Gradient Forward-Propagation for Large-Scale Temporal Video Modelling

Mateusz Malinowski*
DeepMind

Dimitrios Vytiniotis
DeepMind

Grzegorz Świrszcz
DeepMind

Viorica Pătrăucean
DeepMind

João Carreira
DeepMind

Abstract

How can neural networks be trained on large-volume temporal data efficiently? To compute the gradients required to update parameters, backpropagation blocks computations until the forward and backward passes are completed. For temporal signals, this introduces high latency and hinders real-time learning. It also creates a coupling between consecutive layers, which limits model parallelism and increases memory consumption. In this paper, we build upon Sideways, which avoids blocking by propagating approximate gradients forward in time, and we propose mechanisms for temporal integration of information based on different variants of skip connections. We also show how to decouple computation and delegate individual neural modules to different devices, allowing distributed and parallel training. The proposed Skip-Sideways achieves low latency training, model parallelism, and, importantly, is capable of extracting temporal features, leading to more stable training and improved performance on real-world action recognition video datasets such as HMDB51, UCF101, and the large-scale Kinetics-600. Finally, we also show that models trained with Skip-Sideways generate better future frames than Sideways models, and hence they can better utilize motion cues.

1. Introduction

Popular deep video models generally rely on spatio-temporal convolutional networks (3D CNNs) [12, 18, 52, 55, 63] or recurrent networks [16, 24, 34] that are trained with backpropagation (BP) using stochastic gradient descent (SGD) [7, 19, 29, 33, 46, 57, 58, 60, 64]. This is a powerful training paradigm but also an expensive one as it needs to store all the activations in memory to compute Jacobian tensors for the gradient calculations. First, all the activations are computed in the forward mode, from the beginning to the end of the sequence. Next, gradients are computed in the reverse direction, from the end to the beginning. All that

severely limits the scale at which we can train temporal models on large-volume sequences such as videos. Therefore, typically these video models are trained on short video clips (about 2.5s at usual frame rate), in offline batch mode.

Sideways [38] is a recent training technique for video models that decouples the computation along the depth of the network and introduces *computation steps*. As a new frame is fed into the processing pipeline, each layer independently updates the internal state of the network by computing new activations and gradients. Next, these are passed to the layers above and below in the next computation step (see Figure 1 left). Moreover, the information cannot be back-propagated to the same units that produced the activations as this happened in the past computation steps. One may say that "everything flows forward in time", including the backward pass. Due to these properties, *Sideways* is more biologically plausible than the regular backprop, as it does not block the computation and respects the arrow of time.

Although *Sideways*, as originally proposed, operates in a temporal forward fashion, it is not a temporal model *per se*, as it has access only to the present frame at each time step. This results in improved memory efficiency similar to single-frame models, making it suitable for real-time applications. However, this comes with the cost of not integrating information temporally, which limits the expressive power of the resulting models.

In this work, we show that it is possible for models to process one frame at a time similar to *Sideways*, while still being able to extract temporal features. We do this by introducing shortcut (skip) connections in addition to direct connections between the layers of the model. We study the training dynamics of the resulting training procedure, which we call *Skip-Sideways*, and show that shortcut connections lead to more stable training and higher accuracy.

Regular skip connections [21] alter the information flow along the *data* path of the network by allowing activations to 'skip' layers, creating data shortcuts. In the proposed *Skip-Sideways*, activations and gradients along the shortcut connections are also sent forward in time, effectively creating data paths across time, making it possible to extract

---
*The corresponding author: mateuszm@google.com

temporal features. This change not only extends the modelling space, which subsumes spatio-temporal models, but also gives an interesting perspective on shortcut connections that are typically associated with *vanishing or exploding gradients* [20, 23], or *ensembles* [56].

To validate the proposed setting, we train a traditional image architecture, *e.g.*, VGG [49], on action recognition datasets [31, 50], including a large-scale one – Kinetics-600 [12] – by encapsulating the neural modules in *Skip-Sideways* units. Since there is no data dependency between units at any one time step, they can process the data in a depth-asynchronous fashion, maximising parallelism. This results in a significant speed-up and reduced latency. The efficiency of the whole network depends only on the worst-case efficiency of an individual *Skip-Sideways* unit.

To the best of our knowledge, *Skip-Sideways* is the first alternative to backpropagation, more biologically plausible, that is successfully used to train video models at scale.

2. Related Work

Pipelined training. We are inspired by recent works on pipelining during inference [11] and training [26, 38]. In particular, *Sideways* [38] derives approximate backpropagation rules by taking advantage of the smoothness of the input signal. Even though the authors show that training is possible even with the mismatch between gradients and activations, their method is unable to learn about scene dynamics during training and inference, and has not yet been demonstrated outside of small-scale settings. *GPipe* [26] is another training procedure that does exact backprop by splitting batches into micro-batches, allowing more efficient training. However, it maintains a 'bubble' to ensure gradient correctness (Figure 2 in [26]), hence it cannot use all the hardware resources in parallel, it needs to store activations, still blocks input frames, and due to the 'bubble' the performance is not the same across all frames in the sequence.

Shortcut connections. Skip connections have become ubiquitous design elements in modern neural networks [8, 9, 20, 21, 25, 45, 51, 53]. These modules were originally introduced to improve training dynamics, by allowing layers to learn residual representations w.r.t. their preceding layers. This alleviates the modelling burden and results in better propagation of gradients to shallower layers through shortcut connections. Here, the skip connections do not only shortcut between layers along depth, but also along time, by sending activations forward in time [11, 36]. In this way, we can turn frame-based models into spatio-temporal ones.

Alternatives to backpropagation (BP). Even though BP is the most commonly used method for training neural networks, multiple works have shown its limitations. BP's biological plausibility is questionable [5, 6, 30, 35], as the chain rule blocks computations during the forward and backward passes and uses symmetric weights in both passes [41]

– this operating regime is unlikely to exist in biological systems. Moreover, this also results in high-latency and low throughput in training and deployment of temporal models [11, 38], as it unnecessarily couples consecutive layers of the neural network architecture, limiting model parallelism and increasing both memory and energy consumption.

Various decoupling strategies [1, 2, 3, 4, 11, 14, 15, 27, 37, 38, 41, 42] deal with some shortcomings of BP, but are often applied to visually simpler domains, or static images, or require extra buffers, or investigate solely a biological aspect. Here, we are interested in decoupling rules that work well at scale and are applicable to temporally smooth input signals such as videos, that have slowly varying features [62].

Spatio-temporal models. Spatio-temporal convolutional neural networks (3D CNNs) [12, 18, 52, 55, 63] have become popular recently, mostly due to their performance. However, since they process the whole sequence at once, they are unsuitable for real-time applications and are expensive for representing longer videos, as their memory footprint and training latency grow with the length of the sequence when trained with BP. Recurrent neural networks (RNN) [16, 24, 34] aggregate information over time and are more suitable for online and causal settings. However, the same memory and latency problems encountered when training 3D CNNs with BP are also present here as RNNs require costly backprop-through-time (BPTT) for training [47, 57, 59]. BPTT scales linearly with the length of the input signal and requires propagation of information forward and backward in time. Likewise, causal 3D CNNs[43] still uses costly BP training that has to be done offline. In our work, we investigate the setting where: (1) models learn in real-time, (2) with a direct access only to present data frame, (3) the information (activations, gradients) is sent only forward in time, (4) the training latency is low, and (5) we can distribute and parallelize computations.

Recent works [5, 11, 22, 38] model temporal connectivities between modules by connecting consecutive layers at different time steps. Such a structure has implications on learning representation [5, 22] and efficiency [11, 38] as it allows depth-parallel computations of activations and pseudo-gradients. Our work belongs to the latter category and we leave learning representation as a future direction.

Forward propagation in time. Regular training of temporal models with BP or BPTT suffers from the *timing problem*. The common assumption is that the computation of activations and gradients and their propagation are instantaneous [6, 38] – this is not physically possible. Real Time Recurrent Learning (RTRL) [39, 40, 61] and *Sideways* [38] attempt to mitigate this issue. RTRL computes correct gradients in the forward mode. However, this approach scales poorly for larger input signals as it requires costly high-rank tensor computations. *Sideways* and our *Skip-Sideways*, on the other hand, use approximate gradients in the derivations.

3. Background

This section formalizes the learning problem and briefly describes *Sideways* [38] – the method upon which we build.
Notation. We use the same notation as [38]. We denote input temporal sequences by $\boldsymbol{x} = (\boldsymbol{x}^{[t]})_{t=1}^{K}, \boldsymbol{x}^{[t]} \in \mathbb{R}^d$ and outputs by $\boldsymbol{y} = (\boldsymbol{y}^{[t]})_{t=1}^{K}, \boldsymbol{y}^{[t]} \in \mathbb{R}^{d_y}$. To compute Jacobians we use the following notation. Let $\frac{\partial h_l}{\partial h_{l-1}}(\boldsymbol{h}_{l-1}, \boldsymbol{\theta}_l) = \frac{\partial H(h, \theta)}{\partial h}\Big|_{h=\boldsymbol{h}}$ be the Jacobian tensor of $H(h, \theta)$ with respect to the variable h evaluated at $h = \boldsymbol{h}, \theta = \boldsymbol{\theta}$. Note that, we use bold fonts to distinguish between formal and actual values of the variables. We present complete notation in the supplementary material.
Real-time learning. We are mostly interested in efficient, low-latency training, suitable for real-time applications. Therefore, we propose the following real-time constraints on our learning problem. We assume that the model has no direct access to future data, *i.e.* causal setting, and has limited direct access to its past inputs. That is, at time-step t the learning problem is just as a function of $\{(\boldsymbol{x}^{[t-k]}, \boldsymbol{y}^{[t-k]}), \dots, (\boldsymbol{x}^{[t]}, \boldsymbol{y}^{[t]})\}$. In this work, we only investigate the case with $k = 0$. Moreover, training needs to also scale gracefully to longer sequences, so ideally the memory consumption is independent of the inputs sequence length, and the training does not block computations allowing processing the stream of input data at sufficiently high frame-rate without dropping or buffering input frames. In our case, we use 25fps, which is common in video modelling.
Reverse and forward automatic differentiation. BP implements reverse mode for gradient calculation. This allows for efficient gradient calculations by keeping all the tensors low-rank. More precisely, in the reverse mode, we always combine low-rank gradients with higher-rank Jacobian tensors yielding low-rank outputs. That is, if $\nabla_{h_l} l \in \mathbb{R}^n$ and $\frac{\partial h_l}{\partial h_{l-1}}(\boldsymbol{h}_{l-1}, \boldsymbol{\theta}_l) \in \mathbb{R}^{n^2}$, we have $\nabla_{h_l} l \frac{\partial h_l}{\partial h_{l-1}}(\boldsymbol{h}_{l-1}, \boldsymbol{\theta}_l) \in \mathbb{R}^n$ as opposed to the forward mode $\frac{\partial h_l}{\partial h_{l-1}}(\boldsymbol{h}_{l-1}, \boldsymbol{\theta}_l) \frac{\partial h_{l+1}}{\partial h_l}(\boldsymbol{h}_l, \boldsymbol{\theta}_{l+1}) \in \mathbb{R}^{n^2}$ implemented by RTRL [61]. The downside of BP is that it has to store in memory the activations over time until the gradients/Jacobians can be computed.
Sideways alters the chain rule of regular BP. It does so by taking into account the passage of time to evaluate individual components of the chain rule at different time steps. The easiest way to think about it is through the composition of two functions. BP assumes the whole process is instantaneous and, in practice, it blocks computations. Therefore, if $F = G \circ H(\boldsymbol{x}^{[1]})$, we have $\frac{\partial f}{\partial x}(\boldsymbol{x}^{[1]}) = \frac{\partial g}{\partial h}(H(\boldsymbol{x}^{[1]})) \cdot \frac{\partial h}{\partial x}(\boldsymbol{x}^{[1]})$. In this way BP combines Jacobians of the *same* input frame. In contrast, *Sideways* [38] avoids blocking computations by combining Jacobians of *different* input frames and computing *approximate* gradients; we will refer to these approximate gradients as *pseudo-gradients* in the following. For the composition above, *Sideways* com-

putes $\frac{\partial f}{\partial x}(\boldsymbol{x}^{[1]}) = \frac{\partial g}{\partial h}(H(\boldsymbol{x}^{[1]})) \cdot \frac{\partial h}{\partial x}(\boldsymbol{x}^{[2]})$, so that two inputs can be processed at the same time, and it approximates BP if $\boldsymbol{x}^{[1]} \approx \boldsymbol{x}^{[2]}$. Figure 1 (left) illustrates the idea. Note that since there are no vertical arrows to introduce dependencies, all the layers of the network can be executed in parallel using an appropriate distributed hardware setting. The error introduced by this scheme is difficult to formally quantify. *Sideways* relies on the fact that the input sequences are smooth and are sampled with a sufficiently high frame-rate, which was empirically shown to hold for real videos and hence validating *Sideways*. We assume the same here.

4. *Skip-Sideways*

Single-frame (2D) CNNs do not have access to motion information other than perhaps through motion blur or semantics. The same holds for *Sideways*[1]. This narrow temporal receptive field motivates us to extend *Sideways* by introducing shortcut connections. Figures 1 and 2 illustrate the difference between *Sideways* and *Skip-Sideways*. We can see that due to the temporal aggregation happening in the forward propagation of both activations and pseudo-gradients through the shortcut connections, individual units have indirect access to more frames. Moreover, their receptive fields increase linearly with the depth of the network, similarly to 3D CNNs with stride 1 [12, 28, 54, 55]. We formalise the forward propagation of activations and pseudo-gradients with shortcut connections in the next paragraphs.
Forward-propagation of activations (FA). *Skip-Sideways* has two types of connections: direct connections, similar to *Sideways* depicted in Figure 2 with solid arrows, and shortcut connections shown in Figure 2 right, with dotted arrows. In this setting, one unit can receive information from several shallower units, *e.g.*, to pool activations from two units, following formula is used:

$$\boldsymbol{\gamma}_l^{[t]} = \boldsymbol{h}_{l-1}^{[t-1]} \oplus \tau \boldsymbol{h}_{l-2}^{[t-1]} \tag{1}$$

where $\oplus$ is a fusion operator (either addition or concatenation), and τ denotes the operation to match dimensions between $\boldsymbol{h}_{l-1} \in \mathbb{R}^{d_{l-1}}$ and $\boldsymbol{h}_{l-2} \in \mathbb{R}^{d_{l-2}}$, *e.g.*, max-pool or tile. We can also extend the notation above to k units, *i.e.*,

$$\boldsymbol{\gamma}_l^{[t]} = \boldsymbol{h}_{l-1}^{[t-1]} \oplus \tau \boldsymbol{h}_{l-2}^{[t-1]} \oplus \dots \oplus \tau^{k-1} \boldsymbol{h}_{l-k}^{[t-1]}$$

where $\tau^2 = \tau \circ \tau$ and $\circ$ is composition. For the sake of clarity, we keep our discussion constrained to two blocks only. Note that *Sideways* uses a special case when $\boldsymbol{\gamma}_l^{[t]} = \boldsymbol{h}_{l-1}^{[t-1]}$.

The choice of fusion operator $\oplus$ in Equation 1 subtly influences the training behaviour. Both addition and concatenation are used in the literature, *e.g.*, addition in ResNets [21],

---

[1]Note that, due to *Sideways* connectivity, the model does combine some information from different frames by using Jacobians originating in different frames within one training iteration (see Figure 1 left). However, this is a weak form of integration and it happens only during the backprop phase.
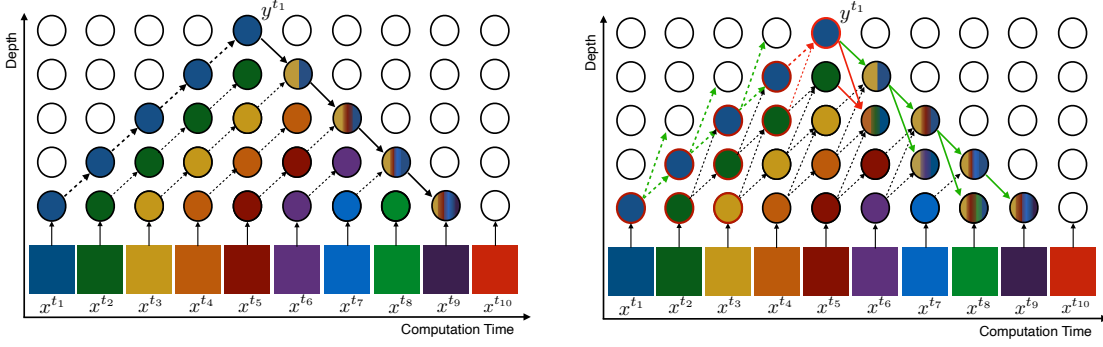
Figure 1: *Sideways* (left) and our *Skip-Sideways* (right) unrolled over computation steps. The coloured squares indicate data frames. Circles indicate neural modules inside *Sideways* or *Skip-Sideways* units. We exemplify a fragment that corresponds to the forward pass from $\boldsymbol{x}^{[t_1]}$ (dotted arrows) along with the backward pass (solid arrows). In *Skip-Sideways*, green arrows illustrate that the information is sent to different units in the next computation step. Red arrows illustrate how information is integrated from different units in the previous computation step. Empty circles indicate units outside of the exemplified fragment. Circles with a single colour indicate where the information has originated from. Circles with multiple colours indicate how the information is combined together in the backward pass (we only exemplify a few such units). Individual colours of these multi-coloured circles indicate where the information has originated from. For instance, combining 'blue gradient' with 'yellow activations' yields 'blue-yellow gradient' (6th computation step and second-last unit). Circles with the red border show the temporal receptive field of the top unit during the forward pass in *Skip-Sideways*. Best viewed in color.

and concatenation in U-NET [48]. Addition is a commutative operation which might prevent the network from differentiating between activations coming from direct or shortcut connections. Concatenation, on the other hand, does not suffer from this ambiguity and allows the network to learn appropriate weights to aggregate the information coming from direct and shortcut connections, respectively. Note also that concatenation is well suited to pool over units with different numbers of channels as $\boldsymbol{h}_l \oplus \boldsymbol{h}_k \in \mathbb{R}^{d_l} \bigoplus \mathbb{R}^{d_k} \cong \mathbb{R}^{d_l + d_k}$ and therefore we do not need to perform extra operations to match the channel dimension; this comes however with an increase in the number of model parameters. We report experiments and ablations for both fusion options.

Forward propagation of pseudo-gradients. To account for the aggregation of activations through schortcut connections, we distribute pseudo-gradients using shortcut connections that mirror their counterparts. Figure 2 shows that a single *Sideways* unit (red circle in the left figure) takes activations $\boldsymbol{h}_{l-1}^{[t-1]}$ and pseudo-gradient $\widetilde{\nabla}_l^{t-1} l$ as inputs, combines them, and outputs new activations $\boldsymbol{h}_l^{[t]}$ and pseudo-gradients $\widetilde{\nabla}_{l-1}^t l$ that are next sent towards the future computation steps upwards and downwards, respectively.

In *Skip-Sideways* (Figure 2, right), we integrate activations from direct and shortcut connections, $\boldsymbol{h}_{l-1}^{[t-1]}, \tau \boldsymbol{h}_{l-2}^{[t-1]}$ together with pseudo-gradients $\widetilde{\nabla}_{h_l}^{t-1} l$ and $\widetilde{\nabla}_{\tau h_l}^{t-1} l$. Next, the neural unit processes such information and redistributes activations through direct and shortcut connections, $\boldsymbol{h}_l^{[t]}$ and $\tau \boldsymbol{h}_l^{[t]}$, together with the corresponding pseudo-gradients,

$\widetilde{\nabla}_{h_{l-1}}^t l$ and $\widetilde{\nabla}_{\tau h_{l-2}}^t l$.

Using total derivative, we can write formally the forward propagation of pseudo-gradients used in *Skip-Sideways*:

$$\widetilde{\nabla}_{\gamma_l}^t l = \left(\widetilde{\nabla}_{h_l}^{t-1} l + \widetilde{\nabla}_{\tau h_l}^{t-1} l \right) \cdot \frac{\partial h_l}{\partial \gamma_l}(\boldsymbol{h}_{l-1}^{[t-1]}, \boldsymbol{h}_{l-2}^{[t-1]}, \boldsymbol{\theta}_l) \quad (2)$$

$$\widetilde{\nabla}_{h_{l-1}}^t l = \widetilde{\nabla}_{\gamma_l}^t l \cdot \frac{\partial \gamma_l}{\partial h_{l-1}}(\boldsymbol{h}_{l-1}^{[t-1]}, \boldsymbol{h}_{l-2}^{[t-1]}) \quad (3)$$

$$\widetilde{\nabla}_{\tau h_{l-2}}^t l = \widetilde{\nabla}_{\gamma_l}^t l \cdot \frac{\partial \gamma_l}{\partial h_{l-2}}(\boldsymbol{h}_{l-1}^{[t-1]}, \boldsymbol{h}_{l-2}^{[t-1]}) \quad (4)$$

$$\widetilde{\nabla}_{\theta_l}^t l = \left(\widetilde{\nabla}_{h_l}^{t-1} l + \widetilde{\nabla}_{\tau h_l}^{t-1} l \right) \cdot \frac{\partial h_l}{\partial \theta_l}(\boldsymbol{h}_{l-1}^{[t-1]}, \boldsymbol{h}_{l-2}^{[t-1]}, \boldsymbol{\theta}_l) \quad (5)$$

where rules 3 and 4 'redistribute' the pseudo-gradients during their forward propagation.

Note that we deliberately define shortcut connections in the way that the approximation comes only from the temporal misalignment between pseudo-gradients and Jacobians also present in *Sideways*.

5. Experiments

Tasks and datasets. We show the benefits of using the proposed setting *Skip-Sideways* over BP and *Sideways* using two tasks: action recognition and future frame prediction, and on three datasets: HMDB51 [31], UCF101 [50], and Kinetics-600 (K600) [12]. In the action recognition task, a single action, represented as a class, is present throughout an entire video sequence, *i.e.*, $\boldsymbol{y}^{[t]} = \boldsymbol{y}^{[t-1]}$ for all t. In the future frame prediction task, the ground truth output $\boldsymbol{y}^{[t]}$ changes at each time-step t. All videos have frame-rate of about 25fps, so that they satisfy our assumptions regarding smoothness
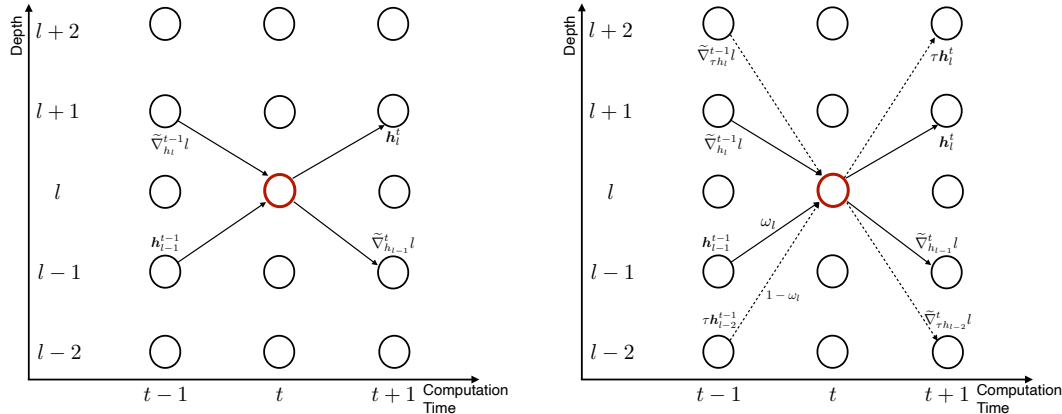
Figure 2: Illustration of the data flow of *Sideways* (left) and *Skip-Sideways* (right). The red circle in the center represents a single unit. It takes activations and pseudo-gradients from the previous step as inputs, processes them, and sends new activations and pseudo-gradients to the next computation step.

and slowly varying features. HMDB51 and UCF101 are used in *Sideways*, so for comparison, we use these same datasets in our ablation studies. To showcase our method *Skip-Sideways* in large-scale settings, we use Kinetics-600 dataset. We use resolutions 112×112 and 224×224 and train on 64 frames for HMDB51, UCF101, and on 32 frames for K600, with standard input augmentation [12].

As we are interested in an alternative to BP that is more scalable and suitable for video modeling, we train all the models from scratch; using features pre-trained with standard BP would defeat the goal of the study. Better results are reported in the literature, but those works use non-causal settings, or use extra features such as flow, extra modalities, different backbones, or various pre-training techniques. These directions are orthogonal to our work, where we investigate alternatives to BP on temporal sequences, in the real-time setting, with no access to future frames.

Distributed setup. For our experiments, we use a distributed hardware with 8 devices, and we split the network modules among these available devices. As visual backbones in *Sideways*, *Skip-Sideways*, and in baselines trained with standard BP, we use VGG8 [49] for the HMDB51 and UCF101 experiments to have direct comparison with *Sideways*, and a deeper VGG16 for the Kinetics-600 (K600) experiments. VGG8, having 8 modules, runs as a fully-parallel model when distributed on our hardware, whereas for VGG16, we assign pairs of consecutive modules to every device. The computation for the modules on the same device happens in sequence using regular BP, and the *Skip-Sideways* direct and shortcuts connections are applied between devices. We provide more details about the experimental setting in the supplementary material.

Baselines. We have implemented multiple baselines for our experiments. First, we re-implemented causal 3D VGG16

based on [43]. This baseline is close to our *Skip-Sideways* as it does not see future frames and it is capable of extracting temporal features. However, it is trained with regular BP.

As results for *Sideways* on large-scale datasets like Kinetics-600 are not reported in the literature, we have re-implemented *Sideways*. We have obtained slightly better results than those reported in the original paper [38]. We include the results from our implementation throughout this section.

For comparison with existing works in terms of hardware efficiency, we also implemented a video model using GPipe-style training [26]. We provide the details of the implementation in the supplementary material.

For the reconstruction task, we use a simple 16-layer deep convolutional model, without pooling or striding, hence maintaining full-resolution throughout the depth of the model; we denote this model as Full-Res. We depart from the traditional encoder-decoder design for two main reasons. First, we hypothesize that preserving higher spatial resolution is important for a dense prediction task like reconstruction. Second, and more interesting in the context of our work, even though typically such a design decision would lead to prohibitively expensive (memory- or speed-wise) training, we can readily afford to train such models by distributing *Skip-Sideways* units to different devices, using the distributed hardware efficiently. We include further details of the architecture in the supplementary material.

5.1. Results

***Skip-Sideways* vs *Sideways*.** Table 1 shows our main results comparing our proposed *Skip-Sideways* to *Sideways* [38]. *Skip-Sideways* significantly improves over *Sideways*, about 4, 2, and 6 per cent points (pp) respectively, showing the benefit of the temporal integration of information.

	HMDB51	UCF101	K600
Sideways [38]	27.3	62.5	55.7
Skip-Sideways	**31.4**	**64.5**	**61.8**

Table 1: *Skip-Sideways* vs *Sideways* [38] on HMDB51, UCF101, and Kinetics-600 (K600); resolution 112×112. We use VGG8 for HMDB51 and UCF101, and VGG16 for K600. We report accuracy in %. *Skip-Sideways* uses concatenation as aggregation operator.

Ablation	Fusion	HMDB51	UCF101
Sideways [38]	-	27.3	62.5
Skip-Sideways	add	29.6	62.8
Skip-Sideways	**concat**	31.4	**64.5**
FA-*Skip-Sideways*	concat	29.7	61.9

Table 2: Ablations on HMDB51, UCF101; resolution 112×112. We use VGG8 for HMDB51 and UCF101. We use addition (add) or concatenation (concat) operator to integrate activations. We report accuracy in %.

Ablation	HMDB51		UCF101	
	reg.	wide	reg.	wide
Sideways [38]	27.3	27.8	62.5	64.6
Skip-Sideways	31.4	31.5	64.5	66.0

Table 3: Regular vs. wide models trained with *Sideways* or *Skip-Sideways*. Wide *Sideways* has the same number of parameters as *Skip-Sideways*. Accuracy of the latter is superior (27.8 vs 31.4). We report accuracy in %.

Training	256 frames
Sideways + VGG16	40
Skip-Sideways + VGG16	40
BP + 3D VGG16	-
BP + 3D VGG16 (Remat)	1

Table 4: The largest batch-size that can be fit into a single GPU under the same setting, for different training schemes. Fitting a batch-size 1 for 3D VGG16 results in out-of-memory (OOM) error. (Remat) uses rematerialization [32].

Ablations. Our ablation studies justify different design choices used in *Skip-Sideways* and compare the results with *Sideways* in Table 2.

As the first analysis, we investigate the effect of using shortcut connections only in the forward pass to aggregate activations, without distributing the pseudo-gradients as de-tailed in Section 4. We report the results in Table 2, 4th row, FA-*Skip-Sideways*. Although FA-*Skip-Sideways* improves the results slightly over *Sideways*, re-distributing the pseudo-gradients as proposed in our full *Skip-Sideways* brings considerable additional improvement.

The choice of fusion operator is important, with a clear preference towards concatenation, *i.e.*, *Skip-Sideways* with concatenation outperforms the *Sideways* baseline by about 4 and 2 per cent points (pp), respectively, and on K600 by 6 pp (see Table 1).

To rule out that the increased performance of *Skip-Sideways* with concatenation is due to the larger number of parameters rather than the temporal integration of information, we run experiments with wider VGG8, so that this wider VGG8 and *Skip-Sideways* with concatenation have similar number of parameters. We report these results in Table 3. For HMDB51, we can see larger improvement by 3.6pp by adding shortcut connections than merely by adding more parameters. The results are less clear for UCF101 where models benefit from more parameters. Nonetheless, in both cases, training with *Skip-Sideways* significantly improves the results over *Sideways*.

Stability. Similar to the original *Sideways* paper [38], we study the training stability with respect to the input framerate, learning rate, and the number of video cuts ('montage'). We have found that *Skip-Sideways* training is slightly more stable than *Sideways*. Interestingly, lower input frame-rate affects less the *Skip-Sideways* performance compared to *Sideways*, which collapses when the frame rate drops. This robustness shows the benefit of faster data paths obtained through shortcut connections. More details on this analysis are included in the supplementary material ('Stability'), where we also show that the method converges for lower learning rates under reasonable smoothness assumptions.

Memory consumption. Our training scheme significantly reduces memory consumption compared to the standard BP training. This is because *Skip-Sideways* is a temporal training scheme that forward-propagates all the information (activations and pseudo-gradients), without the need to store activations throughout the sequence length like in BPTT. To illustrate this behaviour, we have trained 3D VGG16 (with BP), *Sideways*, and *Skip-Sideways*, and we measured the maximal possible batch-size on a single Quadro P5000 (16GB) in all three settings, for the same sequence length. Table 4 shows that *Sideways* and *Skip-Sideways* training allows for much larger batches, whereas BP training has resulted in out-of-memory error. This result opens the door to dealing with long videos, beyond the commonly used 2-3 second clips, or to methods that need very large batches such as contrastive methods [13], or alternative and more expensive architectural designs such as Full-Res. Note that *Skip-Sideways* has a memory usage similar to *Sideways* even though, advantageously, the former has a temporal receptive

field growing linearly with the number of *Skip-Sideways* units, whereas the *Sideways*'s temporal receptive field is one.

Speedup. Besides the important memory savings, another important practical advantage of our proposed training mechanism is the increased speedup obtained due to model parallelism. Table 5 shows our results where we distribute each *Skip-Sideways* unit on a different device (with 8 devices in total). We compare *Skip-Sideways* with the regular BP (running on a single device[2]) and its distributed version, GPipe [26], running on 8 devices. The number of steps per second reported in Table 5 represent the number of whole training iterations per second, including: fetching data[3], forward and backward passes, and weight updates. As we can see, *Skip-Sideways* is almost three times faster than the regular BP training and about two times faster than GPipe. It is also the only one that satisfies our real-time requirements (Section 3) as it can process frames at 25fps, it is causal, and is independent of the sequence length, being able to scale to potentially infinite sequences. Moreover, we also compare our model parallelism with batch parellelism (the last two rows in Table 5), where we train the network with total batch-size 8 distributed among eight devices. Note that these parallelism schemes are complementary and *Skip-Sideways* can also be used with batch-parallelism. In the supplementary material we provide more results.

***Skip-Sideways* vs BP.** In Table 6, we compare VGG16 trained with *Skip-Sideways* to other models trained with BP. Since *Skip-Sideways* is a causal model, for fair comparison, we have implemented a causal equivalent of 3D VGG16 and trained it with BP. First, we can see that there is little difference between I3D [10, 12], an established model for action recognition, and a 3D CNN with the VGG16 backbone [49] that we denote as 3D VGG16 (see 1st and 2nd rows). Hence using 3D VGG16 for action recognition is a sensible choice. Next, we can observe that there is a significant decrease in accuracy when transforming the 3D VGG16 into a causal one (1st and 3rd rows). Our *Skip-Sideways* (last row) outperforms the causal 3D VGG16 showing that it is a valid and competitive training mechanism, while being real-time. We have also found that using another visual backbone (InceptionNet) gives similar results to VGG16.

Note that the state-of-the-art results on Kinetics-600 are about 83% [17, 44]. As mentioned above, these results are generally obtained using various mechanisms that are orthogonal to our work. Here, we use standard visual backbones and focus on alternative to BP training that satisfies real-time requirements and is more scalable.

Future frame prediction. Future frame prediction is our second task. Figure 3 shows results when training models

---
[2]Note that, because of the dependencies between layers, even if we distribute BP training on different devices, the results would be slower due to communication overhead.

[3]Our asynchronous data fetching has a negligible cost.

	$\frac{\text{steps}}{\text{sec}}$	Devices
Skip-Sideways + VGG16	2.3	1
Skip-Sideways + VGG16	3.6	2
Skip-Sideways + VGG16	**7.2**	8
GPipe [26] + VGG16	4.2	8
BP + 3D VGG16	1.9	1
BP + 3D VGG16 (Remat)	1.5	1
BP + Causal 3D VGG16	2.4	1
BP + I3D [12]	2.9	1
BP + 3D VGG16 (Batch-paral.)	5.5	8
BP + I3D (Batch-paral.)	6.5	8

Table 5: Speed comparisons of various training strategies and architectures in number of steps per second. Column *devices* indicates the number of parallel devices used for distributed training. All models use K600, 32 frames, resolution 224×224, and batch-size 8. (Remat) denotes rematerialization [32]. The last two rows are exceptions and use batch-size 1 per device. We use space-to-depth transformation.

with *Sideways*, *Skip-Sideways*. As expected, motion is a powerful cue to predict future video frames, and thereof our *Skip-Sideways* training significantly outperforms *Sideways* training in terms of L2-loss (Table 7). We can observe that *Sideways* produces blurry video frames, as it is unable to extract motion futures to reduce uncertainty over where the pixels will be moving in future frames. On the contrary, the predictions obtained with *Skip-Sideways* are much sharper.

Kinetics-600	Backbone	Causal	Accuracy
BP	3D VGG16	N	72.3
BP	I3D [10, 12]	N	71.9
BP	Causal 3D VGG16	Y	64.2
Skip-Sideways	VGG16	Y	67.3

Table 6: Results on Kinetics-600. We use resolution 224×224 and total batch-size 256 in our experiments.

Kinetics-600	L2-Error
Sideways [38]	0.107
Skip-Sideways	**0.073**
BP	-

Table 7: Results of Full-Res on Kinetics-600, trained with batch size 64, 112×112, to predict future frames at time-step $t + 8$, given the frames up to time-step t. We report pixel error as L2-norm. Training with BP in this configuration results in out-of-memory (OOM) error.
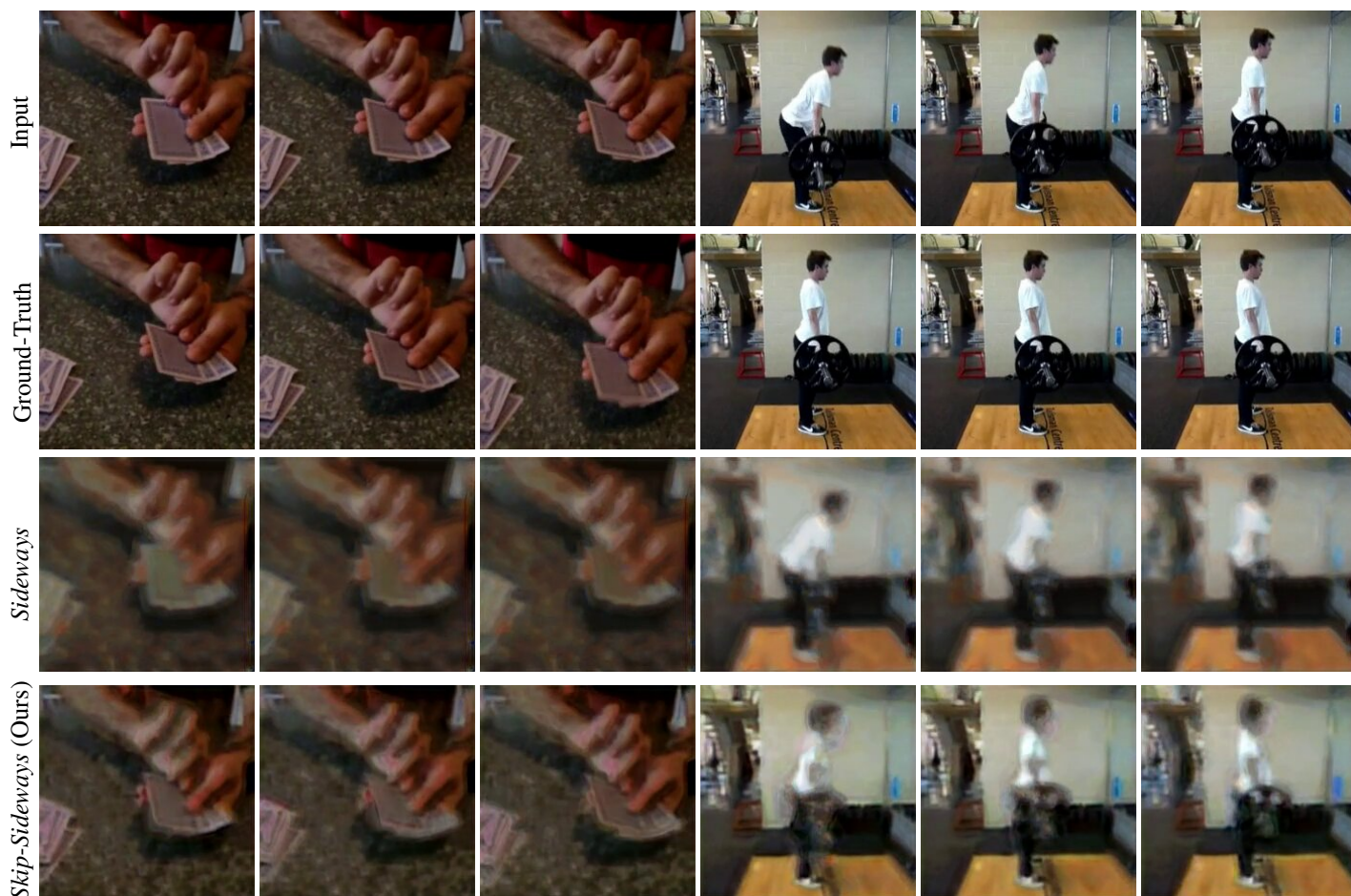
Figure 3: Qualitative results for two video sequences from the validation fold of Kinetics-600, where the models need to generate the frame at step $t + 8$ into future given frames up to t. From top to bottom: input frames, ground-truth output frames, predictions of Full-Res trained with *Sideways*, predictions of Full-Res trained with *Skip-Sideways*. We can observe that the predictions of the model trained with *Sideways* are blurry and semantically similar to inputs, indicating that the model lags behind. On the contrary, the predictions of the model trained with *Skip-Sideways* are sharper and semantically closer to the ground-truth output frames. We use resolution 224×224.

6. Conclusion

Alternatives to regular training based on backprop face many challenges as most of the existing frameworks and hardware systems are developed to optimise training based on backprop. Yet, even with all the existing optimisations, regular backprop has limitations as it does not scale up with the sequence length. It keeps large buffers of activations or gradients, and couples consecutive layers of the network, which blocks model parallelism along the depth of the model. These result in high latency systems, which may be unsuitable for real-time vision. In this work, we show that there are possible alternatives to backprop that reduce coupling between layers, decrease latency, allow for depth-parallelism, and can integrate information temporally, to obtain accuracy comparable to standard 3D CNNs. Our work is the first to showcase an alternative to backprop at scale and we hope that we will inspire further research into hardware architectures and software implementations that go beyond the standard backprop paradigm. As future work, we will investigate different architecture designs, mechanisms to augment the proposed *Skip-Sideways* with long-term memory capabilities [24], and investigate similar mechanisms for learning better representations.

References

[1] David Balduzzi, Hastagiri Vanchinathan, and Joachim Buhmann. Kickback cuts backprop's red-tape: Biologically plausible credit assignment in neural networks. In *AAAI Conference on Artificial Intelligence*, 2015.

[2] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. *arXiv preprint arXiv:1901.08164*, 2019.

[3] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems (NeurIPS)*, pages 153–160, 2007.

[4] Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

[5] Alessandro Betti and Marco Gori. Backprop diffusion is biologically plausible. *arXiv preprint arXiv:1912.04635*, 2019.

[6] Alessandro Betti, Marco Gori, and Giuseppe Marra. Backpropagation and biological plausibility. *arXiv preprint arXiv:1808.06934*, 2018.

[7] Léon Bottou and Yann L Cun. Large scale online learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2004.

[8] William L Briggs, Steve F McCormick, et al. *A multigrid tutorial*, volume 72. Siam, 2000.

[9] Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*, 2017.

[10] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-600 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.

[11] João Carreira, Viorica Patraucean, Laurent Mazare, Andrew Zisserman, and Simon Osindero. Massively parallel video networks. In *European Conference on Computer Vision (ECCV)*, 2018.

[12] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[14] Anna Choromanska, Benjamin Cowen, Sadhana Kumaravel, Ronny Luss, Mattia Rigotti, Irina Rish, Brian Kingsbury, Paolo DiAchille, Viatcheslav Gurev, Ravi Tejwani, et al. Beyond backprop: Online alternating minimization with auxiliary variables. In *International Conference on Machine Learning (ICML)*, 2019.

[15] Wojciech Czarnecki, Grzegorz Świrszcz, Max Jaderberg, Simon Osindero, Oriol Vinyals, and Koray Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. In *International Conference on Machine Learning (ICML)*, 2017.

[16] Debidatta Dwibedi, Pierre Sermanet, and Jonathan Tompson. Temporal reasoning in videos using convolutional gated recurrent units. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1111–1116, 2018.

[17] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[18] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 6202–6211, 2019.

[19] 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 preprint arXiv:1706.02677*, 2017.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision (ECCV)*, pages 630–645. Springer, 2016.

[22] Geoffrey Hinton. How to represent part-whole hierarchies in a neural network. *arXiv preprint arXiv:2102.12627*, 2021.

[23] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[25] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.

[26] Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[27] Zhouyuan Huo, Bin Gu, and Heng Huang. Training neural networks using features replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6659–6668, 2018.

[28] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 35(1):221–231, 2012.

[29] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[30] Jonas Kubilius, Martin Schrimpf, Aran Nayebi, Daniel Bear, Daniel LK Yamins, and James J DiCarlo. Cornet: modeling the neural mechanisms of core object recognition. *BioRxiv*, 2018.

[31] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2011.

[32] Ravi Kumar, Manish Purohit, Zoya Svitkina, Erik Vee, and Joshua Wang. Efficient rematerialization for deep networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[33] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

[34] Zhenyang Li, Kirill Gavrilyuk, Efstratios Gavves, Mihir Jain, and Cees GM Snoek. Videolstm convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 166:41–50, 2018.

[35] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7, 2016.

[36] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[37] Sindy Löwe, Peter O'Connor, and Bastiaan Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[38] Mateusz Malinowski, Grzegorz Swirszcz, Joao Carreira, and Viorica Patraucean. Sideways: Depth-parallel training of video models. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[39] Jacob Menick, Erich Elsen, Utku Evci, Simon Osindero, Karen Simonyan, and Alex Graves. A practical sparse approximation for real time recurrent learning. *arXiv preprint arXiv:2006.07232*, 2020.

[40] Asier Mujika, Florian Meier, and Angelika Steger. Approximating real-time recurrent learning with random kronecker factors. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6594–6603, 2018.

[41] Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[42] Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. *arXiv preprint arXiv:1901.06656*, 2019.

[43] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[44] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Xinmei Tian, and Tao Mei. Learning spatio-temporal representation with local and global diffusion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[45] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[46] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, 1951.

[47] AJ Robinson and Frank Fallside. *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering Cambridge, 1987.

[48] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[49] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[50] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[51] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[52] Jonathan Stroud, David Ross, Chen Sun, Jia Deng, and Rahul Sukthankar. D3d: Distilled 3d networks for video action recognition. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 625–634, 2020.

[53] Richard Szeliski. Locally adapted hierarchical basis preconditioning. In *ACM SIGGRAPH 2006 Papers*, pages 1135–1143, 2006.

[54] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *European conference on computer vision (ECCV)*, pages 140–153. Springer, 2010.

[55] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[56] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems (NeurIPS)*, pages 550–558, 2016.

[57] P. Werbos. Backpropagation through time: what does it do and how to do it. In *Proceedings of IEEE*, volume 78, 1990.

[58] Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pages 762–770. Springer, 1982.

[59] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.

[60] Paul John Werbos. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. Wiley-Interscience, New York, NY, USA, 1994.

[61] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[62] Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002.

[63] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv preprint arXiv:1712.04851*, 1(2):5, 2017.

[64] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *International Conference on Machine learning (ICML)*. ACM, 2004.