We introduce PyTorchVideo, an open-source deep-learning library that provides a rich set of modular, efficient, and reproducible components for a variety of video understanding tasks, including classification, detection, self-supervised learning, and low-level processing. The library covers a full stack of video understanding tools including multimodal data loading, transformations, and models that reproduce state-of-the-art performance. PyTorchVideo further supports hardware acceleration that enables real-time inference on mobile devices. The library is based on PyTorch and can be used by any training framework; for example, PyTorchLightning, PySlowFast, or Classy Vision. PyTorchVideo is available at https://pytorchvideo.org/

---

**ABSTRACT**

We introduce PyTorchVideo, an open-source deep-learning library that provides a rich set of modular, efficient, and reproducible components for a variety of video understanding tasks, including classification, detection, self-supervised learning, and low-level processing. The library covers a full stack of video understanding tools including multimodal data loading, transformations, and models that reproduce state-of-the-art performance. PyTorchVideo further supports hardware acceleration that enables real-time inference on mobile devices. The library is based on PyTorch and can be used by any training framework; for example, PyTorchLightning, PySlowFast, or Classy Vision. PyTorchVideo is available at https://pytorchvideo.org/

**CSC CONCEPTS**

- Computing methodologies → Activity recognition and understanding.

**KEYWORDS**

video representation learning, video understanding

**ACM Reference Format:**

Haoqi Fan*, Tullie Murrell†, Heng Wang‡, Kalyan Vasudev Alwala†, Yanghao Li‡, Yilei Li†, Bo Xiong‡, Nikhila Ravi, Meng Li, Haichuan Yang, Jitendra Malik, Ross Girshick, Matt Feiszli, Aaron Adcock‡, Wan-Yen Lo‡, Christoph Feichtenhofer‡

Facebook AI

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MM '21, October 20–24, 2021, Virtual Event, China.

© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-8651-7/21/10...

https://doi.org/10.1145/3474085.3478329

1 INTRODUCTION

Recording, storing, and viewing videos has become an ordinary part of our lives; in 2022, video traffic will amount to 82 percent of all Internet traffic [2]. With the increasing amount of video material readily available (e.g. on the web), it is now more important than ever to develop ML frameworks for video understanding.

With the rise of deep learning, significant progress has been made in video understanding research, with novel neural network architectures, better training recipes, advanced data augmentation, and model acceleration techniques. However, the sheer amount of video data often makes these tasks computationally demanding; therefore efficient solutions are non-trivial to implement.

To date, there exist several popular video understanding frameworks, which provide implementations of advanced state-of-the-art video models, including PySlowFast [6], MMAction [27], MMAction2 [3], and Gluon-CV [14]. However, unlike a modularized library that can be imported into different projects, all of these frameworks are designed around training workflows, which limit their adoption beyond applications tailored to one specific codebase.

More specifically, we see the following limitations in prior efforts. First, reproducibility – an important requirement for deep learning software – varies across frameworks; e.g. identical models are reproduced with varying accuracy on different frameworks [3, 6, 14, 27]. Second, regarding input modalities, the frameworks are mainly focused on visual-only data streams. Third, supported tasks only encompass human action classification and detection. Fourth, none of the existing codebases support on-device acceleration for real-time inference on mobile hardware.

We believe that a modular, component-focused video understanding library that addresses the aforementioned limitations will strongly support the video research community. Our intention is to develop a library that aims to provide fast and easily extensible components to benefit researchers and practitioners in academia and industry.
We present PyTorchVideo – an efficient, modular and reproducible deep learning library for video understanding which supports the following (see Fig. 1 for an overview):

- a modular design with extendable interface for video modeling using Python
- a full stack of video understanding machine learning components from established datasets to state-of-the-art models
- real-time video classification through hardware accelerated on-device support
- multiple tasks, including human action classification and detection, self-supervised learning, and low-level vision tasks
- reproducible models and datasets, benchmarked in a comprehensive model zoo
- multiple input modalities, including visual, audio, optical-flow and IMU data

PyTorchVideo is distributed with a Apache 2.0 License, and is available on GitHub at https://github.com/facebookresearch/pytorchvideo.

2 LIBRARY DESIGN

Our library follows four design principles, outlined next (§2.1-2.4).

2.1 Modularity

PyTorchVideo is built to be component centric: it provides independent components that are plug-and-play and ready to mix-and-match for any research or production use case. We achieve this by designing models, datasets and data transformations (transforms) independently, only enforcing consistency through general argument naming guidelines. For example, in the pytorchvideo.data module all datasets provide a data_path argument, or, for the pytorchvideo.models module, any reference to input dimensions uses the name dim_in. This form of duck-typing provides flexibility and straightforward extensibility for new use cases.

2.2 Compatibility

PyTorchVideo is designed to be compatible with other frameworks and domain specific libraries. In contrast to existing video frameworks [3, 6, 14, 27], PyTorchVideo does not rely on a configuration system. To maximize the compatibility with Python based frameworks that can have arbitrary config-systems, PyTorchVideo uses keyword arguments in Python as a "naive configuration" system.

PyTorchVideo is designed to be interoperable with other standard domain specific libraries by setting canonical modality based tensor types. For videos, we expect a tensor of shape $[..., C, T, H, W]$, where $T \times H \times W$ are spatiotemporal dimensions, and $C$ is the number of color channels, allowing any TorchVision model or transform to be used together with PyTorchVideo. For raw audio waveforms, we expect a tensor of shape $[..., T]$, where $T$ is the temporal dimension, and for spectrograms, we expect a tensor of shape $[..., T, F]$, where $T$ is time and $F$ is frequency, aligning with TorchAudio.

2.3 Customizability

One of PyTorchVideo’s primary use cases is supporting the latest research methods; we want researchers to easily contribute their work without requiring refactoring and architecture modifications. To achieve this, we designed the library to reduce overhead for adding new components or sub-modules. Notably in the pytorchvideo.models module, we use a dependency injection inspired API. We have a composable interface, which contains injectable skeleton classes and a factory function interface that builds reproducible implementations using composable classes. We anticipate this injectable class design to be useful for researchers that want to easily plug in new sub-components (e.g. a new type of convolution) into the structure of larger models such as a ResNet [16] or SlowFast [9]. The factory functions are more suitable for reproducible benchmarking of complete models, or usage in production. An example for a customized SlowFast network is in Algorithm 1.

2.4 Reproducibility

PyTorchVideo maintains reproducible implementations of all models and datasets. Each component is benchmarked against the reported performance in the respective, original publication. We report performance and release model files online1 as well as on PyTorch Hub2. We rely on test coverage and recurrent benchmark jobs to verify and monitor performance and to detect potential regressions introduced by codebase updates.

3 LIBRARY COMPONENTS

PyTorchVideo allows training of state-of-the-art models on multi-modal input data, and deployment of an accelerated real-time model on mobile devices. Example components are shown in Algorithm 2.

Algorithm 1 Code for a SlowFast network with customized norm and activation layer classes, and a custom head function.

```python
# Create a customized SlowFast Network.
customized_slowfast = create_slowfast(
    activation=CustomizedActivation,
    norm=CustomizedNorm,
    head=create_customized_head,
)
```

3.1 Data

Video contains rich information streams from various sources, and, in comparison to image understanding, video is more computationally demanding. PyTorchVideo provides a modular, and efficient data loader to decode visual, motion (optical-flow), acoustic, and Inertial Measurement Unit (IMU) information from raw video.

PyTorchVideo supports a growing list of data loaders for various popular video datasets for different tasks: video classification task for UCF-101 [22], HMDB-51 [19], Kinetics [18], Charades [20], and Something-Something [11], egocentric tasks for Epic Kitchen [5] and DomSev [21], as well as video detection in AVA [13].

All data loaders support several file formats and are data storage agnostic. For encoded video datasets (e.g. videos stored in mp4 files), we provide PyAV, TorchVision, and Decord decoders. For long videos – when decoding is an overhead – PyTorchVideo provides support for pre-decoded video datasets in the form image files.

---

1Numbers and model weights can be found in https://github.com/facebookresearch/pytorchvideo/blob/master/docs/source/model_zoo.md
2https://pytorch.org/hub/
PyTorchVideo models can be used in combination with different downstream tasks: supervised classification and detection of human actions in video [9], as well as self-supervised (i.e. unsupervised) video representation learning with Momentum Contrast [15], SimCLR [1], and Bootstrap your own latent [12].

3.4 Accelerator

PyTorchVideo provides a complete environment (Accelerator) for hardware-aware design and deployment of models for fast inference.
We thank the PyTorchVideo contributors: Aaron Adcock, Amy Bearman, Bernard Nguyen, Bo Xiong, Chengyuan Yan, Christoph Feichtenhofer, Dave Schnizlein, Haoqi Fan, Heng Wang, Jackson Hamburger, Jitendra Malik, Kalyan Vasudev Alwala, Matt Feiszli, Nikhilka Ravi, Ross Girshick, Tullie Murrell, Wan-Yen Lo, Weiyao Hamburger, Jitendra Malik, Kalyan Vasudev Alwala, Matt Feiszli, Bernard Nguyen, Bo Xiong, Chengyuan Yan, Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Christoph Feichtenhofer. 2018. Scaling Egocentric Vision: The EPIC-KITCHENS Dataset. In ECCV.

REFERENCES

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In ICLR.
[2] Cisco. 2020. Annual Internet Report (2018–2023) White Paper.
[3] MMAction2 Contributors. 2020. OpenMMLab’s Next Generation Video Understanding Toolbox and Benchmark. https://github.com/open-mmlab/mmaction2.
[4] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In Proc. CVPR.
[5] Dima Damen, Hazel D profound, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. 2018. Scaling Egocentric Vision: The EPIC-KITCHENS Dataset. In ECCV.
[6] Haoqi Fan, Yanghao Li, Bo Xiong, Wan-Yen Lo, and Christoph Feichtenhofer. 2020. PySlowFast. https://github.com/facebookresearch/slowfast.
[7] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. 2021. Multiscale Vision Transformers.
[8] Christoph Feichtenhofer. 2020. X3D: Expanding architectures for efficient video recognition. In CVPR.
[9] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. 2019. SlowFast networks for video recognition. In ICCV.
[10] Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross Girshick, and Kaiming He. 2021. A Large-Scale Study on Unsupervised Spatiotemporal Representation Learning. In CVPR.
[11] RagHAV Goyal, Samira Elbrahiimah Khou, Vincent Michalski, Joanna Materzymska, Susanne Westphahl, Heuna Kim, Valentin Haerl, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. 2017. The ‘Something Something’ Video Database for Learning and Evaluating Visual Common Sense. In ICCV.
[12] Jean-Bastien Grill, Florian Florub, Florent Alchte, Corentin Tallec, Pierre H. Richemond, Elena Bucharitskaya, Carl Doersch, Bernardo Alevia Pires, Zhao-han Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koryak Kavfulcuoglu, Rémi Munos, and Michal Valko. 2020. Boostup your own latent: A new approach to self-supervised Learning. In NIPS.
[13] Chunhui Gu, Chen Sun, David A. Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susamma Rico, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. 2018. AVA: A Video Dataset of Spatio-Temporally Localized Atomic Visual Actions. In CVPR.
[14] Jian Guo, He He, Tong He, Leonard Lausen, Mu Li, Hubin Lin, Xingqian Shi, Chengwang Wang, Junyuan Xie, Sheng Zha, Anton Zhong, Zhe Zhan, Zhongyue Zhang, Shuai Zhang, and Yi Zhu. 2020. GhsotCV and GhosNLP: Deep Learning in Computer Vision and Natural Language Processing. In Journal of Machine Learning Research.
[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Mo- mentum Contrast for Unsupervised Visual Representation Learning. In CVPR.
[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In CVPR.
[17] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Baralji Lakshminarayanan. 2020. AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty. ICLR (2020).
[18] Will Kill, Joao Carreira, Karen Simonyan, Brawn Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. 2017. The kinetics human action video dataset. arXiv.
[19] Hildegaard Kuehne, Euhrean Jhjang, Establisar Garrote, Tomaso Poggio, and Thomas Serre. 2011. HMDB: a large video database for human motion recognition. In IJCV.
[20] Gunnar A Sigurdsson, Gis Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. 2016. Hollywood in homes: Crowdsourcing data for activity understanding. In CVPR.
[21] Michel Silva, Washington Ramos, João Ferreira, Felipe Chamone, Mario Campos, and Eriksson R. Nascimento. 2018. A Weighted Sparse Sampling and Smoothing Frame Transition Approach for Semantic Fast-Foward First-Person Videos. In CVPR.
[22] Khurram Soomro, Amir Rosdan Zamir, and Mubarak Shah. 2012. UCF101: A Dataset of 101 Human Actions Collected from Videos in the Wild. Technical Report CRCV-TR-12-01.
[23] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. 2019. Video Classification with Channel-Separated Convolutional Networks. In Proc. ICCV.
[24] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, , and Manohar Paluri. 2018. A closer look at spatiotemporal convolutions for action recognition. In CVPR.
[25] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-local Neural Networks. In CVPR.
[26] Fangyi Xiao, Yong Jae Lee, Kristen Grauman, Jitendra Malik, and Christoph Feichtenhofer. 2020. Audiovisual slowfast networks for video recognition. arXiv preprint arXiv:2001.08740 (2020).
[27] Davina Lin Yue Zhao, Yuanjun Xiong. 2019. MMAction. https://github.com/open-mmlab/mmaction.