KL-CPD for video data
(Selected Topics in Data Science 2022 Course)

Evgenia Romanenkova [1] **Dmitriy Ermilov** [1] **Alexander Lukashevich** [1]

Abstract

KL-CPD, a novel kernel learning framework for time series CPD that optimizes a lower bound of test power via an auxiliary generative model. With deep kernel parameterization, KL-CPD endows kernel twosample test with the data-driven kernel to detect different types of change-points in real-world applications. However, it requires a lot of memory and resources to work. In this project, we try to improve performance of this method in terms of quality and efficiency to make it applicable to video surveillance.

1. Introduction

Change points are abrupt disorders in the distribution of sequential data. The change-point detection (CPD) model signals the appearance of such changes in data streams. Historically, there were many rigid theoretical results on the data-driven CPD approaches, and, in many cases, precise change detection criteria exist. However, these methods are not suitable for complex video data because of the absence of adequate data representation learning. The majority of existing representation-based methods are based on non-principled procedures and consider a CPD problem as a binary-classification problem close to anomaly detection area(Oord et al., 2018; Sultani et al., 2018). In contrast, the KL-CPD method proposed in the paper (Chang et al., 2018) is based on both classic kernel CPD methods (Aminikhanghahi & Cook, 2017; Truong et al., 2020) that use Maximum Mean Discrepancy (MMD) (Li et al., 2015), but a trainable generative model in its core for learning kernels parameters. However, such a model requires many resources for training in case of complex video data. Due to its architecture similar to GAN-based approaches (Goodfellow et al., 2014), we need to encode really high-dimensional sequences and decode them in original space.

*Equal contribution [1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Evgenia Romanenkova <Evgenia.Romanenkova@skoltech.ru>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Our project aims to overcome this technical issue and compare different approaches for reducing neural network space parameters without great changes in the model's architecture and preserving its quality. In particular, our main claims are the following:

- We compare the different common approaches for video processing, such as reducing the video quality or size, using pre-trained encoders with frozen and trainable parameters as well as adding projection layers in KL-CPD models;
- In addition, we consider reducing the number of model's parameters by pruning and/or replacing the core model's layers by their low-rank approximations;
- Finally, we train KL-CPD with a stochastic mask that leads to the sparsity of parameters space.

2. Related work

When talking about the cost of neural networks, the number of parameters is surely one of the most widely used metrics, along with FLOPS (floating-point operations per second).

2.1. Video compression

On the side of video compression, which would significantly reduce the number of trainable parameters and FLOPS, there exist two streams: *image compression* and *video compression*.

Image Compression
Among *image* compression, there are classical approaches, e.g., JPEG compression standard linearly maps the pixels to another representation by using Discrete Cosine Transform (DCT) and quantizes the corresponding coefficients before entropy coding (Wallace, 1992). Aside from classical well-known compression algorithms, recently, Deep Neural Network (DNN) compression algorithm became popular. Such approaches usually are based on Recurrent Neural Networks (RNN) (Toderici et al., 2017; 2016) or Convolutional Neural Networks (CNN) (Ballé et al., 2017; 2018). In the case of RNN aims to sequentially make a forward pass to

obtain a latent vector that can be used as a low-dimensional representation of an input image. Similarly, CNN autoencoder models are adopted for image compression where the latent representation, i.e., the encoder output, is used as a low-dimensional representation. Moreover, Manifold Learning approaches are used for image compression to retrieve the estimation of the manifold on which the images lie (Pless & Souvenir, 2009). Such approaches typically include Isometric Feature Mapping (ISOMAP) or Locally Linear Embedding (LLE).

Video Compression

Similarly, for *video* compression, as an ordered sequence of images, there are, similarly to images, classical, DNN and Manifold Learning approaches. Among classical ones are H.265 (Sullivan et al., 2012) and H.264 (Wiegand et al., 2003) standards. These algorithms follow a predictive coding paradigm. Specifically, the major steps of these algorithms are (Lu et al., 2019): 1) Motion estimation; 2) Motion compensation; 3) Transform and quantization; 4) Inverse transform; 5) Entropy coding; 6) Frame reconstruction. From the DNN field, authors in (Lu et al., 2019) added compression into step one and used different CNNs for each step of the classical algorithms. The loss function was constructed with reconstruction error and the term that penalized bit rate for motion encoding and residual representation. Finally, an application of Manifold Learning is presented in (Nie et al., 2011). The authors proposed using ISOMAP for video compression for further similarity measures between videos.

2.2. Model compression

On the side of model compression, the number of trainable parameters and FLOPs can be reduced by pruning redundant connections in the neural network or by using weights of the neural network in a low-rank format.

Weights Pruning

Models that perform well in the computer vision domain are often over-parameterized and contain redundant connections. Therefore, it is intuitive to reduce computational model complexity by removing parameters themselves. Pruning can be learned during training using regularization or as a post-training step.

One training-based method group focuses on learning a pruning mask through auxiliary parameters during training. This kind of method aims at considering pruning as an optimization problem that tends to minimize both network supervised loss and penalty of auxiliary parameters that are multiplied with their corresponding parameter during the forward pass(Guo et al., 2016).

Another group of training-based methods apply various kinds of penalties to weights themselves to make them progressively shrink toward zero to enforce sparsity(Nowlan & Hinton, 1992). Various methods apply different regularizations on top of the weight decay to increase further the sparsity: using L_1 norm(Liu et al., 2017), LASSO (Least Absolute Shrinkage and Selection Operator)(Gao et al., 2019), Variational Dropout(Blum et al., 2015), etc.

Post-training pruning methods prune weights whose absolute value is the smallest. One straightforward way to apply such pruning in a structured way is to remove neurons in linear layers or channels in convolutional layers depending on their norm (L_1 or L_2 for example)(Zhuang et al., 2018).

Weights in Low-Rank Format

Low-rank methods can replace weights of the neural network layer with its low-rank factorization. For example, linear layers or point-wise convolution (1×1) can be factorized using SVD into two sequential linear layers or point-wise convolutions. In this case, more information is preserved than with pruning (before fine-tuning). It is also possible to train a neural network with weights in low-rank format from scratch. Such an approach forces neural network parameters optimization over a smaller number of parameters.

Input in the considered KL-CPD model has a high-dimensional format (frames, channels, height, width). Therefore, it seems natural to apply further transforms using the Tensor Contraction Layer(TCL) and/or Tensor Regression Layer(TRL)(Kossaifi et al., 2020) rather than flattening output and using linear layers, for example, in GRU. In TCL, each input dimension is multiplied with its own matrix, resulting in fewer parameters and computational complexity for linear transformation. In TRL, the inner product between input and layer weight is calculated, allowing learning of more complex interconnections than in TCL.

Another option is to use Tensor Train (TT) format(Oseledets, 2011) for RNN similar to TT-RNN in (Zhao et al., 2021). Since video features are usually in a high-dimensional space, it leads to huge feature-to-hidden mapping matrices in the RNN model. TT embedding layer averts computations with large feature-to-hidden matrices. Therefore, the input can be efficiently processed using a layer which takes a smaller number of parameters.

3. Proposed Approach and Motivation

The vital drawback of applying KL-CPD to video anomaly detection is the huge dimensionality of input data. Such an issue results in a large number of parameters in the KL-CPD model. Thus, making the training of this model intractable or even not possible due to the limited amount of memory on GPUs.

To this end, we propose several approaches to reduce the

number of trainable parameters in the model and provide motivation behind each approach.

The input video data is fed into the model at 30 frames per second (fps) and each frame is a $240 \times 320 = 76800$ pixels image. Obviously, it is impractical to feed such data into model directly.

We separate our research into three main tracks: preliminary prepossessing input data to compress its overall number of pixels, selecting the proper encoder for data to input in KL-CPD feature embeddings instead of initial video and general KL-CPD model compression.

In detail, the tasks on the first track are the following:

1. Provide more compression of input sequences of images or video with classical methods (JPEG, H.265/H.264 or DFT/DCT).
2. Provide reduction of input size with applying standard transformations like side scale, crop video, etc.

We also plan to investigate the operating with low-dimensional data:

1. Train our own DNN model to retrieve low-dimension representation of input data (RNN/CNN autoencoder).
2. Extract features from initial data by the pre-trained powerful network such as frame-wise applying of similar to ResNet 2D models (He et al., 2016) or models on 3D convolutions (Feichtenhofer et al., 2019).
3. Experiment with reduction of feature extractor output size as it also has huge dimension for video data. We plan to investigate different projection layers (both trainable and non-trainable) as well as perform dimension reduction for obtained embedding via Manifold Learning techniques (ISOMAP, LLE).

Finally, we'll work on model's compression:

1. Experiment with reduction of model parameters by modeling weights in low-rank format.
2. Experiment with insertion of learnable masks into model to prune parameters in structured way.

Now, let us discuss potential drawbacks of each of the proposed approaches for data compression. We will maintain the same order as above.

1. Classical compression approaches are generally good from a perceptual point of view. However, the flawless of such approaches is hidden among their generality. They are not optimal for specific tasks and don't consider the specificity of the CPD task. The important properties vary from type of video, e.g., for explosions, we expect the CPD model to monitor the intensity of white colours; however, it is not true for car accidents where patterns are much more comprehensive. While compression can not influence intensity, we can lose other essential properties for other types of change points. Similar reasons are suitable for standard transformations and operating with low-dimensional space. So, we should research how different methods affect the overall model's quality.
2. DNN approaches might take a lot of effort and memory to implement. However, such approaches look more promising than the others because of their data-based nature if enough data is provided.
3. Pruning and low-rank methods needs hyper-parameter selection and further model fine-tuning to recover drop in performance. It may take some time to find optimal set of hyper-parameters.

4. Preliminary results

The code reproduced the following results is provided by link.

4.1. Evaluation metrics

Following (Romanenkova et al., 2021), we use the same evaluation strategy for estimating the quality of applied tricks for video change point detection via KL-CPD.

Classification quality metrics. For True Positive (TP), we correctly detect changes no earlier than it appears. True Negative (TN) indicates that we rightly don't detect anything. False Negative (FN) defines a case when we miss the disorder while it appears in the real data. The most significant is the False Positive (FP). The obvious case of FP is predicting changes for normal signals. In addition, we detect the FP when the model predicts change before it appears in a sequence with the real disorder. Based on these metrics, we calculate $F1 = \frac{TP}{TP+0.5(FP+FN)}$.

Delay detection and Time to False Alarm. We want to meet two competing goals: minimize the Detection Delay and maximize the Time to False Alarm. The Delay Detection $(\tau - \theta)^+$ is the difference between the true change point θ and the model's prediction of change point τ. The Time to False Alarm τ is the difference in time between a first false positive prediction if it exists and the sequence start.

Area under the detection curve. Varying s, we obtain different trade-offs between mean detection delay (x-axis)

and mean time to a false alarm (y-axis) and get the detection curve. We want to find an approach that minimizes the area under this curve.

4.2. Video dataset details

Following (Romanenkova et al., 2021), we dataset with real-world 240×320 RGB explosion videos and markup provided in the article for our research.

To generate the dataset for the CPD model, we cut videos on small clips with a given step and frequency. We use 30 frame per second, and the number of frames in the clip 16. For training data, we sample clips with step 5, so there are overlapping sections. For test data, we cut the video on non-overlapping clips. We keep these parameters constant for all our experiments, varying transformations that could be applied to the data (e.g. normalization, scaling etc).

4.3. The original benchmark.

We follow the procedure described in the (Romanenkova et al., 2021) to run KL-CPD for the explosion dataset. In particular, we use 3D CNN (Feichtenhofer et al., 2019) as a feature extractor and add Linear layers with dimensions 100 to encoders and a decoder that makes an embedding size small enough to be stored and processed. We use window size and batch size equal to 4.

4.4. Results on preliminary data preprocessing

The most simple approach to processing the video via the KL-CPD approach is to reduce the initial dimension of the input data. We consider the following preliminary data preprocessing techniques:

1. First of all, we consider a simple reduction of input size by resizing each frame of the video to a smaller one via common resizing from torchvision package.
2. We also evaluate the embedding of each frame via manifold learning methods (ISOMAP and Spectral Embedding),
3. Finally, we train a custom feature extractor, namely CNN that is applied to each frame.

Resizing and Manifold learning. Currently, for all preprocessing experiments, we work on *frame level*, i.e. we apply the method to each frame of the input video. Then, the processed frames are stacked together to form a sequence of multidimensional vectors. The obtained sequence is the input in KL-CPD architecture. During *resizing*, we simply reduced the height by the ratio of target height and the width proportional to the reduction of the height.

For manifold learning experiments, we compress frames into 2D and 3D embedded spaces to reveal if separation can be made easily in those spaces. We evaluate the ISOMAP and SpectralEmbedding models fitted on the frames from the training dataset. Figure 1 shows the obtained low-dimensional representations. The blue points correspond to normal frames, and the orange one corresponds to frames with an explosion. As we can see, there is no tangible structure - "exploded" frames are well hidden beneath ordinary frames.

Table 1. Preprocessing results. TCL-1 from Section 4.5.

Method	TN	FP	FN	TP	F1
Resize 48	241	63	10	1	0.0267
Resize 128	238	64	10	3	0.0750
CNN	**290**	**11**	**6**	**8**	**0.4848**
Original	285	16	8	6	0.3333

Custom CNN feature extractor. The considered custom feature extractor has the following architecture:

- Conv2D with 3 input and 12 output channels, kernel size equals 5, stride equals 2, and dilation equals 2 followed by ReLU activation;
- Conv2D with 6 output channels, kernel size equals 3, stride equals 2, and dilation equals 2 followed by ReLU activation and MaxPool2D over 3 cells;
- Finally, the output is flattened.

The results on experiments with preprocessing are summarized in Table 1. As one can observe, trainable CNN provides the best result among resizing approaches and the original benchmark.

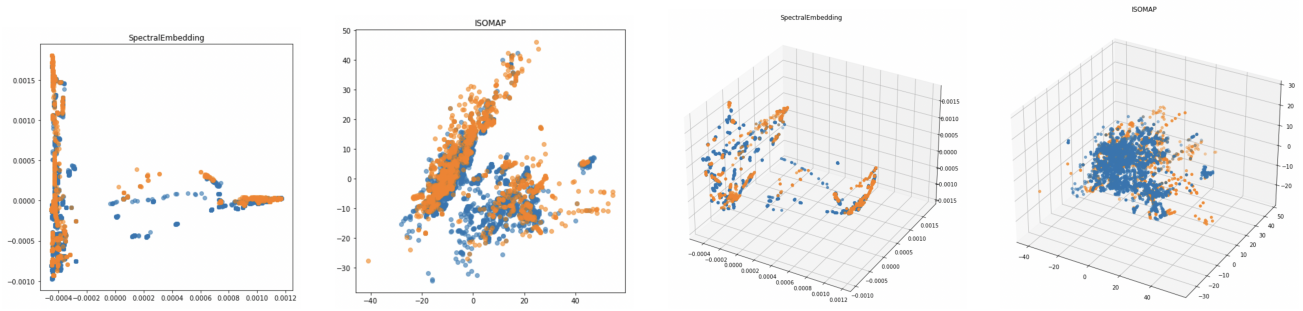
4.5. Results on tensor contraction

In modelling weights in low-rank format experiments, we replace all linear layers in GRU modules in discriminator and generator networks with TCL layers. It means that the input of size (batch size, sequence length, I_1, I_2, I_3) is multiplied with three matrices along with the last three modes to obtain the output of size (batch size, sequence length, O_1, O_2, O_3). It leads to reducing the parameter number in discriminator and generator networks as presented in Table 2.

References

Aminikhanghahi, S. and Cook, D. J. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.

Ballé, J., Laparra, V., and Simoncelli, E. P. End-to-end optimized image compression. 2017.



(a) 2D Spectral Embedding (b) 2D ISOMAP (c) 3D Spectral Embedding (d) 3D ISOMAP

Figure 1. Manifold learning methods' embedding of frames. Frames after the change point (explosion) are coloured with orange, the normal ones — with blue.

Table 2. Results on replacing linear layers to TCL layers in GRU modules. # param denoted number of (trainable) parameters in discriminator and generator networks

Embedding size	Hidden size	# param	Max F1	AUC
100	16	4M	0.133	1.64
(32, 8, 8)	(16, 4, 4)	0.27M	0.112	1.58

Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N. Variational image compression with a scale hyperprior. 2018.

Blum, A., Haghtalab, N., and Procaccia, A. D. Variational dropout and the local reparameterization trick. In *NIPS*, 2015.

Chang, W.-C., Li, C.-L., Yang, Y., and Póczos, B. Kernel change-point detection with auxiliary deep generative models. In *International Conference on Learning Representations*, Vancouver, Canada, 2018. ICLR.

Feichtenhofer, C., Fan, H., Malik, J., and He, K. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6202–6211, 2019.

Gao, S., Liu, X., Chien, L.-S., Zhang, W., and Álvarez, J. M. Vacl: Variance-aware cross-layer regularization for pruning deep residual networks. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 2980–2988, 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Guo, Y., Yao, A., and Chen, Y. Dynamic network surgery for efficient dnns. In *NIPS*, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kossaifi, J., Lipton, Z. C., Khanna, A., Furlanello, T., and Anandkumar, A. Tensor regression networks. *ArXiv*, abs/1707.08308, 2020.

Li, S., Xie, Y., Dai, H., and Song, L. M-statistic for kernel change-point detection. *Advances in Neural Information Processing Systems*, 28, 2015.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2755–2763, 2017.

Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., and Gao, Z. Dvc: An end-to-end deep video compression framework. volume 2019-June, 2019. doi: 10.1109/CVPR.2019.01126.

Nie, X., Liu, J., Sun, J., and Zhao, H. Key-frame based robust video hashing using isometric feature mapping. *Journal of Computational Information Systems*, 7, 2011. ISSN 15539105.

Nowlan, S. J. and Hinton, G. E. Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4:473–493, 1992.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Oseledets, I. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33:2295–2317, 2011.

Pless, R. and Souvenir, R. A survey of manifold learning for images. volume 1, 2009. doi: 10.2197/ipsjtcva.1.83.

Romanenkova, E., Zaytsev, A., Zainulin, R., and Morozov, M. Principled change point detection via representation learning. *arXiv preprint arXiv:2106.02602*, 2021.

Sullivan, G. J., Ohm, J. R., Han, W. J., and Wiegand, T. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22, 2012. ISSN 10518215. doi: 10.1109/TCSVT.2012.2221191.

Sultani, W., Chen, C., and Shah, M. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6479–6488, Salt Lake City, Utah, USA, 2018. IEEE.

Toderici, G., O'Malley, S. M., Hwang, S. J., Vincent, D., Minnen, D., Baluja, S., Covell, M., and Sukthankar, R. Variable rate image compression with recurrent neural networks. 2016.

Toderici, G., Vincent, D., Johnston, N., Hwang, S. J., Minnen, D., Shor, J., and Covell, M. Full resolution image compression with recurrent neural networks. volume 2017-January, 2017. doi: 10.1109/CVPR.2017.577.

Truong, C., Oudre, L., and Vayatis, N. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.

Wallace, G. K. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38, 1992. ISSN 00983063. doi: 10.1109/30.125072.

Wiegand, T., Sullivan, G. J., Bjøntegaard, G., and Luthra, A. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13, 2003. ISSN 10518215. doi: 10.1109/TCSVT.2003.815165.

Zhao, B., Li, X., and Lu, X. Tth-rnn: Tensor-train hierarchical recurrent neural network for video summarization. *IEEE Transactions on Industrial Electronics*, 68:3629–3637, 2021.

Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., and Zhu, J.-H. Discrimination-aware channel pruning for deep neural networks. In *NeurIPS*, 2018.