# Massively Parallel Video Networks

Extended Abstract*

João Carreira†
DeepMind
joaoluis@google.com

Viorica Pătrăucean†
DeepMind
viorica@google.com

Andrew Zisserman
DeepMind
Department of Engineering Science, University of Oxford
zisserman@google.com

Simon Osindero
DeepMind
osindero@google.com

## ABSTRACT

We introduce a class of video understanding models that aims to improve efficiency for both dense and sparse tasks, e.g. per-frame human pose estimation and per-sequence action recognition. Leveraging operation pipelining and variable update rates, these models consolidate an internal state and perform a minimal amount of computation (e.g. as few as two convolutional layers) for each new frame to produce an output. The models are still very deep, with dozens of such operations being performed but in a pipelined fashion that enables depth-parallel computation. We demonstrate that the accuracy of the parallel models is comparable to that of sequential models, setting up the stage for more efficient video understanding.

## 1 EFFICIENT CAUSAL VIDEO PROCESSING

We are interested in improving the efficiency of deep video understanding networks for general temporal tasks such as navigating a scene, tracking objects and people, or recognising actions in videos, in the causal setting, i.e. frame-by-frame and without looking into the future. We study two general design principles for improving efficiency that can be applied to any video network: (i) pipelining with skip connections (see fig. 1), and (ii) exponentially diminishing clock rates along network depth, similar to 3D ConvNets [2, 15].

Let $\Gamma_\theta$ be a deep function approximator parametrised by $\theta$ having $D$ layers, which perform non-linear operations $\Gamma_{d\in\{1,D\}}$ on their inputs. For ConvNets, this operation is typically represented by $k > 0$ sequential convolutions each followed by a non-linearity such as ReLu. Given a video sequence with $n$ frames $f_{i\in\{1,n\}}$ and frame rate $f$, let $\mathcal{G}$ be the oriented graph obtained by unrolling the model over time (see Fig. 1). The nodes of the graph represent the non-linear operations and the oriented edges are the (tensor) activations transferred between them, establishing control dependencies. For simplicity and without loss of generality, we assume that every operation of the network can be computed in one cycle of duration $r$. We define the *latency* $l$ of the model as the interval between the moment when a new frame is available and the moment when the network output for that frame is available. Real-time execution is achieved iff $l \leq f$. We define *throughput* as the output rate, i.e. how often does the network produce an output.

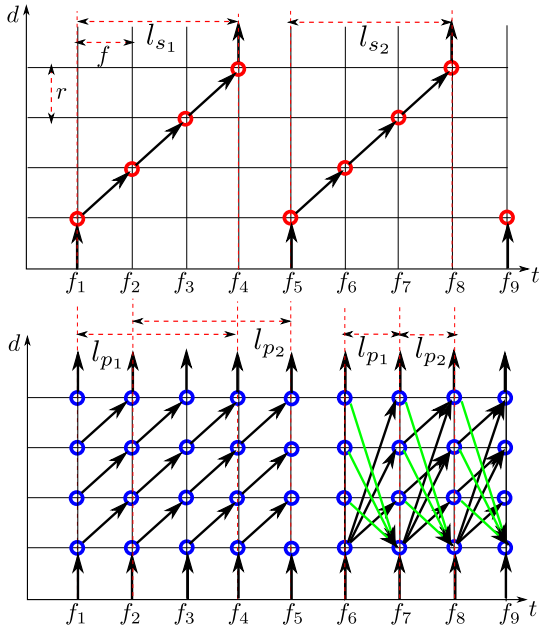


**Figure 1: Top: Standard (sequential) deep video network. Bottom left: pipelined model where layers execute in parallel resulting in higher throughput. Bottom right: pipelined model with feedforward and feedback skip connections, which reduce latency. Vertical axis represents network depth, horizontal axis represents time measured in video frames. For simplicity, we assume each layer takes as much time to execute as loading one frame.**

### 1.1 Pipelining with skip connections

State-of-the-art computer vision systems that operate on videos, e.g. object detectors [10, 11], are most of the time inspired by image models and have, at any point in time, a single layer that is actually *active*, i.e. a single layer that processes the current frame, whilst all the other layers (maybe hundreds of them) wait for their turn to process the data. This is due to the dependency rule that underlies the computation of a network's output: $o_d^i = \Gamma_d(o_{d-1}^{i-1})$ for $d \in \{1, D\}$, $i \in \{1, n\}$, $o_1^i = f_i$. This rule is *sequential in both depth and time*. Every new frame goes through the whole network, one layer

*Complete version of this manuscript is currently under review in CVPR2018.
†Shared first author.

at a time, before the output is known and another frame can be processed[1]. The latency of this sequential model for frame $f_i$ is given by $l_{s_i} = i \times D \times r - f \times (i-1)$. If $D \times r \leq f$, real-time execution is achieved for every frame. If not, the latency of the network increases over time almost linearly with the number of already processed frames. Existing models deal with this issue by skipping frames, artificially increasing $f$, i.e. lower frame rate and implicitly lower throughput $1/(D \times r)$; see fig. 1, top.

Mechanisms to counteract the high latency and low throughput due to sequential operation exist in both biological and human-designed systems. Biological neurons are not tremendously fast, but they come in large numbers and operate in a *massively parallel* fashion [16]. General-purpose computer processors use efficient pipelining strategies. We propose a similar pipelining design for deep video networks, which, in turn, enables depth-parallel processing; see Fig. 1, bottom left. The model still has real-time execution if $D \times r \leq f$. But if not, the model now has guaranteed constant latency $l_p = D \times r$, irrespective of the number of already processed frames. This latency is equal to the latency of the sequential model, but importantly, the throughput is high $1/r$ in this case.

Additionally, skip connections can be used to reduce this constant latency further so that real-time execution can be achieved irrespective of the depth of the model; see Fig 1, bottom right. However, feedforward skip connections can damage the performance of the network due to the reduced computation depth. To counteract this issue, we add feedback skip connections that enhance the context available at every time step.

## 1.2 Causal 3D ConvNets

Temporal tasks require scene representations that encode all the attributes relevant to the task, e.g. object category, shape, relative position, direction of motion etc. These representations must be continuously updated to reflect the scene dynamics. However, not all attributes change at the same rate: objects do not change category very often if at all, but their position can vary extremely fast. Using a unique update rate for the entire representation – as many deep video models currently do [1, 8, 9, 13, 14] – reveals as sub-optimal[2] [12]. We modify the model in Fig. 1, bottom right, such that the deeper layers responsible for extracting more and more abstract features, have an exponentially reduced update rate. We employ 3D filters to capture time dependencies and train the models using backprop-through-time, making sure the output for any new input frame depends only on past inputs while storing as internal state the set of activations required for all temporal kernels. When a new frame is available, the network needs to only extract the fast-varying (shallow) features from the new frame to produce an output.

# 2 EXPERIMENTS

The architecture we experimented with, called D3D, is shown in fig. 2, with output layers for action recognition and human pose heatmap estimation. We incorporate skip connections similar to

[1]Time-budget models [4, 6] use emergency exits to output the predictions computed thus far when time runs out, but they still process the data sequentially.

[2]Models using 3D convolutions incorporate different update rates naturally by using different time strides along depth [2, 15], but this usage is not trivial in causal settings.

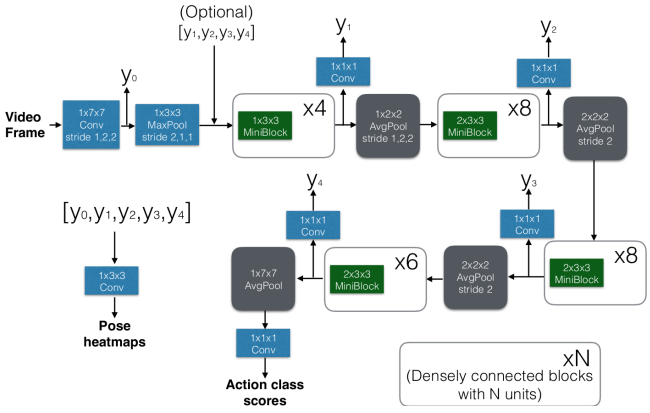


**Figure 2: D3D model composed of densely-connected blocks [3]. Training uses 64-frame sequences and backprop-through-time. Testing is done on whole videos. Each block shows kernel sizes (time x width x height) and strides.**

the popular DenseNet image model [3] – mini-blocks within each block send skip connections to all the other mini-blocks after it. In our case, these skip connections are in time, as shown in fig. 1. The top-down connections arise from the end of each block ($y_0$ to $y_4$). We use the Kinetics dataset [5] for both tasks. For actions, we use ground-truth labels and evaluate using top-1 accuracy. For pose, we generate "ground-truth" keypoints automatically using a state-of-the-art pose estimation model [7] and evaluate using cross-entropy loss on the test set.

We trained 3 versions of the D3D model with different degrees of parallelism: a fully sequential one, an almost fully-parallel model ($k = 2$), and a partly parallel model ($k = 10$), i.e. 10 sequential convolutions are performed before transferring the data to the next unrolling step. The results on the Kinetics test set are presented in table 1. It can be observed that the model with partial parallelism is

| Model | Act. recogn. | Pose estim. | Speedup |
|---|---|---|---|
| D3D-par ($k = 2$) | 53.9 | 0.39 | 9x |
| D3D-par ($k = 10$) | 64.9 | 0.29 | 3.3x |
| D3D-seq ($k = 58$) | 66.5 | 0.26 | 1x |

**Table 1: First column: D3D models with different levels of parallelism. Second column: test top1-accuracy for action recognition (higher is better). Third column: cross-entropy test loss for pose estimation (lower is better). Fourth column: throughput improvement (fps) achievable by parallelization.**

almost on-par with the sequential model, whereas the almost fully parallel one has a slightly lower performance. This points to the fact that our networks can be trained to withstand a significant degree of parallelism without major loss in performance. Our experiments were performed using TensorFlow, which does not support scheduling ops in parallel on the GPU, so we cannot measure yet practical efficiency gains, only a rough approximation of the upper bound shown in table 1, right column – this is left as future work.

## REFERENCES


[1] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. 2017. One-Shot Video Object Segmentation. In *Computer Vision and Pattern Recognition (CVPR)*.

[2] Joao Carreira and Andrew Zisserman. 2017. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[3] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[4] Sergey Karayev, Mario Fritz, and Trevor Darrell. 2014. Anytime Recognition of Objects and Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. http://sergeykarayev.com/recognition-on-a-budget/

[5] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. 2017. The Kinetics Human Action Video Dataset. *CoRR* abs/1705.06950 (2017). arXiv:1705.06950 http://arxiv.org/abs/1705.06950

[6] Stefan Mathe, Aleksis Pirinen, and Cristian Sminchisescu. 2016. Reinforcement Learning for Visual Object Detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[7] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. 2017. Towards Accurate Multi-person Pose Estimation in the Wild. *arXiv preprint arXiv:1701.01779* (2017).

[8] Viorica Pătrăucean, Ankur Handa, and Roberto Cipolla. 2016. Spatio-temporal video autoencoder with differentiable memory. In *International Conference on Learning Representations (ICLR) Workshop*.

[9] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. 2016. A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In *Computer Vision and Pattern Recognition*.

[10] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 6517–6525. https://doi.org/10.1109/CVPR.2017.690

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 91–99.

[12] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. 2016. *Clockwork Convnets for Video Semantic Segmentation*. Springer International Publishing, Cham, 852–868.

[13] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. 2015. Unsupervised Learning of Video Representations Using LSTMs. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15)*. JMLR.org, 843–852. http://dl.acm.org/citation.cfm?id=3045118.3045209

[14] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. 2017. Learning Video Object Segmentation With Visual Memory. In *The IEEE International Conference on Computer Vision (ICCV)*.

[15] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) (ICCV '15)*. IEEE Computer Society, Washington, DC, USA, 4489–4497. https://doi.org/10.1109/ICCV.2015.510

[16] Semir Zeki. 2015. A massively asynchronous, parallel brain. *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 370, 1668 (2015). https://doi.org/10.1098/rstb.2014.0174 arXiv:http://rstb.royalsocietypublishing.org/content/370/1668/20140174.full.pdf