Massively Parallel Video Networks

João Carreira[1], Viorica Pătrăucean[1(✉)], Laurent Mazare[1], Andrew Zisserman[1,2], and Simon Osindero[1]

[1] DeepMind, London, UK
joaoluis@google.com, viorica@google.com, laurent.mazare@gmail.com, zisserman@google.com, osindero@google.com
[2] Department of Engineering Science, University of Oxford, Oxford, UK

Abstract. We introduce a class of causal video understanding models that aims to improve efficiency of video processing by maximising throughput, minimising latency, and reducing the number of clock cycles. Leveraging operation pipelining and multi-rate clocks, these models perform a minimal amount of computation (e.g. as few as four convolutional layers) for each frame per timestep 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 illustrate the proposed principles by applying them to existing image architectures and analyse their behaviour on two video tasks: action recognition and human keypoint localisation. The results show that a significant degree of parallelism, and implicitly speedup, can be achieved with little loss in performance.

Keywords: Video processing · Pipelining · Depth-parallelism

1 Introduction

There is a rich structure in videos that is neglected when treating them as a set of still images. Perhaps the most explored benefit of videos is the ability to improve performance by aggregating information over multiple frames [1–3], which enforces temporal smoothness and reduces the uncertainty in tasks that are temporal by nature, e.g., change detection [4], computing optical flow [5], resolving action ambiguities (standing up/sitting down) [6] etc. An underexplored direction, however, is the ability to improve the processing efficiency. In this paper, we focus on this aspect in the context of the causal, frame-by-frame operation mode that is relevant for real-time applications, and show how to transform slow models to ones that can run at frame rate with negligible loss of accuracy.

J. Carreira and V. Pătrăucean—Shared first authors

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01225-0_40) contains supplementary material, which is available to authorized users.

© Springer Nature Switzerland AG 2018
V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11208, pp. 680–697, 2018.
https://doi.org/10.1007/978-3-030-01225-0_40

Most existing state-of-the-art computer vision systems, such as object detectors [7–9], process video frames independently: each new frame goes through up to one hundred convolutional layers before the output is known and another frame can be processed. This sequential operation in both depth and time can pose several problems: it can limit the rate at which predictions can be made, it can increase the minimum latency with which good predictions are available, and it can also lead to under-utilisation of hardware resources.

General-purpose computer processors encounter the same challenge when executing sequences of program instructions and address it with efficient pipelining strategies, that enable parallel computations. This also resembles the operation mode of biological neurons, which are not tremendously fast, but come in large numbers and operate in a massively parallel fashion [10].

Our proposed design employs similar pipelining strategies, and we make four contributions: first, we propose pipelining schemes tailored to sequence models (we call this *predictive depth-parallelism*); second, we show how such architectures can be augmented using *multi-rate clocks* and how they benefit from skip connections. These designs can be incorporated into any deep image architecture, to increase their throughput (frame rate) by a large factor (up to 10x in our experiments) when applied on videos. However they may also negatively impact accuracy. To reduce this impact, and as a third contribution, we show that it is possible to get better parallel models by *distilling* them from sequential ones and, as a final contribution, we explore other wiring patterns – *temporal filters and feedback* – that improve the expressivity of the resulting models. Collectively, this results in video networks with the ability to make accurate predictions at very high frame rates.

We will discuss related work in the next section. Then, we will move on to describe predictive depth-parallelism, multi-rate clocks and our other technical contributions in Sect. 3. In Sect. 4 we present our main experiments on two types of prediction tasks with different latency requirements: human keypoint localisation (which requires predicting a dense heatmap for each frame in a video); and action recognition (where a single label is predicted for an entire video clip), before the paper concludes.

2 Related Work

The majority of existing video models rely on image models [11–13] executed frame-by-frame, the main challenge being to speed up the image models to process sequentially 25 frames per second. This can be achieved by simplifying the models, either by identifying accurate architectures with fewer parameters [14], by pruning them post-training [15], or by using low-bit representation formats [16]. All of these can be combined with our approach.

A different type of model incorporates recurrent connections [17–19] for propagating information between time steps [18,19]. One simple propagation scheme, used by Zhu et al. [20] proposed periodically warping old activations given fresh external optical flow as input, rather than recomputing them. Our pipelining

strategy has the advantage that it does not require external inputs nor special warping modules. Instead, it places the burden on learning.

There are also models that consider the video as a volume by stacking the frames and applying 3D convolutions to extract spatio-temporal features [6,21]. These models scale well and can be trained on large-scale datasets [22–24] due to the use of larger temporal convolution strides at deeper layers. Although they achieve state-of-the-art performance on tasks such as action recognition, these methods still use purely sequential processing in depth (all layers must execute before proceeding to a next input). Moreover, they are not causal – the 3D convolutional kernels extract features from future frames, which makes it challenging to use these models in real-time.

In the causal category, a number of hierarchical architectures have been proposed around the notion of *clocks*, attaching to each module a possibly different clock rate, yielding temporally multi-scale models that scale better to long sequences [25]. The clock rates can be hard-coded [26] or learnt from data [27]. Some recent models [28,29] activate different modules of the network based on the temporal and spatial variance of the inputs, respectively, yielding adaptive clocks. There is also a group of time-budget methods that focuses on reducing latency. If the available time runs out before the data has traversed the entire network, then emergency exits are used to output whatever predictions have been computed thus far [30,31]. This differs from our approach which aims for constant low-latency output.

Ideas related to pipelining were discussed in [28]; a recent paper also proposed pipelining strategies for speeding up backpropagation for faster training in distributed systems [32–34]. Instead, we focus on pipelining at inference time, to reduce latency and maximise frame rate.

3 Efficient Online Video Models

Consider the directed graph obtained by unrolling a video model with n layers over time (see Fig. 1), where the layers of the network are represented by the nodes and the activations transferred between layers are represented by the edges of the graph. All the parameters are shared across time steps. Edges create dependencies in the computational graph and require sequential processing. Video processing can be efficiently parallelised in the offline case, by processing different frames in different computing cores, but not in the online case.

Depth-Parallel Networks. In basic depth-sequential video models, the input to each layer is the output of the previous layer at the same time step, and the network outputs a prediction only after all the layers have processed in sequence the current frame; see Fig. 1(a). In the proposed design, every layer in the network processes its input, passes the activations to the next layer, and immediately starts processing the next input available, without waiting for the whole network to finish computation for the current frame; Fig. 1(b). This is achieved by substituting in the unrolled graph the vertical edges by diagonal ones, so the input to each layer is still the output from the previous layer,

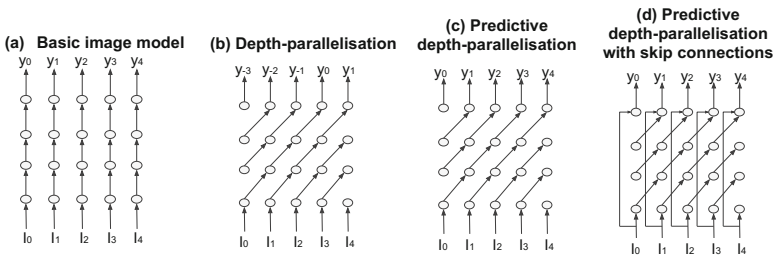

Fig. 1. Illustration of a standard sequential video model that processes frames independently, and depth-parallel versions. The horizontal direction represents the time and the vertical direction represents the depth of the network. The throughput of the basic image model depicted in **(a)** can be increased for real-time video processing using depth-parallelisation, shown in **(b)**. This makes it possible to, given a new frame, process all layers in parallel, increasing throughput if parallel resources are available. But this also introduces a delay of a few frames – in this example, the output at time t corresponds to the input at time $t-3$. It is possible to train the network to anticipate the correct output in order to reduce the latency **(c)**. This task can be made easier if the model has skip-connections, as illustrated in **(d)** – this way the model has access to some fresh features (albeit these fresh features have limited computational depth).

as usual, but *from the previous time step*. This makes it possible to process all layers at one time step in parallel, given enough computing cores, since there are no dependencies between them.

Latency and Throughput. We define *computational latency*, or just latency, as the time delay between the moment when a frame is fed to the network and the moment when the network outputs a prediction for that frame. It is the sum of the execution times of all layers for processing a frame. We consider *throughput* as the output rate of a network, i.e. for how many frames does the network output predictions for in a time unit. For the sequential model, throughput is roughly the inverse of the computational latency, hence the deeper the model, the higher the computational latency and the lower the throughput. Here resides a quality of the proposed depth-parallel models: irrespective of the depth, the model can now make predictions at the rate of its slowest layer.

It is useful to also consider the concepts of *information latency* as the number of frames it takes before the input signal reaches the output layer along the network's shortest path. For example, in Fig. 1, the information latency for the video model illustrated in (a) is 0, and for the model in (b) it is equal to 3. We define *prediction latency* as the displacement measured in frames between the moment when a network receives a frame and the moment when the network tries to emit the corresponding output. The prediction latency is a training choice and can have any value. Whenever the prediction latency is smaller than the information latency, the network must make a prediction for an input that it did not process yet completely.

For most of our experiments with depth-parallel models we used a prediction latency of zero based on the assumption that videos may be predictable over short horizons and we train the network to compensate for the delay in its inputs and operate in a predictive fashion; see Fig. 1(c). But the higher the information latency, the more challenging it is to operate with prediction latency of zero. We employ temporal skip connections to minimise the information latency of the different layers in the network, as illustrated in Fig. 1(d). This provides fresher (but shallower) inputs to deeper layers. We term this overall paradigm *predictive depth-parallelism*. We experimented thoroughly with the setting where prediction latency is zero and also report results with slightly higher values (e.g. 2 frames).

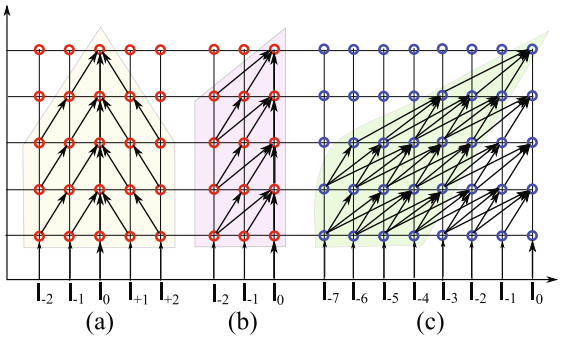

Fig. 2. Temporal receptive fields of: **(a)** standard; **(b)** causal; and **(c)** pipelined models.

Pipelined Operations and Temporal Receptive Field. Depth-parallelism has implications regarding the temporal receptive field of the network. In any standard neural network, by design, the temporal receptive field of a layer, i.e. the frames its input data comes from, is always a subset of the temporal receptive field of the next deeper layer in the network, resulting in a symmetric triangular shape; see Fig. 2(a). Stacked temporal convolutions and pooling layers are used for increasing the temporal visual field for deeper layers. In causal models the temporal receptive field is a right-angled triangle – no layer in the network has access to future frames; see Fig. 2(b). In the proposed design, the temporal receptive field along the depth of the network has a skewed triangular shape, the shallower layers having access to frames that the deeper layers cannot yet see (information latency). For example in Fig. 2(c), the latest frame that the deepest layer can see at time $t = 0$ is the frame I_{-4}, assuming a temporal kernel of 3, which, since we define a prediction latency of zero, means it must predict the output 4 frames in advance. Adding temporal skip connections reduces the information latency; at the extreme the receptive field becomes similar to the causal one, bringing it to zero.

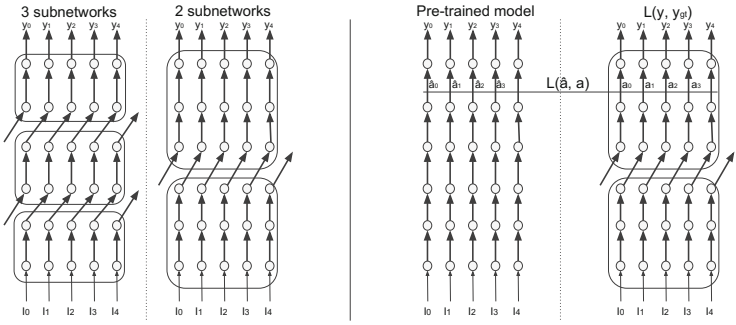

Fig. 3. Left: neural networks with three parallel subnetworks of two layers and two parallel subnetworks of three layers. **Right**: sequential-to-parallel distillation, the additional loss $L(\hat{a}, a)$ leverages intermediate activations of the pre-trained sequential model.

Levels of Parallelism. For simplicity, the proposed design ideas were illustrated in Fig. 1 using the "extreme" models, i.e.: (a) which is fully-sequential (with only vertical edges); and (b–c): which are fully parallel (lacking any vertical edge). However, there is a whole space of semi-parallel models in between, which makes it possible to trade off accuracy and efficiency.

A simple strategy to transform an image model with a linear-chain layer-architecture into a semi-parallel video model is to traverse the network starting from the first layer, and group together contiguous layers into sequential blocks of k layers that we will call *parallel subnetworks* and which can execute independently – see the two diagrams on the right side of Fig. 3, left; basic pseudocode is given in the supp. material.

3.1 Multi-rate Clocks

Features extracted deeper in a neural network tend to be more abstract and to vary less over time [28], obeying the so-called *slowness principle* [35] – fast varying observations can be explained by slow varying latent factors. For example, when tracking a non-rigid moving object, the contours, which are shallow features, change rapidly, but the identity of the object typically does not change at all. Since not all features change at the same rate as the input rate, it is then possible to reduce computation by reusing, and not recomputing, the deeper, more abstract, features. This can be implemented by having multi-rate clocks: whenever the clock of a layer does not tick, that layer does not compute activations, instead it reuses the existing ones. 3D ConvNets implement this principle by using temporal strides but does not keep state and hence cannot efficiently operate frame-by-frame. In our recurrent setting, multi-rate clocks can be implemented by removing nodes from the unrolled graph and preserving an internal state to cache outputs until the next slower-ticking layer can consume them. We used a set of fixed rates in our models, typically reducing clock rates by a factor

of two whenever spatial resolution is halved. Instead of just using identity to create the internal state as we did, one could use any spatial recurrent module (conv. versions of vanilla RNNs or LSTMs). This design is shown in Fig. 4(d).

For pixelwise prediction tasks, the state tensors from the last layer of a given spatial resolution are also passed through skip connections, bilinearly upsampled and concatenated as input to the dense prediction head, similar to the skip connections in FCN models [36], but arise from previous time steps[1].

3.2 Temporal Filters and Feedback

The success of depth-parallelism and multi-rate clocks depends on the network being able to learn to compensate for otherwise delayed, possibly stale inputs, which may be feasible since videos are quite redundant and scene dynamics are predictable over short temporal horizons. One way to make learning easier would seem to be by using units with temporal filters. These have shown their worth in a variety of video models [6,21,38]. We illustrate the use of temporal filters in Fig. 4(b) as *temporalisation*. Interestingly, depth-paralellisation by itself also induces temporalisation in models with skip connections.

For dense predictions tasks, we experimented with adding a feedback connection – the outputs of the previous frame are fed as inputs to the early layers of the network (e.g. stacking them with the output of the first conv. layer). The idea is that previous outputs provide a simple starting solution with rich semantics which can be refined in few layers – similar to several recent papers [39–43]. This design is shown in Fig. 4(c).

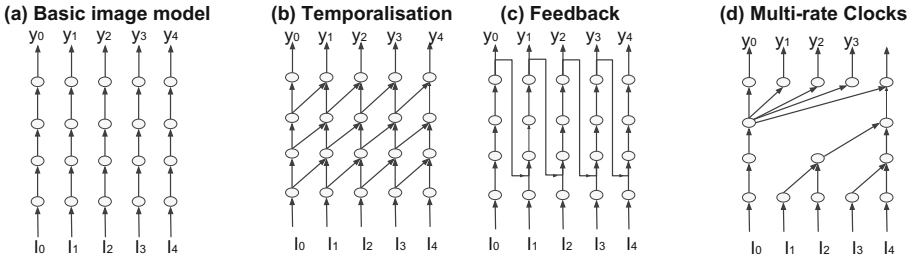

Fig. 4. Basic image models (left) can be extended along the temporal domain using different patterns of connectivity. Temporalisation adds additional inputs to the different computation nodes, increasing their temporal receptive field. Feedback re-injects past high-level activations to the bottom of the network. Both connectivity patterns aim to improve the expressivity of the models. For increasing throughput, having multi-rate clocks avoids always computing deeper activations (here shown for a temporal model), and instead past activations are copied periodically.

[1] More sophisticated trainable decoders, such as those in U-Nets [37], could also be used in a similar pipelined fashion as the encoder.

3.3 Sequential-to-Parallel "Distillation"

The proposed parallel models reduce latency, but their computational depth for the current frame at the moment where they produce an output is also reduced compared to their fully sequential counterparts; additionally they are designed to re-use features from previous states through the multi-rate clocks mechanism. These properties typically make learning more difficult. In order to improve the accuracy of our parallel models, we adopt a strategy similar to distillation [44], or to Ladder networks [45], wherein a *teacher* network is privileged relative to a *student* network, either due to having a greater capacity or (in the case of Ladder networks) access to greater amounts of information.

In our case, we consider the sequential model as the teacher, since all of its layers always have access to fresh features extracted from the current frame. We first train a causal fully-sequential model with the same overall architecture as the parallel model. Then we modify the loss of the parallel model to encourage its activations to match those of the sequential model for some given layers, while still minimising the original classification error, such that it predicts how the abstract features would have looked, had the information from the current frame been available. This is illustrated for one layer on the right side of Fig. 3. In our experiment we used the average of this new loss over $m = 3$ layers. The overall loss L_d with distillation is:

$$L_d = L(y, y_{gt}) + \lambda \sum_{i=1}^{m} \frac{1}{n_i} \left\| \hat{a}^{(i)} - a^{(i)} \right\|^2$$

where $L(y, y_{gt})$ is the initial cross-entropy loss between the predictions of the parallel network y and the ground truth y_{gt}, and the second term is the normalised Euclidean distance between the activations of the pre-trained sequential model $\hat{a}^{(i)}$ for layer i and the activation of the parallel model $a^{(i)}$ for the same layer; n_i denotes the number of feature channels of layer i. A parameter λ is used to weight the two components of the new loss. We set $\lambda = 1$ for the dense keypoint prediction and $\lambda = 100$ for action recognition.

4 Experiments

We applied the proposed principles starting from two popular image classification models: a 54 layer DenseNet [12] and Inception [11], which has 22 conv. layers. We chose these models due to their differences in connectivity. Inception has some built-in parallelism due to the parallel branches in the Inception blocks. DenseNet has no parallelism and instead has dense skip connections within blocks, which helps reduce information latency when parallelised. Full details on the architectures are provided in the supp. material.

We instantiated a number of model variations using the principles set in the previous section. In all cases we are interested in the online, causal setting (i.e. no peeking into the future), where efficiency matters the most. In the majority of the experiments we trained models with 0 prediction latency (e.g. the output at

time t should correspond to the input at time t), the most challenging setting. We name pipelined DenseNet models as Par-DenseNet and Inception-based models as Par-Inception.

For evaluation, we considered two tasks having different latency and throughput requirements: (1) action classification, where the network must output only one label prediction for the entire video sequence, and (2) human keypoint localisation, where the network must output dense per-frame predictions for the locations of human joints – in our case spatial heatmaps for the keypoints of interest (see Fig. 5).

Table 1. Test accuracy as percentage for action recognition on the miniKinetics dataset [46], using networks with multi-rate clocks and temporal filters. The number of parallel subnetworks is shown in the second column. For the semi-parallel case, Par-Inception uses 5 parallel subnetworks and Par-DenseNet 7. The non-causal, single subnetwork Par-Inception in the first row is equivalent to the I3D model [6].

Model	#Par. Subnets	Par-Inception Top-1	Par-Dense. Top-1
Non-causal	1	71.8	-
Sequential causal	1	71.4	67.6
Semi-parallel causal	5 (7)	66.0	61.3
Parallel causal	10 (14)	54.5	54.0

The dataset for training and evaluation in all cases was miniKinetics [46], which has 80k training videos and 5k test videos. MiniKinetics is a subset of the larger Kinetics [24], but more pratical when studying many factors of variation. For heatmap estimation we populated miniKinetics automatically with poses from a state-of-the-art 2D pose estimation method [47] – that we will call *baseline* from now on – and used those as ground truth. This resulted in a total of 20 million training frames[2].

4.1 Action Recognition

For this task we experimented with three levels of depth-parallelism for both architectures: fully sequential, 5, and 10 parallel subnetworks for Par-Inception models and fully sequential, 7, and 14 parallel subnetworks for Par-DenseNet models. Table 1 presents the results in terms of Top-1 accuracy on miniKinetics. The accuracy of the original I3D model [6] on miniKinetics is 78.3%, as reported in [46]. This model is non-causal, but otherwise equivalent to the fully sequential version of our Par-Inception[3].

[2] This is far higher than the largest 2D pose video dataset, PoseTrack [48], which has just 20k annotated frames, hardly sufficient for training large video models from scratch (although cleanly annotated instead of automatically).

[3] Note that this was pre-trained using ImageNet, hence it has a significant advantage over all our models that are trained from scratch.

There is a progressive degradation in performance as more depth-parallelism is added, i.e. as the models become faster and faster, illustrating the trade-off between speedup and accuracy. One possible explanation is the narrowing of the temporal receptive field, shown in Fig. 2. The activations of the last frames in each training clip do not get to be processed by the last classifier layer, which is equivalent to training on shorter sequences – a factor known to impact negatively the classification accuracy. We intend to increase the length of the clips in future work to explore this further. Promisingly, the loss in accuracy can be reduced partially by just using distillation; see Subsect. 4.3.

4.2 Human Keypoint Localisation

For this task we experimented with 5 different levels of depth-parallelism for Par-DenseNet: fully sequential and 2, 4, 7 and 14 parallel subnetworks. For Par-Inception, we used three different depth-parallelism levels: fully sequential, 5, and 10 parallel subnetworks. We employed a weighted sigmoid cross-entropy loss. Since the heatmaps contain mostly background (no-joint) pixels, we found it essential to weight the importance of the keypoint pixels in the loss – we used a factor of 10. For evaluation, we report results on the miniKinetics test set in terms of weighted sigmoid cross-entropy loss.

Results using the pipelining connectivity with multi-rate clock models are shown in Fig. 6, left. For both models, it can be observed that the performance improves as more layers are allowed to execute in sequence. Par-Inception has slightly better performance for higher degrees of parallelism, perhaps due to its built-in parallelism; Par-DenseNet models become better as less parallelism is used.

Since Par-DenseNet offers more possibilities for parallelisation, we used it to investigate more designs, i.e.: with/without multi-rate clocks, temporal filters and feedback. The results are shown in Fig. 6, right. Versions with temporal filters do better than without except for the most parallel models – these have intrinsically temporal receptive fields because of the skip connections in time, without needing explicit temporal filters. Feedback helps slightly. Clocks degrade accuracy a little but provide big speedups (see Subsect. 4.6). We show predictions for two test videos in Fig. 5.

4.3 Sequential to Parallel Distillation

As mentioned in Sect. 3, we investigated training first a sequential model, then fitting the parallel model to a subset of its activations in addition to the original loss function. This led to significant improvements for both models. The parallel causal Par-Inception model obtains a relative improvement in accuracy of about 12%, from 54.5% to 61.2% for action recognition. The improvement for multi-rate Par-DenseNet model on the keypoint localisation task is shown in Fig. 7.

4.4 Training Specifically for Depth-Parallelism

Is it important to train a model specifically for operating in parallel mode or can we rewire a pretrained sequential model and it will work just as well at inference time? We ran an experiment where we initialiased Par-DenseNet models with different levels of parallelism with the weights from the DenseNet fully sequential model and ran inference on the miniKinetics test set. The results are shown in Fig. 8, left, and indicate the importance of training with depth-parallelism enabled, so the network learns to behave predictively. We similarly evaluated the test loss of Par-DenseNet models with different levels of parallelism when initialised from a fully-parallel trained model. As expected, in this case the behaviour does not change much.

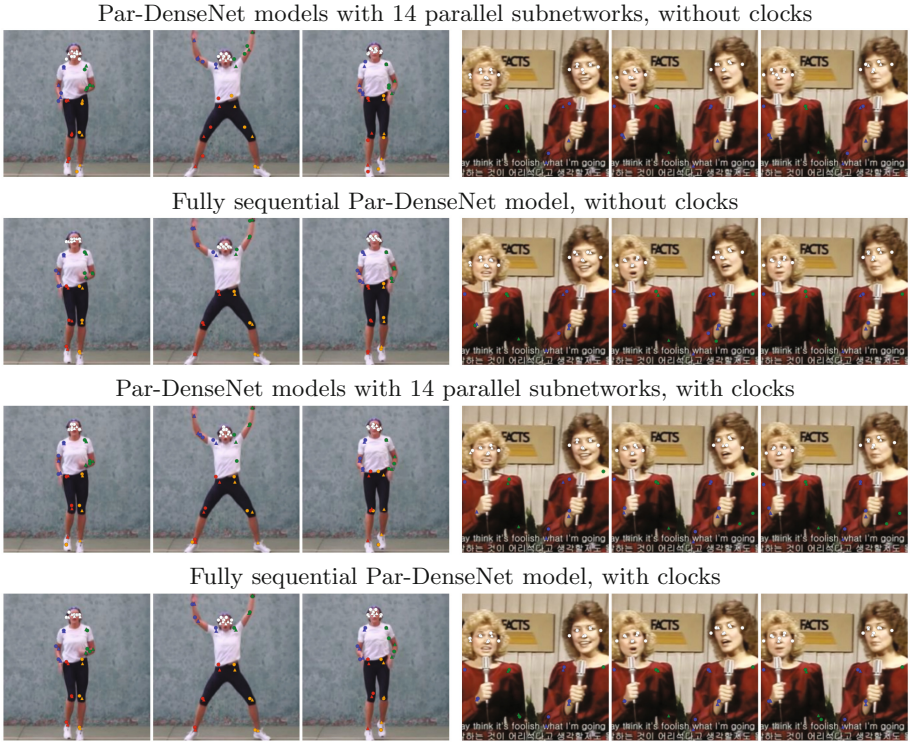

Fig. 5. Example outputs on a subset of frames one second apart from two videos of the miniKinetics test set. "Ground truth" keypoints from the model [47] used to automatically annotate the dataset are shown as triangles, our models predictions are shown as circles. Note that the parallel models exhibit some lag when the legs move quickly on the video on the left. Best seen zoomed on a computer screen in color. (Color figure online)

4.5 Effect of Higher Prediction Latency

All the results above were obtained when training for 0 frames of prediction latency. However, if a parallel model is several times faster than a sequential one, we can afford to introduce a prediction latency greater than zero frames. Figure 8, right, shows results for Par-DenseNet models in this setting. As expected, the test loss decreases as the prediction latency increases, since more layers get to process the input frame before a prediction needs to be made. Strikingly, by using a predictive delay of 2 frames, models with up to 4 depth-parallel subnetworks are as accurate as fully sequential models with 0 frame predictive latency.

4.6 Efficiency Measurements

In this section, we present the efficiency improvements achieved by the proposed models, comparing the cases with and without multi-rate clocks and with different numbers of parallel subnetworks. Our parallel models improve efficiency under the assumption that parallel computation resources are available. We benchmark our models on CPUs and GPUs by running inference on a CPU with 48 cores and on hosts with 2, 4, and 8 k40 GPUs, respectively. The GPUs were on the same machine to avoid network latency. For benchmarking, each model is run on 3000 frames and we average the time used to process each frame. Results are presented in Table 2. A figure illustrating the loss in accuracy as the throughput is increased can be found in the supp. material.

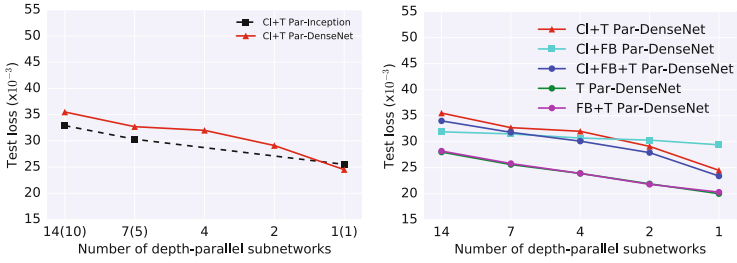

Fig. 6. Weighted sigmoid cross-entropy (lower is better) for human keypoint localisation on miniKinetics test set for zero prediction latency. "Cl" denotes models with multi-rate clocks, "T" – models with temporal filters, "FB" – models with feedback. **Left**: Comparison between Par-Inception and Par-DenseNet for different levels of parallelism. Note that in terms of number of sequential convolutions, 14 subnetworks for Par-DenseNet are equivalent to 10 subnetworks for Par-Inception, and similar for 7(5). **Right**: Variations of Par-DenseNet. In the absence of parallelisation (1 subnetwork), the accuracy of the best models with multi-rate clocks is just slightly worse to that of a much slower sequential model. Parallelisation penalises the accuracy of models with clocks more. The basic Par-DenseNet can have up to 4 parallel subnetworks with modest drop of accuracy.

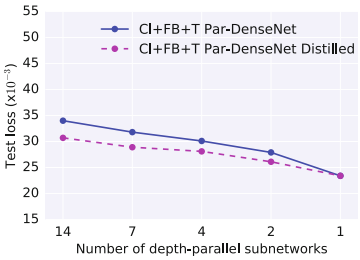

Fig. 7. Comparison between the weighted sigmoid cross-entropy (lower is better) of models with different levels of parallelism and the same models distilled from sequential for human keypoint localisation on miniKinetics test set for zero prediction latency. Results presented for a DenseNet model with multi-rate clocks ("Cl"), temporal filters ("T"), and feedback ("FB"). See text for details.

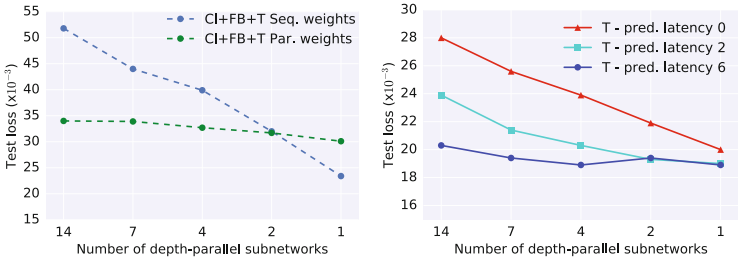

Fig. 8. Left: Seq. weights - Behaviour of Par-DenseNet with different levels of parallelism at inference time when trained with sequential connectivity. Par. weights - behaviour of Par-DenseNet with different levels of parallelism at inference time when trained with fully-parallel connectivity. **Right**: Test loss for Par-DenseNet when prediction latency is allowed to be greater than zero.

Our models are implemented using TensorFlow (TF) [49], hence: (1) when running on a multi-core CPU, we can run multiple operations in parallel and to parallelise a single operation, e.g., for conv layers. This means that the sequential model becomes faster with more cores, but only up to a certain point, when the overhead cancels out the gain from parallelism. The proposed parallel models benefit far more from having many CPU cores. (2) Multiple operations cannot run in parallel on the same GPU, hence there is little benefit in running our models on a single GPU. (3) A single operation cannot be split between GPUs. This explains why the sequential image model performance does not improve with more GPUs.

Par-DenseNet. Our Par-DenseNet architecture has a total of $4+8+8+6=26$ miniblocks so when using 14 parallel subnetworks, each parallel subnetwork is made of at most 2 miniblocks. When not using multi-rate clocks, 26 miniblocks are executed for each frame resulting in 416 miniblocks executions for a sequence of 16 frames. However when using multi-rate clocks, only 86 miniblocks are executed for such a sequence, which theoretically results in a speedup of $4.8\times$. We observe some smaller speedup but this is likely to be explained by the miniblocks having different sizes.

Par-Inception. Our models have 9 inception blocks. The most parallel version uses 10 parallel subnetworks: one for the initial convolutions and one for each inception block. For the sequential version, roughly a third of the time is spent on these initial convolutions. This explains why we do not observe speedups greater than 3 for the models without clocks when using more GPUs and we do not see much difference between using 4 and 8 GPU. More details together with execution timelines are included in the supp. material.

Table 2. Throughput improvement factors for Par-DenseNet and Par-Inception models relative to a sequential network without multi-rate clocks. For Par-DenseNet the fastest model processes 7x more frames per second, whereas the fastest Par-Inception model processes 5x more frames per second; see supp. material for absolute numbers in frames per second.

Model	# Par. subnets	48 cores	2 GPUs	4 GPUs	8 GPUs
Par-DenseNet without multi-rate clocks					
Sequential	1	1.0	1.0	1.0	1.0
Semi-parallel	2	1.3	1.6	1.7	1.7
Semi-parallel	4	1.8	1.7	2.5	2.9
Semi-parallel	7	2.2	1.6	2.6	3.7
parallel	14	2.6	1.7	2.7	3.8
Par-DenseNet with multi-rate clocks					
Sequential	1	2.6	3.4	3.4	3.4
Semi-parallel	2	3.0	3.9	4.0	4.0
Semi-parallel	4	3.6	4.5	5.1	5.2
Semi-parallel	7	4.6	4.5	5.6	6.1
Parallel	14	5.1	5.0	6.2	7.4
Par-Inception without multi-rate clocks					
Sequential	1	1.0	1.0	1.0	1.0
Semi-parallel	5	1.3	1.8	2.7	2.7
Parallel	10	1.3	1.8	2.6	2.6
Par-Inception with multi-rate clocks					
Sequential	1	2.4	2.6	2.6	2.6
Semi-parallel	5	3.0	3.4	5.0	5.0
Parallel	10	3.0	3.4	4.9	5.0

5 Conclusion

We introduced the paradigm of processing video sequences using networks that are constrained in the amount of sequential processing they can perform, with the goal of improving their efficiency. As a first exploration of this problem, we proposed a family of models where the number of sequential layers per frame is a design parameter and we evaluated how performance degrades as the allowed number of sequential layers is reduced. We have also shown that more accurate parallel models can be learned by distilling their sequential versions. We benchmarked the performance of these models considering different amounts of available parallel resources together with multi-rate clocks, and analysed the trade-off between accuracy and speedup. Interestingly, we found that the proposed design patterns can bring a speedup of up to 3× to 4× over a basic model that processes frames independently, without significant loss in performance in human action recognition and human keypoint localisation tasks. These are also general techniques – applicable to any state-of-the-art model in order to process video more efficiently. As future work we plan to investigate further the space of possible wirings using automated strategies.

Acknowledgements. We thank Carl Doersch, Relja Arandjelovic, Evan Shelhamer, and Dominic Grewe for valuable discussions and feedback on this work, and Tom Runia for finding typos in our architecture specification.

References

1. Jampani, V., Gadde, R., Gehler, P.V.: Video propagation networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 3154–3164. IEEE Computer Society (2017)
2. Pfister, T., Charles, J., Zisserman, A.: Flowing convnets for human pose estimation in videos. In: 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, 7–13 December 2015, pp. 1913–1921. IEEE Computer Society (2015)
3. Zhu, X., Wang, Y., Dai, J., Yuan, L., Wei, Y.: Flow-guided feature aggregation for video object detection. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, 22–29 October 2017, pp. pp. 408–417. IEEE Computer Society (2017)
4. Alcantarilla, P.F., Stent, S., Ros, G., Arroyo, R., Gherardi, R.: Street-view change detection with deconvolutional networks. In: Hsu, D., Amato, N.M., Berman, S., Jacobs, S.A. (eds.) Robotics: Science and Systems XII, University of Michigan, Ann Arbor, Michigan, USA, 18–22 June 2016 (2016)
5. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: FlowNet 2.0: evolution of optical flow estimation with deep networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 1647–1655. IEEE Computer Society (2017)
6. Carreira, J., Zisserman, A.: Quo Vadis, action recognition? A new model and the kinetics dataset. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 4724–4733. IEEE Computer Society (2017)

7. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 28, pp. 91–99. Curran Associates, Inc. (2015)
8. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. pp. 6517–6525. IEEE Computer Society (2017)
9. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, 22–29 October 2017, pp. 2980–2988. IEEE Computer Society (2017)
10. Zeki, S.: A massively asynchronous, parallel brain. Philos. Trans. R. Soc. Lond. B Biol. Sci. **370**(1668), 103–116 (2015)
11. Szegedy, C., et al.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, 7–12 June 2015, pp. 1–9. IEEE Computer Society (2015)
12. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 2261–2269. IEEE Computer Society (2017)
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014)
14. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. CoRR abs/1704.04861 (2017)
15. Chen, W., Wilson, J.T., Tyree, S., Weinberger, K.Q., Chen, Y.: Compressing neural networks with the hashing trick. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning, ICML 2015, vol. 37, pp. 2285–2294. JMLR.org (2015)
16. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or −1. arXiv e-prints abs/1602.02830, February 2016
17. Srivastava, N., Mansimov, E., Salakhutdinov, R.: Unsupervised learning of video representations using LSTMs. In: Bach, F.R., Blei, D.M. (eds.) Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015. JMLR Workshop and Conference Proceedings, vol. 37, pp. 843–852. JMLR.org (2015)
18. Pătrăucean, V., Handa, A., Cipolla, R.: Spatio-temporal video autoencoder with differentiable memory. In: International Conference on Learning Representations (ICLR) Workshop (2016)
19. Tokmakov, P., Alahari, K., Schmid, C.: Learning video object segmentation with visual memory. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, 22–29 October 2017, pp. 4491–4500. IEEE Computer Society (2017)
20. Zhu, X., Xiong, Y., Dai, J., Yuan, L., Wei, Y.: Deep feature flow for video recognition. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 4141–4150. IEEE Computer Society (2017)
21. Tran, D., Bourdev, L.D., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, 7–13 December 2015, pp. 4489–4497. IEEE Computer Society (2015)

22. Heilbron, F.C., Escorcia, V., Ghanem, B., Niebles, J.C.: Activitynet: a large-scale video benchmark for human activity understanding. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, 7–12 June 2015, pp. 961–970. IEEE Computer Society (2015)
23. Gu, C., et al.: AVA: a video dataset of spatio-temporally localized atomic visual actions. CoRR abs/1705.08421 (2017)
24. Kay, W., et al.: The kinetics human action video dataset. CoRR abs/1705.06950 (2017)
25. Koutník, J., Greff, K., Gomez, F.J., Schmidhuber, J.: A clockwork RNN. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21–26 June 2014. JMLR Workshop and Conference Proceedings, vol. 32, pp. 1863–1871. JMLR.org (2014)
26. Vezhnevets, A.S., et al.: Feudal networks for hierarchical reinforcement learning. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017. Proceedings of Machine Learning Research, PMLR, vol. 70, pp. 3540–3549 (2017)
27. Neil, D., Pfeiffer, M., Liu, S.: Phased LSTM: accelerating recurrent network training for long or event-based sequences. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, 5–10 December 2016, Barcelona, Spain, pp. 3882–3890 (2016)
28. Shelhamer, E., Rakelly, K., Hoffman, J., Darrell, T.: Clockwork convnets for video semantic segmentation. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9915, pp. 852–868. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49409-8_69
29. Figurnov, M., et al.: Spatially adaptive computation time for residual networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 1790–1799. IEEE Computer Society (2017)
30. Karayev, S., Fritz, M., Darrell, T.: Anytime recognition of objects and scenes. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, 23–28 June 2014, pp. 572–579. IEEE Computer Society (2014)
31. Mathe, S., Pirinen, A., Sminchisescu, C.: Reinforcement learning for visual object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016, pp. 2894–2902. IEEE Computer Society (2016)
32. Petrowski, A., Dreyfus, G., Girault, C.: Performance analysis of a pipelined backpropagation parallel algorithm. Trans. Neural Netw. **4**(6), 970–981 (1993)
33. Chen, X., Eversole, A., Li, G., Yu, D., Seide, F.: Pipelined back-propagation for context-dependent deep neural networks. In: INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, 9–13 September 2012, pp. 26–29. ISCA (2012)
34. Jaderberg, M., et al.: Decoupled neural interfaces using synthetic gradients. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017. Proceedings of Machine Learning Research, PMLR, vol. 70, pp. 1627–1635 (2017)
35. Wiskott, L., Sejnowski, T.J.: Slow feature analysis: unsupervised learning of invariances. Neural Comput. **14**(4), 715–770 (2002)
36. Shelhamer, E., Long, J., Darrell, T.: Fully convolutional networks for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(4), 640–651 (2017)

37. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
38. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, 8–13 December 2014, Montreal, Quebec, Canada, pp. 568–576 (2014)
39. Carreira, J., Agrawal, P., Fragkiadaki, K., Malik, J.: Human pose estimation with iterative error feedback. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016, pp. 4733–4742. IEEE Computer Society (2016)
40. Belagiannis, V., Zisserman, A.: Recurrent human pose estimation. In: 12th IEEE International Conference on Automatic Face & Gesture Recognition, FG 2017, Washington, DC, USA, 30 May–3 June 2017, pp. 468–475. IEEE Computer Society (2017)
41. Li, K., Hariharan, B., Malik, J.: Iterative instance segmentation. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016, pp. 3659–3667. IEEE Computer Society (2016)
42. Stollenga, M.F., Masci, J., Gomez, F.J., Schmidhuber, J.: Deep networks with internal selective attention through feedback connections. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, 8–13 December 2014, Montreal, Quebec, Canada, pp. 3545–3553 (2014)
43. Zamir, A.R., Wu, T., Sun, L., Shen, W.B., Shi, B.E., Malik, J., Savarese, S.: Feedback networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 1808–1817. IEEE Computer Society (2017)
44. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. CoRR abs/1503.02531 (2015)
45. Rasmus, A., Valpola, H., Honkala, M., Berglund, M., Raiko, T.: Semi-supervised learning with ladder networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, NIPS 2015, Cambridge, MA, USA, vol. 2, pp. 3546–3554. MIT Press (2015)
46. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning for video understanding. CoRR abs/1712.04851 (2017)
47. Papandreou, G., et al.: Towards accurate multi-person pose estimation in the wild. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
48. Iqbal, U., Milan, A., Andriluka, M., Ensafutdinov, E., Pishchulin, L., Gall, J., Schiele, B.: PoseTrack: a benchmark for human pose estimation and tracking. arXiv:1710.10000 [cs] (2017)
49. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). Software available from tensorflow.org