Benchmark of Self-Supervised Models for Video Understanding on the CATER Dataset

Final Project Report

Théo Moutakanni
theo.moutakanni@supelec.fr

Etienne Boisseau
etienne.boisseau@ensae.fr

Abstract

Deep Learning has already proved to be very useful on static images. The capabilities of CNN to extract information in a supervised or unsupervised way from pictures (even when training on only one picture) have been extensively analysed throughout a large number of experiments. Its generalization to video may seems straight-forward, but when we are dealing with long-term video understanding, the capabilities of classical CNN are bottle-necked by the temporal aspect of the data. Supervised and unsupervised training have already been successfully applied to short-term video datasets. But long-term understanding of a scene is still an open issue, with the CATER dataset showing the flaws of current spatio-temporal models. There is currently no benchmark on how self-supervised learning leverages long-term dependencies when learning without scene and objects bias. In this project, we benchmark methods for self-supervised learning applied to action and scene recognition on the CATER dataset. The code is available here.[1]

1. Introduction

Deep Neural Networks are formidable feature extractors. But while they may extract very useful abstract features on a static image, a video is more than the sum of its parts. Thus, simply concatenating or pooling features [12] extracted a small number of frames (3D blocks) is not a good way to perform long-term video understanding. A more holistic approach is then needed to efficiently extract the temporal information that is contained in a video, like a LSTM layer.

Self-Supervised Learning (SSL) is a way to capture meaningful features using a pretext task to train a neural network. It allows a model to be trained on unlabeled data and force it to develop a semantic representation which can later be used for a variety of downstream tasks.

[1] https://github.com/TheoMoutakanni/CATER-SSL

The CATER Dataset [4], for Compositional Actions and TEmporal Reasoning in dynamic tabletop scenes, is a dataset made to benchmark the ability of video understanding models to leverage the time component of the actions without any implicit bias from the scene. Scene bias is a common phenomenon in action recognition tasks, in which the static setting of the scene can be used as a shortcut for video-understanding models to do action recognition or VideoQA **without understanding the actions performed in the video** [10][16]. For instance, a model may easily guess that a character is jumping based on the fact that the scene contains a trampoline [2].

1.1. The CATER Dataset

The CATER Dataset is a synthetic dataset featuring 3D-rendered videos. In these videos, objects such as cubes and cones are placed on a 6x6 grid and undergo a series of "actions", which may include rotating them, sliding them, or containing one item inside another, where it is hidden. The particularity of this dataset is that actions are spatially and temporally connected in the long term. By hiding an object inside another one and moving both of them, we are making a spatial action that needs temporal context to be understood.

There are 3 downstream tasks defined on the CATER Dataset :

1. Atomic action recognition (Detect whether actions X, Y, Z have occured in the video)
2. Compositional action recognition (Recognize a combination of two actions such as "X before Y", "Z during Y")
3. Item localization (Determining the location of an object at the end of the video, even though the object may be contained inside another during the video)

The first benchmark on the CATER dataset shows that using LSTM networks (instead of averaging) to aggregate representations of small subsets of the video increases the

accuracy on downstream tasks (which is not always the case on other datasets [1]). Flow (short-term temporal features) also contains very little information for tasks 2) and 3). This dataset is thus a good proxy to analyse spatio-temporal models for long-term temporal understanding.

2. Problem Definition

The CATER paper provides a benchmark of multiple supervised models, but no self-supervised models.

We implement and benchmark self-supervised models to compare their embeddings and long-term understanding of the scenes, without any scene bias.

Self-supervised models on videos may be sorted in several categories :

1. Learning by order contrastive tasks:[13][19] [3][20]
2. Learning by predicting the future: [18][11][5]
3. Learning by tracking objects through time: [17][9]
4. Learning by data augmentation contrastive tasks: [7][14][16]

These methods all show good performance on traditional video datasets which may exhibit scene bias. The question we might ask ourselves, and which this paper tries to answer, is how these methods compare when used on a dataset which by construction does not have any scene bias: the CATER dataset.

The last group will not be tested, as useful data-augmentations for the downstream tasks are not trivial to find, and may not generalize well outside of the CATER dataset, but we will test one paper from each of the first three groups.

3. Related Work

This project falls in line with two non-intersecting bodies of work:

- A benchmark of fully supervised methods on the CATER dataset, shown in the original CATER paper [4].
- Results of self-supervised methods on other datasets, shown in the methods' respective papers.

Our work fills the gap by providing a benchmark of self-supervised methods on the CATER dataset.

4. Methodology

In order to get meaningful comparisons, when a 2D encoder network is mentioned, we will use Resnet-50. When it is a 3D encoder, we will be using a R3D-18 backbone[2] [6][15], ie. I3D [1] with Resnet3D modules.

We will then train at least 1 model for each of the first three groups, using the official code when available. The selected methods are :

1. Self-supervised spatiotemporal learning via video clip order prediction [19]
2. Memory-augmented dense predictive coding for video representation learning [5]
3. Learning correspondence from the cycle-consistency of time [17]

After training on the CATER dataset, they will be compared by freezing all their weights and by training a linear classifier or a MLP on top of the model encoder for the 3 downstream tasks. We will use average pooling (linear classifier) or a 2 layer biLSTM (MLP) to pool temporal features when needed.

By doing so, we will be able to understand if the models are able to extract long-term spatiotemporal dependencies without any supervision.

We will now present the three methods we chose to benchmark.

4.1. Selected Methods

4.1.1 Self-supervised spatiotemporal learning via video clip order prediction [19]

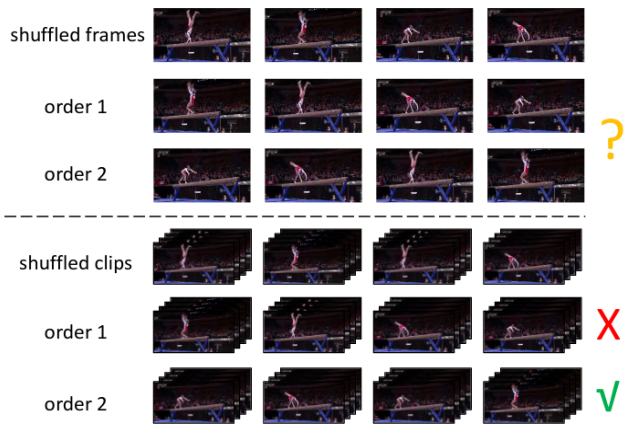

Figure 1. The model learns to sort video clips instead of single frames.

In this paper, Xu et al. propose a new method for self-supervised learning on video data. This is similar to the

[2] https://pytorch.org/docs/stable/torchvision/models.html

work of Lee et al. [8] on self-supervised learning by sorting frame sequences. The idea is that a neural network can learn to extract meaningful spatio-temporal features if its training objective is to predict in what order the frames of a video have occured.

However, as the above picture taken from the paper shows, it can be difficult for even a very good model to figure out the order of a sequence of frames, since frames contain only static scene information and no dynamic information about the movement of objects. Another weakness of the single-frame approach is that the network only learns to extract features on static images, which must later be pooled when analyzing a video, whereas we would like the model to learn to extract features on sequences of images (videos).

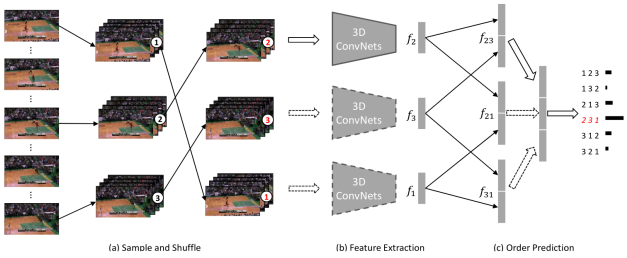

Figure 2. Model architecture

Xu et al. address these two concerns in their paper, by introducing a new pretext task: instead of sorting frames, a neural network model learns to sort short video clips. The above picture shows exactly how this works: First, a number of non-overlapping clips (usually 3) are extracted at random from a video. Then, the clips are shuffled and passed to the network in a random order. The network then has to correctly sort the clips by their order of appearance in the video.

Their method obtains state-of-the-art results when the pre-trained model is fine-tuned on the downstream task of action recognition, improving upon the single-frame-based methods by a significant margin.

4.1.2 Memory-augmented dense predictive coding for video representation learning [5]

This method is based on the future prediction pretext task. The idea is that if a network learns to predict future frames based on past frames, then it will have learned useful spatio-temporal features along the way. A previous approach, Dense Predictive Coding (DPC), models this task in the following way:

1. The video is separated into blocks of 5 frames $x_1 \dots x_n$.
2. Each block is encoded into a feature map by a 3D CNN f.
3. An aggregation function g aggregates consecutive feature maps into a context vector $c_t = g(f(x_1) \dots f(x_t))$.
4. A predictor ϕ maps c_t to a prediction of the following feature map $f(x_{t+1})$.

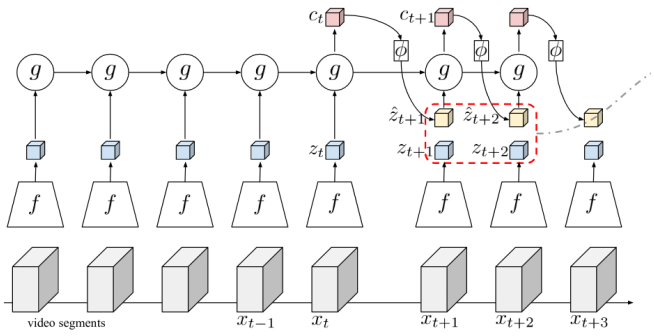

Figure 3. Original DPC architecture

Since the feature map extractor f is learned, a simple scheme like this would lead to trivial solutions (like f always outputting the same feature map regardless of input). Therefore, on top of $f(x_{t+1})$, which is the target to predict, additional distractor examples (such as random noise, frame blocks from other videos or frame blocks from the same video which are not x_{t+1}) are passed through f and the training objective for ϕ is *contrastive learning*, i.e. to make a prediction of $f(x_{t+1})$ which is close to the target *and* far away from the distractors.

This training objective encourages the learned feature maps f to be predictable from the previous frame blocks (which makes sure they don't encode noise), but informative enough to be distinguishable from other frame blocks; it also encourages the context vector c_t to contain all necessary information to predict the future, and it is this context vector which is later used in downstream tasks as e.g. the input to a linear model.

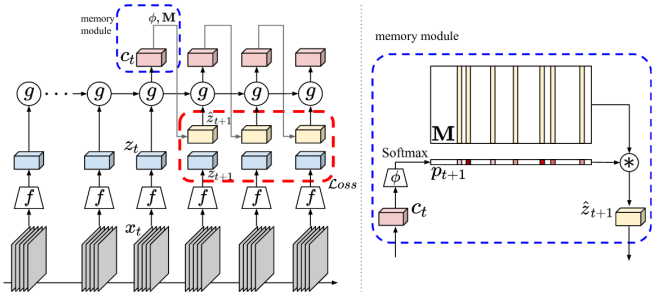

Figure 4. MemDPC architecture, including a memory module

Han et al. propose a variation on this architecture: MemDPC. It constitutes an improvement over the DPC algorithm by introducing a memory. Instead of predicting $f(x_{t+1})$ directly, now the predictor ϕ only tries to predict coefficients which are used to predict $f(x_{t+1})$ as a convex combination of memory entries.

The authors claims that this allows the model to "handle multiple hypotheses" for the future block. However it could also be said that the original DPC model likely also dealt with multiple hypotheses for the future, and their method merely makes this explicit rather than implicit.

In their tests, MemDPC reaches a marginal improvement over DPC in the action prediction downstream task.

4.1.3 Learning correspondence from the cycle-consistency of time [17]

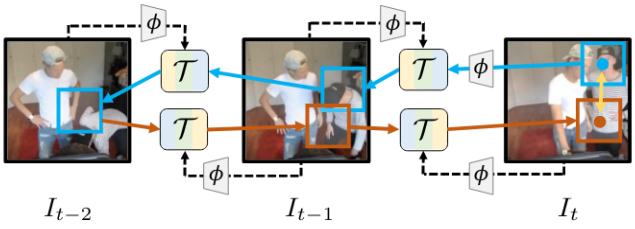

Figure 5. Cycle-consistency model architecture

In this paper, Wang et al. propose a general framework for learning image representations for visual correspondence from unlabeled videos. The main principle of this method is *learning to track*: The model learns useful feature representations by having to track objects backwards and forwards in time.

The task is modelled as follows (see figure above):

1. A video is separated into k frames up to time t: $I_{t-k:t}$.
2. A learned model ϕ turns each frame into a feature map: $x_{t-k:t}$.
3. A patch p_t is selected in image $I_{t-k:t}$ and passed through ϕ to give a feature map x_t^p of the patch.
4. A differentiable tracker function $\mathcal{T}$ compares the feature map of the patch, x_t^p, with the feature map of the previous image, x_{t-1}, and finds the most similar patch in it.
5. In the same way, the found patch is again localized in the previous image feature map x_{t-2}, until x_{t-k}. The process continues in the other direction, going forwards in time until the patch is localized in the feature map x_t.
6. The position of the patch that was tracked backwards and forwards in time is compared to the original patch position and their distance constitutes a loss which is backpropagated through all the operations the patch went through.

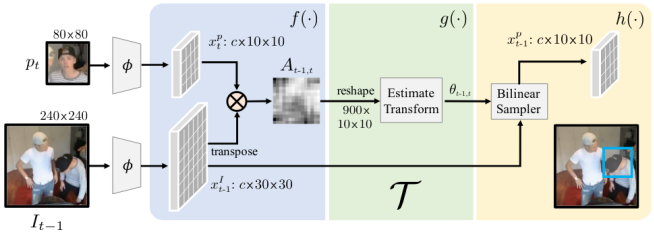

Figure 6. Tracker function architecture

The tracker function $\mathcal{T}$, illustrated above, is designed to be fairly weak in order to put the burden of representation on the model ϕ.

This method boasts impressive results, reaching good results on many supervised tasks *without fine-tuning*. Simply using a nearest-neighbors method on the feature maps of each frame given ϕ, they reach good results on a wide range of tasks such as texture propagation, semantic and instance mask propagation, and optical flow computation, which shows that their learned representation captures a large amount of useful semantic information.

5. Evaluation

We ran tests as described in the previous section, and will detail here the results of these experiments.

5.1. Method 1: Self-supervised spatiotemporal learning via video clip order prediction

For this method, we failed to reproduce the results of the original paper. We used the same hyperparameters as the paper (input video clips cropped randomly to

112x112, Adam optimizer, learning rate 0.001, momentum 0.9, weight decay 0.0005, clip length of 16 frames, 8 frames between each clip, 3 clips), with the only difference being the batch size (which we reduced from 8 to 4 due to hardware constraints) and the number of epochs (The authors used 300 epochs but we only completed 10, due to time and hardware constraints).

The model's inability to learn was made clear by its validation accuracy, which stayed close to random performance (16.7%) during the whole training process.

This was tested both on a dataset used by the authors themselves (UCF101) and on the CATER dataset.

It is likely that the number of epochs was the main factor in not being able to reproduce the results of the paper, since the authors outlined that improvement was slow during training. Other hyperparameters were experimented with but showed no better performance.

5.2. Method 2: Memory-augmented dense predictive coding for video representation learning

The same hyperparameters were used as in the paper, with an input of 8 clips of 5 frames of dimension 128x128, taken with a downsampling coefficient of 3. The size of the memory codeblock is 1024. The model is trained on 10 epochs with a batch size of 16, a learning rate of 0.001 and a weight decay of 1e-5 with the Adam optimizer.

We use the given code to pretrain a model. At the end of the 10 training epochs, the model has an accuracy of 34% on the pretext contrastive task. Thus, the model can determine which clips' features are the one coming next one out of three times. This seems pretty bad, given the fact that the future is predictable from short clips to short clips, but when new actions are performed, they are randomly choosen and increase the number of possible futures.

After pretraining a model using self-supervised learning, we remove the last head and put a new one to predict labels on the 2 downstream multilabel classification tasks.

We use the provided code and add a BiLSTM at the end to aggregate the outputs of each feature extractor over the clips.

After training the last layer of the model for 10 epochs on the downstream tasks 1 and 2, we get a mAP of 67% for the 1st task and 38% for the second one.

The results are better than a random network, but not comparable to a supervised network.

5.3. Method 3: Learning correspondence from the cycle-consistency of time

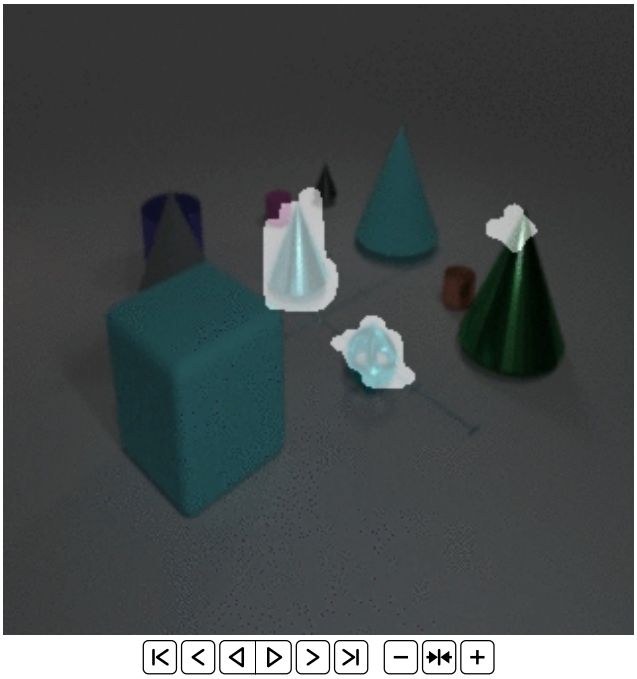

Figure 7. Tracking of the metallic blue cone. A pdf viewer with Adobe is needed. Alternatively: link to gif.

We use the same hyperparameters as in the paper, with just a ResNet18 instead of the 50 layer one. The image size is 256, no weight decay, a learning rate of 2e-4, a temperature of $\sqrt{512}$ and a grid size of 9. The training is done during 30 epochs with the Adam optimizer. The final loss is pretty low, and the network train well on the training dataset.

We use the provided code and add a BiLSTM at the end to aggregate the outputs of each feature extractor over the clips.

After training the last layer of the model for 10 epochs on the downstream tasks 1 and 2, we get a mAP of 67% for the 1st task and 40% for the second one.

We also extracted the tracking of multiple videos using a single label mask on the first frame as input. As we can see on Fig. 7, the tracking of occluded objects doesn't work, and is fooled when the object is hidden and moved, even when the full video is available. Unfortunately, even with easy shape tracking and without camera motion, this self-supervised network can't achieve great results. This methods seems to not capture long temporal relationships.

5.4. Comparison

Model	Task 1	Task 2
Random	56.2	19.5
R3D+NL+LSTM (supervised)	98.9	53.1
Clip Ordering	-	17.1
MemDPC	67	37
TimeCycle	67	40

Table 1. Benchmark between the 3 supervised methods and the supervised SoTA from the CATER paper.

We can see on the Table 1 that the self-supervised methods get better results than a random network, but are far away from the supervised methods.

6. Discussion

Self-supervised learning is a great way to leverage unlabeled datasets which are order of magnitude bigger than labeled ones. But it seems that all these methods lack a real understanding of the data they are working with, and are able to shortcut useful information for bias in the scene.

This benchmark allows us to better understand how the pretext tasks allow the models to learn.

MemDPC and TimeCycle reach the same accuracy, while Clip Ordering fails to train because of time and hardware issues. MemDPC and TimeCycle may attends to the same spatio-temporal relationships, or extract the same useful features, plus the ones they need for their pretext tasks.

This means that two very different training scheme can leads to features which are correlated.

To push further this benchmark, we could have worked on the task 3 of the CATER dataset using camera motion, but the time and hardware restrictions reduced our ambitions. A better analysis on the impact of the LSTM layers is also something to do to complete this benchmark, as it is the main part of the CATER dataset.

Finally, an analysis of the correlation of the different networks' features could also increase the value of this report.

References

[1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 2

[2] Jinwoo Choi, Chen Gao, Joseph C. E. Messou, and Jia-Bin Huang. Why can't I dance in the mall? learning to mitigate scene bias in action recognition. *CoRR*, abs/1912.05534, 2019. 1

[3] Alaaeldin El-Nouby, Shuangfei Zhai, Graham W. Taylor, and Joshua M. Susskind. Skip-clip: Self-supervised spatiotemporal representation learning by future clip order ranking, 2019. 2

[4] Rohit Girdhar and Deva Ramanan. Cater: A diagnostic dataset for compositional actions and temporal reasoning. *Int. Conf. Learn. Represent.*, 2020. 1, 2

[5] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. 2020. 2, 3

[6] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? 06 2018. 2

[7] Longlong Jing, Xiaodong Yang, Jingen Liu, and Yingli Tian. Self-supervised spatiotemporal feature learning via video rotation prediction, 2018. 2

[8] H. Lee, J. Huang, M. Singh, and M. Yang. Unsupervised representation learning by sorting sequences. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 667–676, 2017. 3

[9] Xueting Li, Sifei Liu, Shalini De Mello, Xiaolong Wang, Jan Kautz, and Ming-Hsuan Yang. Joint-task self-supervised learning for temporal correspondence. In *NeurIPS*, 2019. 2

[10] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 1

[11] Zelun Luo, Boya Peng, De-An Huang, Alexandre Alahi, and Li Fei-Fei. Unsupervised learning of long-term motion dynamics for videos. *CVPR*, Jul 2017. 2

[12] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. *arXiv:1706.06905*, 2017. 1

[13] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and Learn: Unsupervised Learning using Temporal Order Verification. In *ECCV*, 2016. 2

[14] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning, 2020. 2

[15] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition, 2018. 2

[16] Jinpeng Wang, Yuting Gao, Ke Li, Yiqi Lin, Andy J. Ma, Hao Cheng, Pai Peng, Rongrong Ji, and Xing Sun. Removing the background by adding the background: Towards background robust self-supervised video representation learning, 2020. 1, 2

[17] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019. 2, 4

[18] D. Wei, J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. 2018. 2

[19] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. 2019. 2

[20] D. Zhukov, J.-B. Alayrac, I. Laptev, and J. Sivic. Learning actionness via long-range temporal order verification. 2020. 2