

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.model` 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

```

1 # Create a customized SlowFast Network.
2 customized_slowfast = create_slowfast(
3     activation=CustomizedActivation,
4     norm=CustomizedNorm,
5     head=create_customized_head,
6 )

```

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

reproducible implementations using composable classes. We anticipate this injectable class design to being 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 Table 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 Table 2.

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.

3.2 Transforms

Transforms - as the key components to improve the model generalization - are designed to be flexible and easy to use in PyTorchVideo. PyTorchVideo provides factory transforms that include common

¹Numbers and model weights can be found in https://github.com/facebookresearch/pytorchvideo/blob/master/docs/source/model_zoo.md

²<https://pytorch.org/hub/>

```

1  from pytorchvideo import data, models, accelerator
2  # Create visual and acoustic models.
3  visual_model = models.slowfast.create_slowfast(
4      model_num_class=400,
5  )
6  acoustic_model = models.resnet.create_acoustic_resnet(
7      model_num_class=400,
8  )
9  # Create Kinetics dataloader.
10 kinetics_loader = torch.utils.data.DataLoader(
11     data.Kinetics(
12         data_path=DATA_PATH,
13         clip_sampler=data.make_clip_sampler(
14             "uniform",
15             CLIP_DURATION,
16         ),
17     ),
18     batch_size=BATCH_SIZE,
19 )
20 # Deploy model.
21 visual_net_inst_deploy = accelerator.deployment.\
22     convert_to_deployable_form(net_inst, input_tensor)

```

Table 2: Code snippet to train, run inference, and deploy a state-of-the-art video model with PyTorchVideo.

recipes for training state-of-the-art video models [7–9]. Recent data augmentations are also provided by the library (e.g. MixUp, CutMix, RandAugment [4], and AugMix [17]). Finally, users have the option to create custom transforms by composing individual ones.

3.3 Models

PyTorchVideo contains highly reproducible implementations of popular models and backbones for video classification, acoustic event detection, human action localization (detection) in video, as well as self-supervised learning algorithms.

The current set of models includes standard single stream video backbones such as C2D [25], I3D [25], Slow-only [9] for RGB frames and acoustic ResNet [26] for audio signal, as well as efficient video networks such as SlowFast [9], CSN [23], R2+1D [24], and X3D [8] that provide state-of-the-art performance. PyTorchVideo also provides multipathway architectures such as Audiovisual SlowFast networks [26] which enable state-of-the-art performance by disentangling spatial, temporal, and acoustic signals across different pathways. It further supports methods for low-level vision tasks for researchers to build on the latest trends in video representation learning.

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

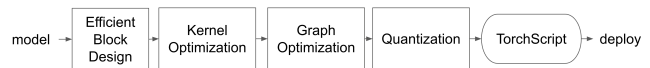


Figure 2: Acceleration and deployment pipeline.

on device, including efficient blocks and kernel optimization. The deployment flow is illustrated in Figure 2.

Specifically, we perform kernel-level latency optimization for common kernels in video understanding models (e.g. conv3d). This optimization brings two-fold benefits: (1) latency of these kernels for floating-point inference is significantly reduced; (2) quantized operation (int8) of these kernels is enabled, which is not supported for mobile devices by vanilla PyTorch. Our Accelerator provides a set of efficient blocks that build upon these optimized kernels, with their low latency validated by on-device profiling. In addition, our Accelerator provides kernel optimization in deployment flow, which can automatically replace modules in the original model with efficient blocks that perform equivalent operations. Overall, the PyTorchVideo Accelerator provides a complete environment for hardware-aware model design and deployment for *fast* inference.

4 BENCHMARKS

PyTorchVideo supports popular video understanding tasks, such as video classification [9, 11, 18], action detection [13], video self-supervised learning [10] and efficient video understanding on mobile hardware. We provide comprehensive benchmarks for different tasks and a large set of model weights for state-of-the-art methods. This section provides a snapshot of the benchmarks. A comprehensive listing can be found in the model zoo³. The models in PyTorchVideos’ model zoo reproduce performance of the original publications and can seed future research that builds upon them, bypassing the need for re-implementation and costly re-training.

4.1 Video classification

model	Top-1	Top-5	FLOPs (G)	Param (M)
I3D R50, 8×8 [25]	73.3	90.7	37.5 × 3 × 10	28.0
X3D-L, 16×5 [8]	77.4	93.3	26.6 × 3 × 10	6.2
SlowFast R101, 16×8 [9]	78.7	93.6	215.6 × 3 × 10	53.8
MViT-B, 16×4 [7]	78.8	93.9	70.5 × 3 × 10	36.6

Table 3: Video classification results on Kinetics-400 [18].

PyTorchVideo implements classification for various datasets, including UCF-101 [22], HMDB-51 [19], Kinetics [18], Charades [20], Something-Something [11] and Epic-Kitchens [5]. Table 3 shows a benchmark snapshot on Kinetics-400 for three popular state-of-the-art methods, which are measured in Top-1 and Top-5 accuracies. Further classification models with pre-trained weights, that can be used for a variety of downstream tasks, are available online³.

4.2 Video action detection

The video action detection task aims to perform spatiotemporal localization of human actions in videos. Table 4 shows detection performance in mean Average Precision (mAP) on the AVA dataset [13] using Slow and SlowFast networks [9].

³A full set of supported models on different datasets can be found from https://pytorchvideo.readthedocs.io/en/latest/model_zoo.html

model	mAP	Param (M)
Slow R50, 4×16 [9]	19.50	31.78
SlowFast R50, 8×8 [9]	24.67	33.82

Table 4: Video action detection results on AVA dataset [13].

SSL method	Kinetics	UCF101	AVA	Charades	SSv2
SimCLR [1]	62.0	87.9	17.6	11.4	52.0
BYOL [12]	68.3	93.8	23.4	21.0	55.8
MoCo [15]	67.3	92.8	20.3	33.5	54.4

Table 5: Performance of 4 video self-supervised learning (SSL) methods on 5 downstream datasets [10].

model	Vanilla PyTorch	PyTorchVideo	Speed Up
	Latency (ms)	Latency (ms)	
X3D-XS (fp32) [8]	1067	233	4.6×
X3D-S (fp32) [8]	4249	764	5.6×
X3D-XS (int8) [8]	(not supported)	165	-

Table 6: Speedup of PyTorchVideo models on mobile CPU.

4.3 Video self-supervised learning

We provide reference implementations of popular self-supervised learning methods for video [10], which can be used to perform unsupervised spatiotemporal representation learning on large-scale video data. Table 5 summarizes the results on 5 downstream datasets.

4.4 Efficient mobile video understanding

The Accelerator environment for efficient video understanding on mobile devices is benchmarked in Table 6. We show several efficient X3D models [8] on a Samsung Galaxy S9 mobile phone. With the efficient optimization strategies in PyTorchVideo, X3D achieves 4.6 - 5.6× inference speed up, compared to the default PyTorch implementation. With quantization (int8), it can be further accelerated by 1.4×. The resulting PyTorchVideo-accelerated X3D model runs around 6× *faster than real time*, requiring roughly 165ms to process one second of video, *directly on the mobile phone*. The source code for an on-device demo in iOS⁴ and Android⁵ is available.

5 CONCLUSION

We introduce PyTorchVideo, an efficient, flexible, and modular deep learning library for video understanding that scales to a variety of research and production applications. Our library welcomes contributions from the research and open-source community, and will be continuously updated to support further innovation. In future work, we plan to continue enhancing the library with optimized components and reproducible state-of-the-art models.

ACKNOWLEDGMENTS

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, Nikhila Ravi, Ross Girshick, Tullie Murrell, Wan-Yen Lo, Weiyao Wang, Yanghao Li, Yilei Li, Zhengxing Chen, Zhicheng Yan.

⁴<https://github.com/pytorch/ios-demo-app/tree/master/TorchVideo>

⁵<https://github.com/pytorch/android-demo-app/tree/master/TorchVideo>

REFERENCES

- [1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*.
- [2] Cisco. 2020. Annual Internet Report (2018–2023) White Paper.
- [3] MMAAction2 Contributors. 2020. OpenMMLab’s Next Generation Video Understanding Toolbox and Benchmark. <https://github.com/open-mmlab/mmaaction2>.
- [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 Doughty, 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 Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Freund, 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 Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap 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, Susanna Ricco, 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, Haibin Lin, Xingjian Shi, Chenguang Wang, Junyuan Xie, Sheng Zha, Aston Zhang, Hang Zhang, Zhi Zhang, Zhongyue Zhang, Shuai Zheng, and Yi Zhu. 2020. GluonCV and GluonNLP: 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. 2021. Momentum 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 Balaji Lakshminarayanan. 2020. AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty. *ICLR (2020)*.
- [18] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. 2017. The kinetics human action video dataset. *arXiv*.
- [19] Hildegard Kuehne, Hueihan Jhuang, Estibaliz Garrote, Tomaso Poggio, and Thomas Serre. 2011. HMDB: a large video database for human motion recognition. In *ICCV*.
- [20] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. 2016. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*.
- [21] Michel Silva, Washington Ramos, João Ferreira, Felipe Chamone, Mario Campos, and Erickson R. Nascimento. 2018. A Weighted Sparse Sampling and Smoothing Frame Transition Approach for Semantic Fast-Forward First-Person Videos. In *CVPR*.
- [22] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. *UCF101: A Dataset of 101 Human Actions Calsses 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] Fanyi 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] Dahua Lin Yue Zhao, Yuanjun Xiong. 2019. MMAAction. <https://github.com/open-mmlab/mmaaction>.