quality, we can detect context-free anomalies.

8. WF Dataset

In this work, the WF dataset used for investigating video context anomaly detection contains complex scenario dynamics. In Fig. 9, we arrange video frames from the WF dataset in a grid format with each row showing 12 frames with a sampling gap of 2 hours within a day. The activities within the WF dataset vary significantly at different times of the day, and show strong correspondence with the game schedule. The context information for the video is organized as follows: The time of day is indicated on a 0 to 23 hour scale, and the day of the week is denoted by the numbers 0 to 6, representing Monday to Sunday. The game time indicator is set to 1 if the video's timestamp falls within 2 hours before and 3 hours after a game starts. The game schedule is marked similarly to the time of day, with a value of 0 indicating no game. Each context element is represented using one-hot encoding, resulting in a 56-dimensional binary vector through concatenation.

```python
def local_b_align(src_1,src_2, tau):
    '''
    Local level batch wise alignment logit will be [N, B, B]
    Assume both feature representation src_1, src_2 have shape [B,N,C]
    tau: learnable temperature parameter
    '''
    # normalized features
    src_1 = src_1 / src_1.norm(dim=-1, keepdim=True)
    src_2 = src_2 / src_2.norm(dim=-1, keepdim=True)

    # cosine similarity as logits
    logit_scale = tau.exp()

    #calculate inter batch logits [N,B,C]@[N,C,B]-> [N,B,B] across batch contrastive
    logits_per_src1 = logit_scale * torch.bmm(src_1.permute(1,0,2),src_2.permute(1,2,0))
    logits_per_src2 = logits_per_src1.permute(0,2,1)

    return logits_per_src1,logits_per_src2
def local_p_align(src_1,src_2, tau):
    '''
    Local level patch wise alignment logit will be [B, N, N]
    Assume both src_1, src_2 has shape [B,N,C]
    tau: learnable temperature parameter
    '''
    # normalized features
    src_1 = src_1 / src_1.norm(dim=-1, keepdim=True)
    src_2 = src_2 / src_2.norm(dim=-1, keepdim=True)

    # cosine similarity as logits
    logit_scale = tau.exp()

    #calculate intra frame logits [B,N,C]@[B,C,N]-> [B,N,N] across local patch contrastive
    logits_per_src1 = logit_scale * torch.bmm(src_1,src_2.permute(0,2,1))
    logits_per_src2 = logits_per_src1.permute(0,2,1)

    return logits_per_src1,logits_per_src2
```

Figure 7. Pytorch-style pseudo code for calculating local alignment logits. **Top:** the pseudo code of calculating batch-wise local alignment; **Bottom:** the pseudo code of calculating patch-wise local alignment.

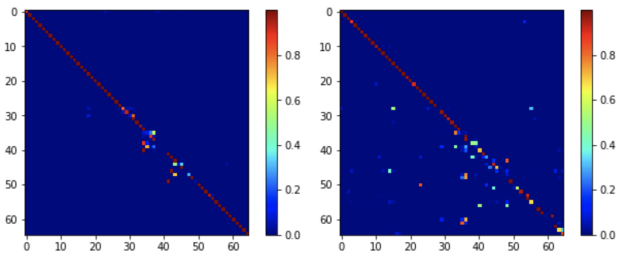

Figure 8. Visualization of patch-wise local alignment results. The x-y axis represents the spatial index of the token. **Left:** A normal frame result in the Ped2 dataset; **Right:** An abnormal frame result in the Ped2 dataset.

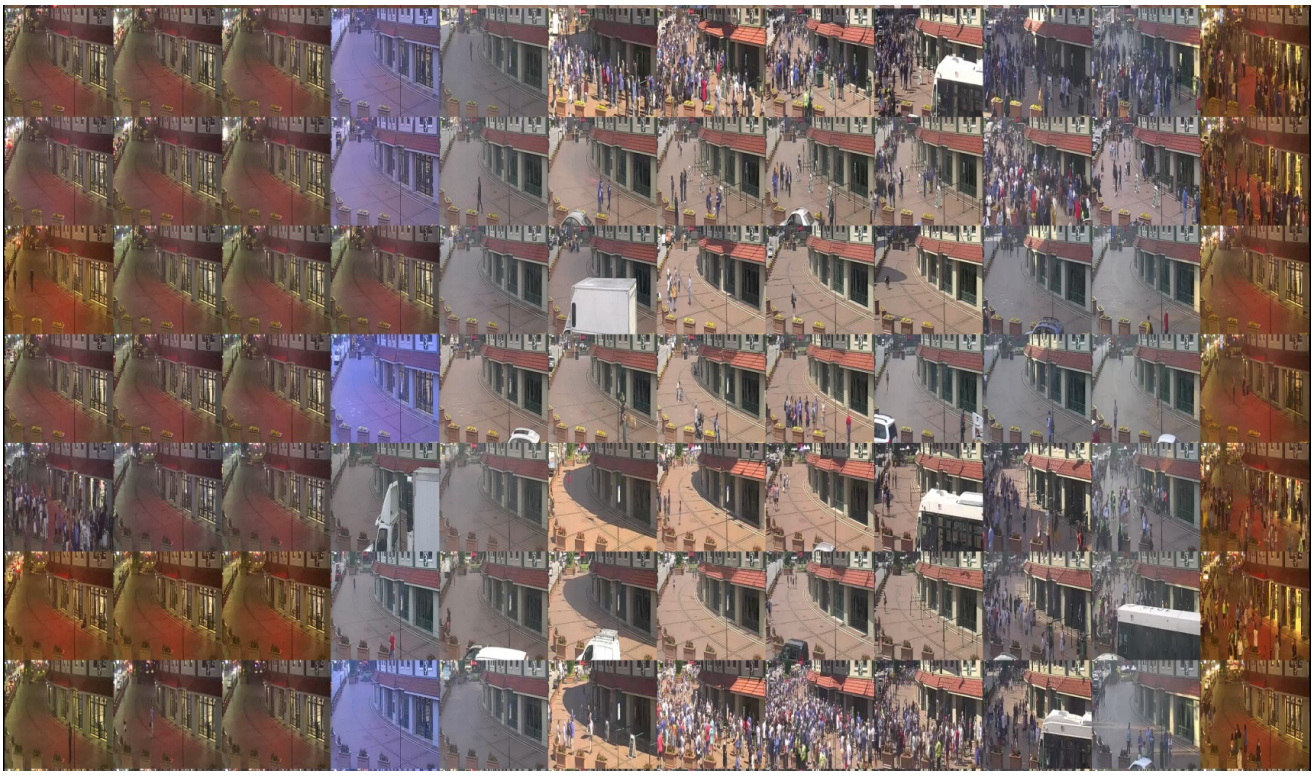

Figure 9. Sample day grids from the WF Dataset. Each row represents a snapshot of a day containing re-scaled frames that are evenly sampled every 2 hours from Wrigley Field Stadium (Best view in color).

Original Context	**Modification Intention**	**Altered Context Anomaly**
10 4 0 **14**	day game morning → non-game day morning	10 4 0 **0**
16 4 1 14	during day game → time ahead to 2 hours earlier	**14** 4 1 14
13 0 **0 0**	non-game day → game starts at 14:00	13 0 **1 14**
22 5 **1 19**	night game people exiting the stadium → cancel game schedule	22 5 **0 0**

Table 4. Sample pseudo anomaly context alteration. The context information from the original context and altered context are listed as: time of day, day of the week, game indicator, game schedule respectively.

The pseudo-contextual anomalies that are introduced in Sec. 4.1 are achieved through altering the original context of normal videos with different intentions to mimic various context-anomalous scenarios. We provide examples of alteration from the pseudo-contextual anomalies in Table 4.