Supplementary Material: X3D: Expanding Architectures for Efficient Video Recognition

Christoph Feichtenhofer

Facebook AI Research (FAIR)

Appendix

This document provides supplementary material as referenced in the main paper: Sec. A contains additional implementation details for: AVA Action Detection (§A.1), Charades Action Classification (§A.2), and Kinetics Action Classification (§A.3). Sec. B contains further results and ablations on Kinetics-400.

A. Additional Implementation Details

A.1. Details: AVA Action Detection

Detection architecture. We exactly follow the detection architecture in [3] to allow direct comparison of X3D with SlowFast networks as a backbone. The detector is similar to Faster R-CNN [22] with minimal modifications adapted for video. Since our paper focuses on efficiency, by default, we do not increase the spatial resolution of res_5 by $2\times$ [3]. Region-of-interest (RoI) features [4] are extracted at the last feature map of res_5 by extending a 2D proposal at a frame into a 3D RoI by replicating it along the temporal axis, similar as done in previous work [7, 16, 24], followed by application of frame-wise RoIAlign [8] and temporal global average pooling. The RoI features are then max-pooled and fed to a per-class, sigmoid classifier for prediction.

Training. For direct comparison, the training procedure and hyper-parameters for AVA follow [3] without modification. The network weights are initialized from the Kinetics models and we use step-wise learning rate decay, that is reduced by $10\times$ when validation error saturates. We train for 14k iterations (68 epochs for ~211k data), with linear warm-up [6] for the first 1k iterations and use a weight decay of 10^{-7}, as in [3]. All other hyper-parameters are the same as in the Kinetics experiments. Ground-truth boxes, and proposals overlapping with ground-truth boxes by IoU > 0.9, are used as the samples for training. The inputs are instantiation-specific clips of size $\gamma_t \times 112 \gamma_s \times 112 \gamma_s$ with time stride γ_τ.

The region proposal extraction also follows [3] and is summarized here for completeness. We follow previous works that use pre-computed proposals [7, 16, 24]. Our region proposals are computed by an off-the-shelf person detector, *i.e.*, that is not jointly trained with the action detection models. We adopt a person-detection model trained with *Detectron* [5]. It is a Faster R-CNN with a ResNeXt-101-FPN [18, 29] backbone. It is pre-trained on ImageNet and the COCO human keypoint images [19]. We fine-tune this detector on AVA for person (actor) detection. The person detector produces 93.9 AP@50 on the AVA validation set. Then, the region proposals for action detection are detected person boxes with a confidence of > 0.8, which has a recall of 91.1% and a precision of 90.7% for the person class.

A.2. Details: Charades Action Classification

For **Charades**, we fine-tune the Kinetics models. All settings are the same as those of Kinetics, except the following. A per-class sigmoid output is used to account for the mutli-class nature. We train on a single machine for 24k iterations using a batch size of 16 and a base learning rate of 0.02 with $10\times$ step-wise decay if the validation error saturates. We use weight decay of 10^{-5}. We also increase the model temporal stride by $\times 2$ as this dataset benefits from longer clips. For inference, we temporally max-pool prediction scores [3].

A.3. Details: Kinetics Action Classification

Training details. We use the initialization in [9]. We adopt synchronized SGD training on 128 GPUs following the recipe in [6]. The mini-batch size is 8 clips per GPU (so the total mini-batch size is 1024). We train with Batch Normalization (BN) [15], and the BN statistics are computed within each 8 clips, unless noted otherwise. We adopt a half-period cosine schedule [20] of learning rate decaying: the learning rate at the n-th iteration is $\eta \cdot 0.5[\cos(\frac{n}{n_{\text{max}}}\pi)+1]$, where n_{max} is the maximum training iterations and the base learning rate η is set as 1.6. We also use a linear warm-up strategy [6] in the first 8k iterations. Unless specified, we train for 256 epochs (60k iterations with a total mini-batch size of 1024, in ~240k Kinetics videos).We use momentum of 0.9, weight decay of 5×10^{-5} and dropout [10] of 0.5 is used before the final classifier.

For **Kinetics-600**, we extend the training epochs (and schedule) of above by $2\times$. All other hyper-parameters are exactly as for Kinetics-400.

Implementation details. Non-Local (NL) blocks [28] are not used for X3D. For SlowFast results, we use exactly the same implementation details as in [3]. Specifically, for SlowFast models involving NL, we initialize them with the counterparts that are trained without NL, to facilitate convergence. We only use NL on the (fused) Slow features of res_4 (instead of res_3+res_4 [28]). For X3D and EfficientNet3D, we follow previous work on 2D mobile architectures [11, 25, 26], using SE blocks [13] (also found beneficial for efficient video classification in [30]) and swish non-linearity [21]. To conserve memory, we use SE with original reduction ratio of $1/16$ only in every other residual block after the 3×3^2 conv; swish is only used before and after these layers and all other weight layers are followed by ReLU non-linearity [17]. We do not employ the "linear-bottleneck" design used in mobile image networks [11, 23, 25, 26], as we found it to sometimes cause instability in distributed training.

Expansion details. To expand the model specified in Table 1 of the main paper, we set all initial expansion factors, $\mathcal{X}_0$, to one *i.e.* $\gamma_t{=}\gamma_s{=}\gamma_w{=}\gamma_b{=}\gamma_d{=}1$; A temporal sampling rate γ_τ is not defined for the X2D base model as it does not have multiple frames. The smallest possible common expansion for this model is defined by increasing the number of frames from 1 to two; therefore we set the expansion-rate $\hat{c}$ to match the cost of increasing the temporal input length of the model by a factor of two (the smallest possible common increase in the first expansion step), which roughly doubles the cost of the model, $\hat{c} = 2$.

Then, in every step of our expansion we train a models, one for expanding each axis, such that its complexity doubles ($\hat{c} = 2$). For the individual axes this roughly[1] equals to the following operations:

- X-Fast: $\gamma_\tau \leftarrow 0.5\gamma_\tau$, reduces the sampling stride to double frame-rate while sampling the same input duration, this doubles the temporal size $\gamma_t \leftarrow 2\gamma_t$.
- X-Temporal: Increases frame-rate by $\gamma_\tau \leftarrow 0.75\gamma_\tau$ *and* input duration to double the input size $\gamma_t \leftarrow 2\gamma_t$ (*i.e.* $1.5\times$ higher frame-rate and $1.5\times$longer input duration).
- X-Spatial: Expands the spatial resolution proportionally $\gamma_s \leftarrow \sqrt{2}\gamma_s$.
- X-Depth: Expands the depth of the network by around $\gamma_d \leftarrow 2.2\gamma_d$.
- X-Width: Expands the global width for all layers by $\gamma_w \leftarrow 2\gamma_w$.
- X-Bottleneck: Expands the bottleneck width by roughly $\gamma_b \leftarrow 2.25\gamma_b$.

[1]The exact expansion factors slightly vary across steps to match the complexity increase, $\hat{c}$ (which is observable in Fig. 2 of the main paper).

The exact scaling factors slightly differ from one expansion step to the other due to rounding effects in network geometry (layers stride, activation size *etc.*).

Since the stepwise expansion also allows to elegantly integrate regularization (which is typically increased for larger models), we perform a regularization expansion if the training error of the previous expansion step starts to deviate from the validation error. Specifically, we start the expansion with double the batch-size and half learning schedule than described above, then the BN statistics are computed within each 16 clips which lowers regularization and improves performance on small models. The batch-size is then decreased by $2\times$ at the 8^{th} step of expansion which increases generalization. We perform another regularization expansion at the 11^{th} step by using drop-connect with rate 0.1 [14].

B. Additional Results

Fig. A.1 shows a series of extra plots on Kinetics-400, analyzed next (this extends Sec. 4 of the main paper):

Inference cost. Here we aim to provide further ablations for the effect of using *fewer* testing clips for efficient video-level inference. In Fig. A.1 we show the trade-off for the full inference of a video, when varying the number of temporal clips used. The vertical axis shows the top-1 accuracy on K400-val and the horizontal axis the overall inference cost in FLOPs for different models.

First, for comparison, the plot on top-left is the same as the one shown in Fig. 3 of the main paper. The plot on top-right shows this same plot with a logarithmic scale applied to the FLOPs axis. Using this scaling it is clearer to observe that smaller models (X3D-S and X3D-M) can provide up to $20\times$ reduction in terms of multiply-add operations used during inference.

For example, 3-clip X3D-S produces 71.4% top-1 at 5.9 GFLOPs, whereas 10-clip CSN-50 [27] produces 70.8 top-1 at 119 GFLOPs ($20.2\times$ higher cost), or 10-clip X3D-S 72.9% top-1 at 19.6 GFLOPs, and 10-clip CSN-101 [27] 71.8% top-1 at 159 GFLOPs ($8.1\times$ higher cost).

The lower two plots in Fig. A.1 show the identical results on the test set of Kinetics-400 (which has been publicly released with Kinetics-600 [2]). Note that the test set is more challenging which leads to overall lower accuracy for all approaches [1]. We observe consistent results on the test set, illustrating good generalization of the models.

Mobile components. Finally, we ablate the effect of mobile components employed in X3D and EfficientNet3D. Since the components can have different effects of models from the small and large computation regime, we ablate the effects on a small (X3D-S) and a large model (X3D-XL).

First, we ablate channel-wise separable convolution [12], a key component in mobile ConvNets. We ablate two versions: (i) A version that reduces the bottleneck ratio (γ_b)

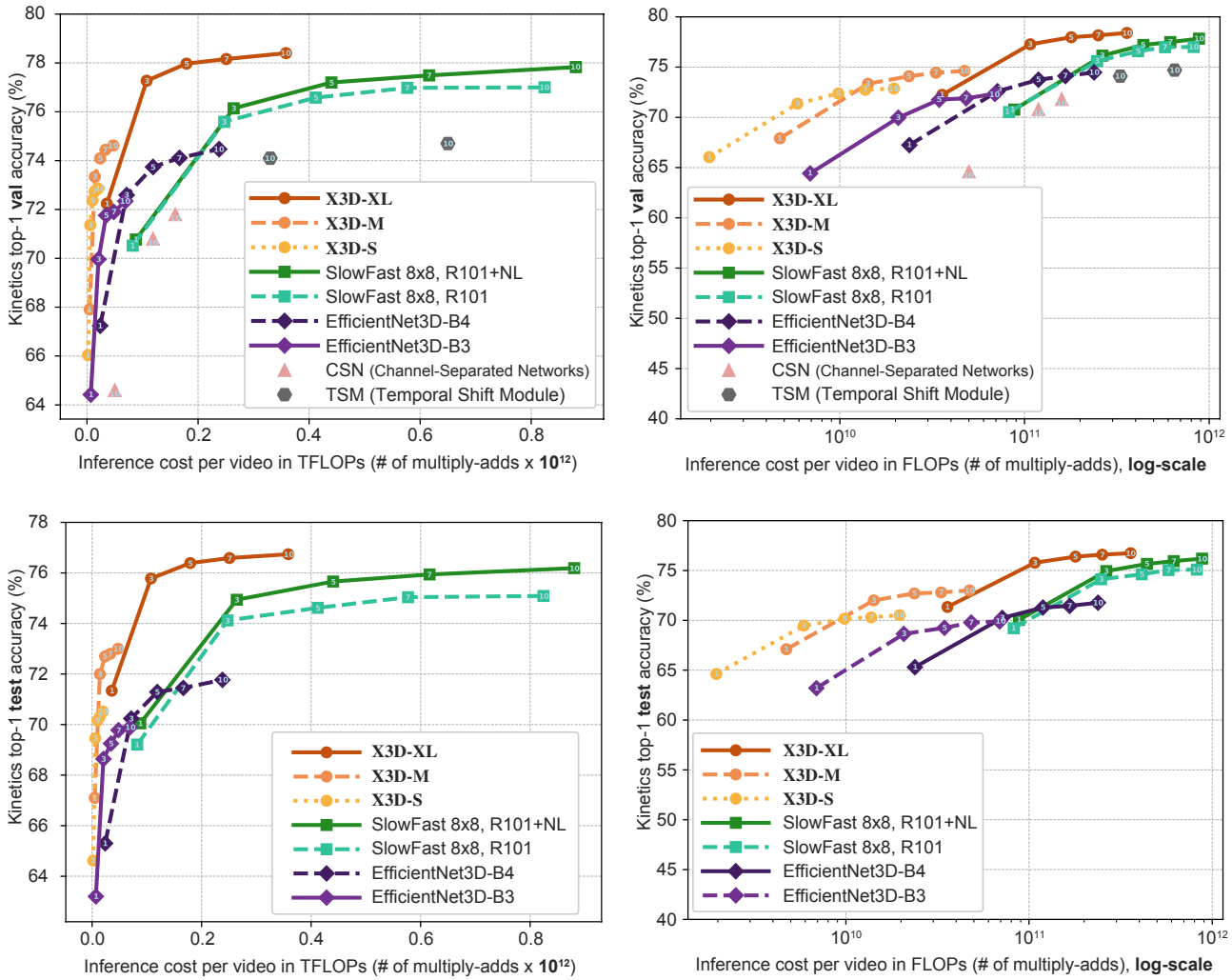

Figure A.1. **Accuracy/complexity trade-off** on K400-**val** (top) & **test** (bottom) for varying # of inference clips per video. The top-1 accuracy (vertical axis) is obtained by K-Center clip testing where the number of temporal clips $K \in \{1, 3, 5, 7, 10\}$ is shown in each curve. The horizontal axis shows the full inference cost per video. The left-sided plots shows a linear and the right-sided plot a logarithmic scale.

accordingly, such that the overall architecture preserves the multiple-add operations (FLOPs), and (ii) a version that keeps the originally, expanded bottleneck ratio.

Table A.1 shows the results. For case (i) we see that performance drops significantly by 4% top-1 accuracy for X3D-S and by 2.4% for X3D-XL. For case (ii), we see that the performance of the baselines increases by 0.3% and 0.8% top-1 accuracy for X3D-S and X3D-XL, respectively. This shows that separable convolution is important for small-computation budgets, however, for best-performance a non-separable convolution can provide gains (at high cost).

Second, we ablate swish non-linearities [21] (that are only implemented before and after the "bottleneck" convolution, to conserve memory). We observe that removing swish has a smaller performance decrease of 0.9% for X3D-S and 0.4% for X3D-XL, and therefore could be changed to ReLU (which can be implemented in-place) if memory is priority.

Third, we ablate SE blocks [13] (that are only used in every other residual block, to conserve memory). We observe that removing SE has a larger effect on performance, decreasing accuracy by 1.6% for X3D-S and 1.3% for for X3D-XL. These are similar effects on performance as have been shown Non-local (NL) attention blocks [28], and are also in line with [30], where SE attention blocks have been found beneficial for efficient video classification.

References

[1] ActivityNet-Challenge. http://activity-net.org/challenges/2017/evaluation.html, 2017. 2

[2] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about Kinetics-600. *arXiv:1808.01340*, 2018. 2

[3] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In *Proc. ICCV*, 2019. 1, 2

[4] Ross Girshick. Fast R-CNN. In *Proc. ICCV*, 2015. 1

model	top-1	top-5	FLOPs (G)	Param (M)
X3D-S	72.9	90.5	1.96	3.76
− CW conv γ_b= 0.6	68.9	88.8	1.95	3.16
− CW conv	73.2	90.4	17.6	22.1
− swish	72.0	90.4	1.96	3.76
− SE	71.3	89.9	1.96	3.60

(a) Ablating mobile components on a Small model.

model	top-1	top-5	FLOPs (G)	Param (M)
X3D-XL	78.4	93.6	35.84	11.0
− CW conv, γ_b= 0.56	76.0	92.6	34.80	9.73
− CW conv	79.2	93.5	365.4	95.1
− swish	78.0	93.4	35.84	11.0
− SE	77.1	93.0	35.84	10.4

(b) Ablating mobile components on an X-Large model.

Table A.1. Ablations of mobile components for video classification on K400-val. We show top-1 and top-5 classification accuracy (%), parameters, and computational complexity measured in GFLOPs (floating-point operations, in # of multiply-adds $\times 10^9$) for a single clip input of size $\gamma_t \times 112 \gamma_s \times 112 \gamma_x$. Inference-time computational cost is reported GFLOPs $\times 10$, as a fixed number of *10-Center* views is used. The results show that removing channel-wise separable convolution (CW conv) with unchanged bottleneck expansion ratio, γ_b, drastically increases mutliply-adds and parameters at slightly higher accuracy, while swish has a smaller effect on performance than SE.

[5] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018. 1

[6] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training ImageNet in 1 hour. *arXiv:1706.02677*, 2017. 1

[7] 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. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *Proc. CVPR*, 2018. 1

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proc. ICCV*, 2017. 1

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. CVPR*, 2015. 1

[10] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012. 1

[11] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. *arXiv:1905.02244*, 2019. 2

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. CVPR*, 2018. 2, 3

[14] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Proc. ECCV*, 2016. 2

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015. 1

[16] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proc. CVPR*, 2018. 1

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. CVPR*, 2017. 1

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proc. ECCV*, 2014. 1

[20] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv:1608.03983*, 2016. 1

[21] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 2, 3

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proc. CVPR*, 2018. 2

[24] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *ECCV*, 2018. 1

[25] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MnasNet: Platform-aware neural architecture search for mobile. In *Proc. CVPR*, 2019. 2

[26] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 2

[27] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proc. ICCV*, 2019. 2

[28] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proc. CVPR*, 2018. 2, 3

[29] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proc. CVPR*, 2017. 1

[30] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv:1712.04851*, 2017. 2, 3