Bio-inspired parameter reuse: Exploiting inter-frame representation similarity with recurrence for accelerating temporal visual processing

Zuowen Wang Longbiao Cheng Joachim Ott Pehuen Moure Shih-Chii Liu
Institute of Neuroinformatics, University of Zurich and ETH Zurich
{zuowen, longbiao, jott, pehuen, shih}@ini.uzh.ch

Editors: Marco Fumero, Emanuele Rodolà, Clementine Domine, Francesco Locatello, Gintare Karolina Dziugaite, Mathilde Caron

Abstract

Feedforward neural networks are the dominant approach in current computer vision research. They typically do not incorporate recurrence, which is a prominent feature of biological vision brain circuitry. Inspired by biological findings, we introduce **RecSlowFast**, a recurrent slow-fast framework aimed at showing how recurrence can be useful for temporal visual processing. We perform a variable number of recurrent steps of certain layers in a network receiving input video frames, where each recurrent step is equivalent to a feedforward layer with weights reuse. By harnessing the hidden states extracted from the previous input frame, we reduce the computation cost by executing fewer recurrent steps on temporally correlated consecutive frames, while keeping good task accuracy. The early termination of the recurrence can be dynamically determined through newly introduced criteria based on the distance between hidden states and without using any auxiliary scheduler network. RecSlowFast **reuses a single set of parameters**, unlike previous work which requires one computationally heavy network and one light network, to achieve the speed versus accuracy trade-off. Using a new *Temporal Pathfinder* dataset proposed in this work, we evaluate RecSlowFast on a task to continuously detect the longest evolving contour in a video. The slow-fast inference mechanism speeds up the average frame per second by 279% on this dataset with comparable task accuracy using a desktop GPU. We further demonstrate a similar trend on CamVid, a video semantic segmentation dataset.

1 Introduction

Current deep learning based computer vision research is largely dominated by feedforward neural networks [8, 12]. However, various studies on biological visual systems have indicated the importance of recurrent neural activities in functionalities such as attentive and conscious vision [21], perceptual grouping [30], pattern completion [37] and object recognition [15, 16, 29]. These neurophysiological findings have inspired researchers to incorporate recurrence in their networks to achieve better performances or better biological plausibility. For example, the studies of [19, 20] show that the output of shallow recurrent CNNs aligns better with measurements from the ventral stream on an object recognition task, and a higher correlation corresponds to better generalization of the model. The authors in [17] propose that top-down and horizontal connections between neurons in the cortex are crucial for supporting incremental perceptual grouping and utilizing high-level object cues. A related work [24] demonstrates a single-layer recurrent CNN with substantially fewer parameters

Proceedings of the I edition of the Workshop on Unifying Representations in Neural Models (UniReps 2023).

outperforms all feedforward baselines in long-range spatial dependencies modeling. However, the traditional application of iterative recurrent structures involves imposing a fixed number of recurrent steps, disregarding the characteristics and demands of both the input data and the task.

It has been observed in [15] that primate visual systems required extended processing time for more complex object recognition tasks compared to the simpler control group. This phenomenon was attributed to recurrent activities within the visual ventral stream, suggesting that biological visual systems can dynamically allocate different amounts of computation depending on the complexity of the task at hand, all within the same biological network. Inspired by these findings, we developed the **Recurrent Slow-Fast (RecSlowFast)** framework that uses varying recurrent steps for different frames in a video and leverages layer reuse and the similarity between frames for better performance-speed trade-off and parameter efficiency. The major contributions are as follows[1]:

- Introduction of a biologically motivated recurrent neural network (RNN) framework named *recurrent slow-fast networks (RecSlowFast)*. RecSlowFast uses **within input timestep recurrence** achieved by layer reuse for dynamically adjusting the amount of computation spent on an input frame and **cross-input timestep recurrence** by leveraging hidden states from earlier frames to reduce the number of recurrent steps needed for subsequent frame processing.
- Based on the hidden states similarities obtained across input frames, we introduce a novel recurrence halting criteria that dynamically allocates computational resources (recurrent steps). Notably, our approach circumvents the need for an auxiliary neural network to make halting decisions.
- Construction of a Temporal Pathfinder dataset (*T-Pathfinder*), which extends the widely used static pathfinder variants [13, 17, 24, 25] to the temporal domain by gradually changing the lengths of the contours across frames. The dataset has two subsets with different difficulty levels. The RecSlowFast framework is evaluated and compared against other baselines including recurrent and feedforward convolutional networks on T-Pathfinder.

2 Related work

Recurrence in visual processing Recurrence has been found vital for various functionalities in the visual ventral stream. [15, 14] show it requires a longer processing duration for the visual ventral stream to solve difficult object recognition tasks than the easy ones. Several works focus on better neural recording alignment with recurrent structures for object recognition [19, 20, 16] and natural movie stimuli [34]. The authors of [28] found that architecture search with evolutionary algorithms yields layer-local recurrence and long-range feedback which achieve better trade-offs between task performance and small network size. Our work aims to exploit recurrence with temporal similarities of hidden states to reduce the amortized computation cost.

Iterative inference The architectural flexibility offered by recurrent structures through the configurable number of recurrent steps has long been of research interest. The early work of adaptive computing time (ACT) [11] introduces a halting unit into the RNN to terminate the recurrence. Similarly, [41] proposed a self-instantiating recurrent structure. In [22] the authors proposed a network that reuses the weights and concatenates the resulting feature map with the input of each residual block [12], achieving high parameter and computation efficiency compared with ResNet. Several other works [32, 4, 40] studied methods of deciding when to halt the recurrence, including the use of auxiliary gating neural networks [40]. In the two related works [33, 5] the authors discovered that RNNs are able to extrapolate to larger maze sizes or longer prefix sum length even when trained on smaller problem scales, by simply running more recurrent steps during inference. Different from these works, RecSlowFast employs the variable recurrent step inference for temporal visual processing to exploit the correlation and redundancy in the frames.

Implicit layers Another line of work attempts to model sequential data by directly finding these equilibrium points with root-finding [3, 7]. Such methods are equivalent to feedforward networks with infinite depths. Subsequent works [2, 27, 9] studied reusing the states obtained from the equilibrium point of the previous input timestep in order to accelerate the root-finding of the next timestep. Our work is different from implicit layer works in the sense that RecSlowFast does not require a root-finder solver which typically comes with additional computing and memory overhead and puts constraints on the types of usable layers. We conducted comparisons and the results are in A.3.

[1] code and dataset: `https://github.com/ZuowenWang0000/RecSlowFast_Neurips_Unireps`

Two-path video processing In the domain of video processing with deep neural networks, several studies have proposed ways of exploiting the temporal correlation nature of videos to accelerate the networks. One approach involves calculating optical flow between frames with a light network and wrapping the features extracted by a computationally heavy network [43, 39]. Another line of work [26, 42, 10] uses both a heavy and a light network to extract features in an interleaved manner with the heavy network operating less frequently. Unlike the aforementioned methods, our design utilizes a single network that operates in a two-path-like manner, as the amount of computation can be dynamically adjusted for an RNN.

3 Recurrent slow-fast networks

We hypothesize that the hidden state information from processing past temporally correlated visual inputs could accelerate the processing of incoming inputs. Taking two consecutive frames $\mathcal{I}_1$ and $\mathcal{I}_2$ in Figure 1(a) as an example, in a semantic segmentation task, it is much easier to complete the task for $\mathcal{I}_2$ after $\mathcal{I}_1$, despite minor changes on the sidewalk. Meanwhile, modifying the solution of $\mathcal{I}_2$ for processing $\mathcal{I}_3$ is slightly more challenging since the difference between $\mathcal{I}_2$ and $\mathcal{I}_3$ is larger than $\mathcal{I}_1$ and $\mathcal{I}_2$.

This concept forms the basis of our generic building block. Formally, for a given input sequence $x = (x_1, ..., x_T)$, consisting of either raw frames or feature maps, the corresponding hidden states h_t^n for timestep $t \in \{1, ..., T\}$ and the n^{th} recurrent step is computed by the recurrent function r_θ parameterized with θ as follow:

$$h_t^n := r_\theta(h_t^{n-1}, x_t) \tag{1}$$

$$h_{t+1}^0 := h_t^{N(t)} \tag{2}$$

where h_1^0 is the hidden states initialization h_{init} for the input, x_1, on the 1st timestep. $N(t)$ gives the number of recurrent steps to be executed for input x_t at timestep t, allowing a variable computation workload across inputs. For consecutive inputs x_t and x_{t+1}, we reuse the last hidden state resulting from x_t as the initial state for x_{t+1}, as shown in Equation 2. The h_{init} state is initialized to zero.

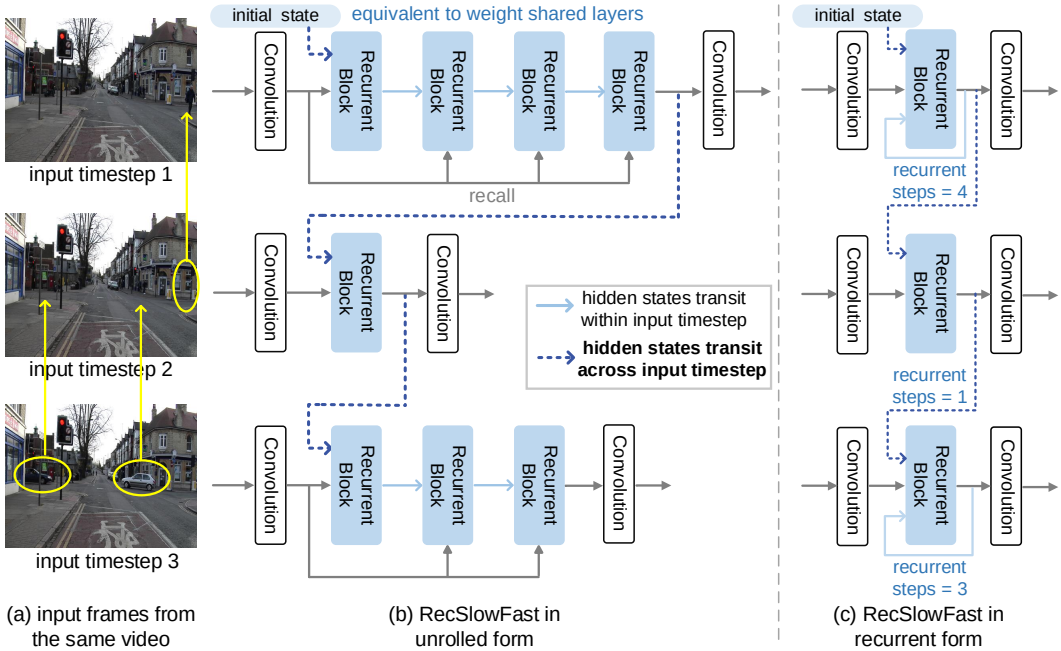


Figure 1: Illustration of the RecSlowFast framework. The three input frames are from the CamVid dataset [6]. The major differences between frames are highlighted with yellow circles and arrows.

There exist two types of hidden states transition in the RecSlowFast framework. One is *within input timestep transition* (depicted in Equation 1 and marked with solid blue arrows in Figure 1), which feedback the output hidden state h_t^{n-1} for computing the next hidden state h_t^n, on the same input

x_t. Another is *across input timestep transition* (depicted in Equation 2 and marked with dashed blue arrows in Figure 1). At the end of the recurrence for an input timestep t, we recycle the extracted hidden state $h_t^{N(t)}$ and use it as the initial state h_{t+1}^0 for the processing of the next input timestep $t+1$. For processing consecutive frames, the network does not have to start the processing from initialization state h_{init} but instead starts from the hidden state extracted on the previous frame, and this enables reduced computation while maintaining task accuracy. We verify this in Section 4.2 by comparing RecSlowFast with networks that do not conduct across input timestep transitions, namely $\forall t \in \{1, ..., T\}, h_t^0 = h_{\text{init}}$, which are called *Stateless* in the rest of the paper.

We also incorporate the findings in [5] of the *recall* connections, inspired by skip connections [12, 36] which pass information from an earlier layer to a later layer for more stable training of very deep networks. We use recall to stabilize the training of the RecSlowFast network. As shown in Figure 1 and Equation 1, the recall connections forward the same input x_t to every recurrent step at timestep t.

Instantiating RecSlowFast with horizontal gated recurrent unit and convolutional long short-term memory In this work, we focus on two versions of simple instantiations of recurrent block in RecSlowFast in order to rule out other deep network architectural factors and focus on studying the variable recurrent step mechanism. We use the horizontal gated recurrent unit (hGRU) [17] and convolutional long short-term memory (cLSTM) [35] in our experiment (complete description is in Appendix A.1). Both types of recurrent blocks are preceded and followed by two convolutional layers. Thus the whole network is described as follows:

$$x_t := f_{\text{conv1}}(I_t) \tag{3}$$

$$h_t^n := r_\theta(h_t^{n-1}, x_t), \quad n = 1, ..., N(t) \tag{4}$$

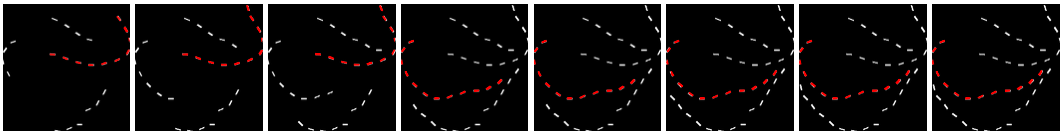
$$y_t^{\text{pred}} := f_{\text{conv2}}(h_t^{N(t)}) \tag{5}$$

where I_t is the raw input frame, f_{conv1} and f_{conv2} are the head and output convolutional layers.

4 The Temporal-Pathfinder dataset and experiment results

$\mathcal{I}_1$ $\mathcal{I}_2$ $\mathcal{I}_3$ $\mathcal{I}_4$ $\mathcal{I}_5$ $\mathcal{I}_6$ y_1^{GT} y_6^{GT}

(a) An example from *T-Pathfinder-easy*. y_1^{GT} and y_6^{GT} show the ground truth of $\mathcal{I}_1$ and $\mathcal{I}_6$.



$\mathcal{I}_1$ $\mathcal{I}_2$ $\mathcal{I}_3$ $\mathcal{I}_4$ $\mathcal{I}_5$ $\mathcal{I}_6$ $\mathcal{I}_7$ $\mathcal{I}_8$

(b) An example from *T-Pathfinder-Hard* where the longest contour is marked in red.

Figure 2: (a) and (b) shows two examples from the *T-Pathfinder-Easy* and *T-Pathfinder-Hard* subset respectively. For the example in (b) the ground truth longest contour is annotated with red color on top of the original contour, which is grey scale. Notice the changes between frames caused by the growing of contour segments in the longest contour as well as the distractors. See Appendix A.6 for the complete sequence and more examples. Zoom-in for better visibility.

4.1 Construction of the Temporal Pathfinder (*T-Pathfinder*) dataset

For evaluating the recurrent slow-fast framework, we first construct a new *T-Pathfinder* dataset, which is a temporally extended variant of several similar synthetic visual tasks [24, 17] inspired by cognitive science [13]. The *T-Pathfinder* dataset consists of videos of visually consecutive frames (see Figure 2 for examples). The starting frame of a video contains several contours, where one of them is longer than the rest. The shorter contours are *distractors*. In each succeeding frame of the video, every contour can grow longer by certain segments with a predefined probability. The task for the network

on this *T-Pathfinder* dataset is to detect the longest contour in each frame and output a dense binary segmentation map with two classes, where pixel value 1 indicates this pixel is a member of the longest contour, otherwise, the pixel value is 0.

We constructed two subsets with different difficulty levels, named *T-Pathfinder-Easy* and *T-Pathfinder-Hard*. In *T-Pathfinder-Easy*, the distractors are designed to never surpass the initial longest contour candidate, despite their random growths in each frame. In *T-Pathfinder-Hard*, there is a probability that distractors could grow longer than the initial longest contour candidate in the first frame. The bottom row in Figure 2 shows an example where one of the distractors grew from 9 segments in frame $\mathcal{I}_3$ to 12 segments in frame $\mathcal{I}_4$. Thus resulting in a new longest contour candidate for $\mathcal{I}_4$. In *T-Pathfinder-Hard*, if one or multiple distractors grew to the same length as the longest contour, the ground truth is still set to the previous longest contour.

All frames in the *T-Pathfinder* dataset have a resolution of 128×128 pixels. The *T-Pathfinder-Easy* and *T-Pathfinder-Hard* datasets, both begin with the creation of 4 distractors, but they differ in the length range and contour growth probability. In the *T-Pathfinder-Easy* dataset, each video consists of 6 consecutive frames. The longest contour in the initial frame is set to be 10 segments. Distractors have lengths uniformly and randomly chosen from a range of 1 to 3. The extension probability is 0.5 for every contour in each consecutive frame. In contrast, the *T-Pathfinder-Hard* dataset features 8 consecutive frames per video. The longest contour in the initial frame is set to be 10. The distractors have lengths uniformly and randomly chosen from a range of 1 to 8. Furthermore, the distractors grow in each new frame with a probability of 0.8, while the longest contour grows with a probability of 0.2. This makes the contours more likely to surpass the current longest contour. Each time a contour is allowed to grow, it does so uniformly and randomly with 1 to 3 segments. 10,000 sequences are generated for each subset and for each 2000 are randomly selected as test split.

4.2 RecSlowFast results on *T-Pathfinder*

We applied our RecSlowFast framework on the two *T-Pathfinder* subsets and compare them with baselines. We start our experiment with a *fixed schedule*. For both subsets of *T-Pathfinder*, the model uses more steps to process the first frame of the sequence, then uses a smaller or equal number of steps per frame to process the rest of the sequence. We annotate this type of fixed schedule with "**s**{*slow steps*}**f**{*fast steps*}", where s represents the number of recurrent steps executed on the beginning frame and f represents the number of steps per frame for the rest of the video. For example, "s6f1" in Table 1 indicates the hGRU model spends 6 recurrent steps on the first frame ($N(t = 1) = 6$), and 1 recurrent step for each of the rest ($N(t \neq 1) = 1$), while "s6f6" represents the model spending 6 recurrent steps on every frame of the video. Unless stated, we use the same schedule for training and inference. The computation complexity (excluding f_{conv1} and f_{conv2}) can be estimated by counting the total recurrent steps for the video. For a 6-frame-sequence, if the schedule is "s6f2", then in total $1 \times 6 + 5 \times 2 = 21$ recurrent steps will be spent for processing this sequence. In comparison with always spending 6 recurrent steps on every frame, the schedule "s6f2" will be approximately $(36 - 21)/36 \approx 42\%$ cheaper in computation for the recurrent block.

We use backpropagation through time (BPTT) for training the RecSlowFast framework. For each input frame $\mathcal{I}_t$ a loss $L_t(y_t^{\text{GT}}, y_t^{\text{pred}})$ is computed for the ground truth y_t^{GT} and prediction y_t^{pred} pair. The total loss is $L_{\text{total}} = (\sum_{t=1}^{T} L_t)/T$. To mitigate the excessive computational graph size and associated memory requirements during BPTT training, we truncate the gradient flow from later timesteps to earlier ones. As a result, BPTT is confined within each individual input timestep, significantly reducing memory demands. We use the focal loss [23] with $\gamma = 2$ as the loss function. It is used to address the class imbalance problem and prevents the model from always outputting zeros because the majority of pixels are zero in the ground truth frames. The Adam optimizer [18] with a starting learning rate of 0.001 is used and the learning rate is multiplied by 0.7 after every 25 epochs. Batch size 10 is used for all experiments. The validation split is randomly selected from 10% of the training split. We train in a total of 100 epochs and use the checkpoint with the best validation loss for testing. The evaluation metric used is the mean intersection of union (mIoU). All mIoU values presented are averages from three runs. The original mIoU scores and corresponding standard deviations can be found in the Appendix. All the frame per second (FPS) measurements were conducted on an Nvidia RTX 3080 GPU with an AMD Ryzen 7 5800X CPU.

Cross-input-timestep hidden state transition boosts temporal visual processing performance We first compare the RecSlowFast with their *Stateless* counterparts, which always reset the hidden state

T-Pathfinder-Easy										
hGRU	s6f6		s4f4		s2f2		s1f1		s6f1	
	mIoU	FPS	mIoU	FPS	mIoU	FPS	mIoU	FPS	mIoU	FPS
RecSlowFast	.994	110	.994	164	.966	311	.879	561	.992	342
Stateless	.972		.919		.685		.536		.586	

Table 1: RecSlowFast-hGRU task accuracy and inference speed on *T-Pathfinder-Easy*. For comparison, a feedforward CNN baseline with 6 convolutional layers and residual connections has mIoU of 0.895 and FPS of 213 with $\sim 3\times$ parameters of RecSlowFast-hGRU (846k vs. 284k). More detailed results and architecture description of the feedforward CNN baseline are in A.3 and A.2.

to initialization for every new input frame. Table 1 shows that when trained and tested with the same schedule, RecSlowFast always outperforms the Stateless version of the hGRU network. Meanwhile, even with **only 1** recurrent step per frame, the RecSlowFast-hGRU-s1f1 is able to outperform the Stateless hGRU with 4 recurrent steps per frame. A similar trend is observed on the more difficult *T-Pathfinder-Hard* subset. In Figure 3 both the hGRU and cLSTM instantiations of RecSlowFast are shown. Although the absolute value of mIoU is lower than on the *T-Pathfinder-Easy* subset, RecSlowFast always outperforms the Stateless version by a large margin, when the same recurrent block and schedule are being used. The RecSlowFast-hGRU-s3f3 beats the Stateless-hGRU-s8f8 in task accuracy while having an average frame per second $2.6\times$ more. For both the RecSlowFast framework and Stateless networks, increasing recurrent steps up to a certain number improves the task performance, even though the model does not use more parameters. This finding is similar to [5, 33, 19, 20], however, we show this improvement in a temporal visual processing task. The comparison between RecSlowFast and Stateless shows that the network is continuously improving the prediction based on the cross-input-timestep hidden states.

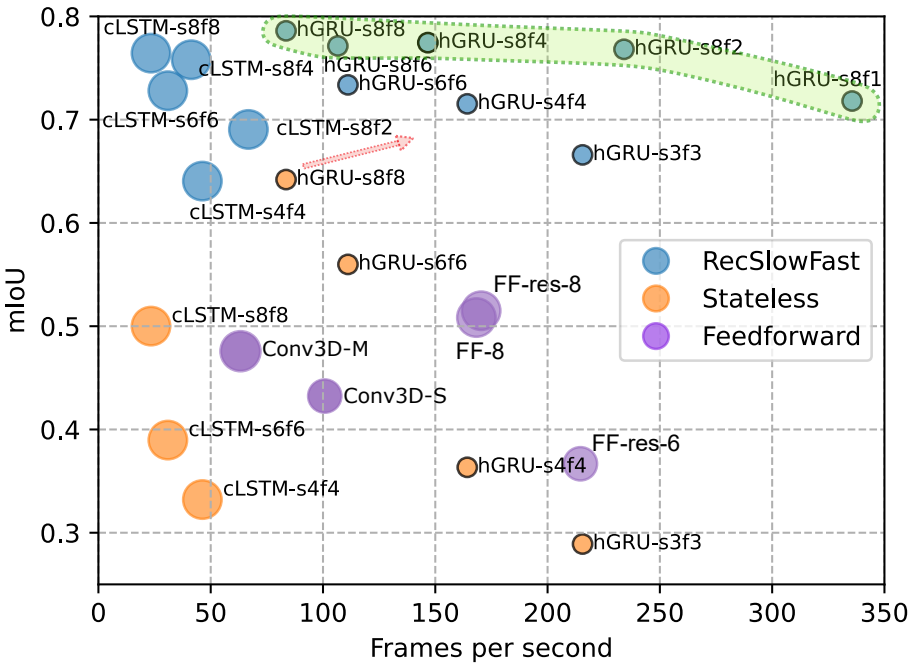


Figure 3: Different RecSlowFast schedules with hGRU and cLSTM instantiations, compared with their Stateless counterparts and feedforward CNN baselines on *T-Pathfinder-Hard*. The green shade shows the trend that we could reduce recurrent steps for non-initial frames to lower computation costs while maintaining performance. The red arrow indicates by reusing the hidden state from last timestep, the performance get boosted even with fewer recurrent steps. Circle sizes represent the number of parameters. Detailed descriptions for feedforward baselines FF-* and Conv3D-* are in Appendix A.2.

Reduced recurrence steps for non-initial frames lower computation cost Since the cross-input-timestep hidden state helps to improve the accuracy, one natural question is whether we need the same number of recurrent steps for processing the later input frames, even if the hidden states from the earlier timesteps are reused. We study this question by setting the number of fast steps differently from the slow steps for both training and testing. In Table 1 we studied the "s6f1" schedule for RecSlowFast-hGRU, which is $3.1\times$ faster than RecSlowFast-hGRU-s6f6, but is still able to have the same high mIoU. RecSlowFast-hGRU-s6f1 also outperforms RecSlowFast-hGRU-s1f1 by a large margin, indicating a better quality starting hidden state helps to solve the later frames more easily, under the assumption that the frames are temporally highly correlated. Figure 3 shows more fine-grained scheduling on *T-Pathfinder-Hard* with hGRU and cLSTM. RecSlowFast-hGRU-s8f2 is able to run $\sim 3\times$ as fast as "s8f8" with a small drop of the mIoU. For RecSlowFast with cLSTM, the speed up from "s8f8" to "s8f4" is $\sim 1.8\times$ with $\sim 0.6\%$ of absolute mIoU loss. Feedforward CNNs, although having the largest number of parameters, are worse than RecSlowFast-hGRUs at modeling the visual temporal sequence and having lower mIoUs in both *T-Pathfinder* subsets. In summary, these results demonstrate that by using cross-input-timestep hidden states, we can decrease the overall computational cost for temporal visual processing by employing fewer recurrent steps.

Correlation between input features distance and hidden states distance We attempt to heuristically explain why having fewer recurrent steps for subsequent frames is possible, especially when the input frame changes are small, by analyzing the relationship of the distances between input features of two consecutive frames and the distances between their last recurrent step hidden states, namely $d(x_t, x_{t+1})$ and $d(h_t^{N(t)}, h_{t+1}^{N(t+1)})$ where $d(\cdot, \cdot)$ denotes the distance metric. In our analysis, we first flatten the feature maps or hidden states and then calculate the ℓ_2 distances. We use the RecSlowFast-hGRU-s8f8 (one of the top performing checkpoints from Figure 3) to calculate the feature maps and hidden states. The first feature map x_1 and hidden state $h_{t=1}^{8}$ are not used to compute their distance with initialization h_{init}, since that will create a false strong correlation. The per video normalized distance correlation is shown in Figure 4(a). $\ell_2(x_t, x_{t+1})$ and $\ell_2(h_t^{N(t)}, h_{t+1}^{N(t+1)})$ are highly correlated with correlation coefficient 0.8. This shows that if the input feature x_{t+1} changed a lot from x_t, reflected by the large distance, then the distance between the end hidden states $h_{t+1}^{N(t+1)}$ and $h_t^{N(t)}$ will also likely to be large. Conversely, two similar or same feature maps will induce small hidden states distance in the RecSlowFast framework. Figure 4(b) shows an exemplary PCA visualization of hidden states. Four input frames with ground truth annotated with red are overlaid on top of the hidden state trajectory. From $\mathcal{I}_5$ to $\mathcal{I}_6$, lots of segments were added and the ground truth longest contour candidate also changed. This causes a long traverse of the hidden states when processing $\mathcal{I}_6$.

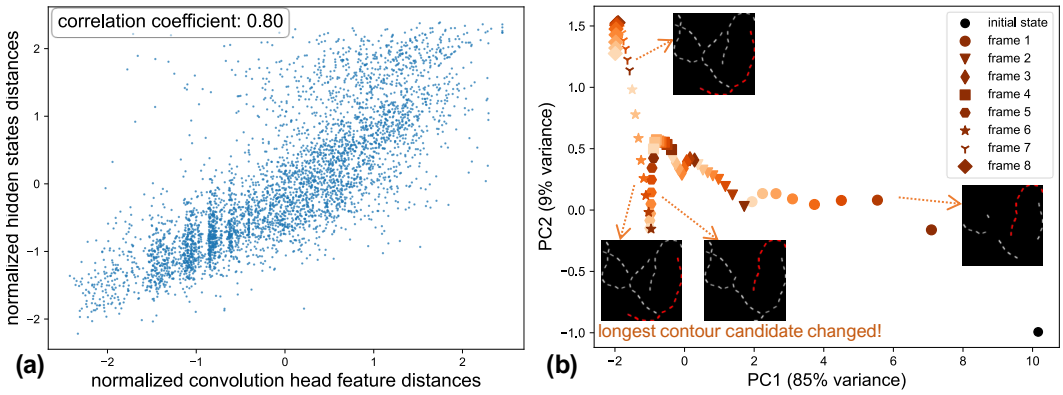


Figure 4: **(a)** There exists a strong correlation (correlation coefficient=0.8) for $d(h_t^{N(t)}, h_{t+1}^{N(t+1)})$ and $d(x_t, x_{t+1})$. The model used is RecSlowFast with hGRU-s8f8 and the dataset is the test split of *T-Pathfinder-Hard*. **(b)** A visualization for an exemplary hidden states trajectory of the model used for (a). Symbols with the same shape represent hidden states extracted from the same frame. A lighter color indicates a further recurrent step in that frame. The PCA reduction details are in Section A.4 and the full example is shown in Section A.6 Figure 8.

The limit of recurrence with the same weights When we increased the number of recurrent steps for RecSlowFast with hGRU on *T-Pathfinder-Hard* from s8f8 (mIoU: 0.79), we observed that model

performance did not continue to improve with s10f10 (mIoU: 0.77) or s12f12 (mIoU: 0.77). A potential solution to enhance expressiveness could involve incorporating multiple recurrent blocks, each equipped with a different set of parameters and its own variable recurrent step schedule.

5 Distance-based recurrence halting criteria

Inspired by the empirical findings in Figure 4, we design distance-based recurrence halting criteria that do not impose a fixed inference schedule and can dynamically choose the executed recurrent steps without any auxiliary neural network for scheduling. In Algorithm 1 the pseudo code is given. In each recurrent step, we calculate the distance $d_{\text{curr}} = d(h_t^n, h_t^{n-1})$ between the current and previous hidden states, and compare it with the previous distance $d_{\text{prev}} = d(h_t^{n-1}, h_t^{n-2})$ (proper initial value and indices are assumed here). The comparison is conducted by calculating the distance ratio $d_{\text{curr}}/(d_{\text{prev}}+\epsilon)$, where ϵ is a small value used for preventing zero division. A thresholding hyperparameter Δ is selected, so that if the distance ratio is below this threshold, we regard the recurrence as already converged, thus exiting it. Conversely, if the distance ratio surpasses Δ, the model is likely adapting to new input so another recurrent step will be carried out until reaching the preset maximum number of steps N_{max}. We set N_{max} to 12 which is $1.5\times$ of the original training steps, meaning we allow the model to extrapolate outside of its training regime if it is not converging within the training steps. This is similar to [5, 33].

Algorithm 1: RecSlowFast inference with distance-based early halting of recurrence

Input : Video frames $I_1, I_2, \ldots, I_T$, initial state h_{init}
Output : Predictions $y_1^{\text{pred}}, y_2^{\text{pred}}, \ldots, y_T^{\text{pred}}$

```
h_1^0 ← h_init;                          %initialize hidden state
for t ← 1 to T do
    n ← 0, d_prev ← 0, x_t ← f_conv1(I_t);
    while n ≤ N_max do
        n ← n + 1;
        h_t^n ← r_θ(h_t^{n-1}, x_t);     %one recurrent step
        d_curr ← d(h_t^n, h_t^{n-1});
        if d_curr/(d_prev + ε) < Δ then
            break;                       %early halting
        else
            d_prev ← d_curr;             %update distance
    y_t^pred ← f_conv2(h_t^n);
```

Experiment results For the results in Table 2, the model being studied for different inference strategies is the RecSlowFast-hGRU-s8f8 trained on *T-Pathfinder-Hard*. We studied two distance metrics, ℓ_2 and ℓ_1. Threshold values from 0.8 to 1.1 with step 0.1 are being examined, and the average number of recurrent steps spent on the videos is recorded. Notice that **no** finetuning was conducted. For both distance metrics ℓ_2 and ℓ_1, the average number of steps is effectively reduced, when Δ is increasing. Our distance based halting method enables controlling the performance-speed trade-off with one single threshold hyparameter, which is a desired property. If we compare distance based hallting whose averaged steps is similar to naively running same number of recurrent steps on every frames, it provides a better task accuracy. This indicates that distance based halting works better in allocating the total computing budget.

inference strategy		Δ	mIoU	averaged steps
distance based halting	ℓ_2	0.8	.78	6.7
		0.9	.77	5.7
		1.0	.76	4.3
		1.1	.74	3.3
	ℓ_1	0.8	.79	9.5
		0.9	.78	6.3
		1.0	.76	3.6
		1.1	.73	2.8
baseline	s8f8		.79	8.0
	s6f6		.73	6.0
	s4f4		.71	4.0
	s3f3		.67	3.0

Table 2: Distance based halting with different thresholds

6 Video semantic segmentation

To verify the RecSlowFast framework on larger-scale task,
we carried out preliminary experiments with the semantic segmentation task on the CamVid dataset [6]. We use the commonly studied CamVid version first processed in the SegNet work [1]. Each frame is of resolution 360×480 pixels. There are 11 semantic classes and one unlabeled class. 367, 101 and 233 frames are included in the train, validation, and test split respectively. The dataset is collected by

a camera on a driving car and the frame rate 1 Hz, thus two consecutive frames can be quite different with a driving speed and a low camera frame rate. We train and test on consecutive frame pairs instead of entire video sequences.

We use a dual-gated recurrent unit (DRU) from [38] with U-Net [31] type of skip connections, to implement the RecSlowFast framework. The DRU network consists of an encoder with 4 downsampling convolutional layers and a decoder with 4 upsampling convolutional layers. The DRU recurrent unit is placed in the middle as a recurrent structure with memory. The output of the entire pipeline is also fed back as the input. Details on the DRU network, training, and dataset are in Section A.5.

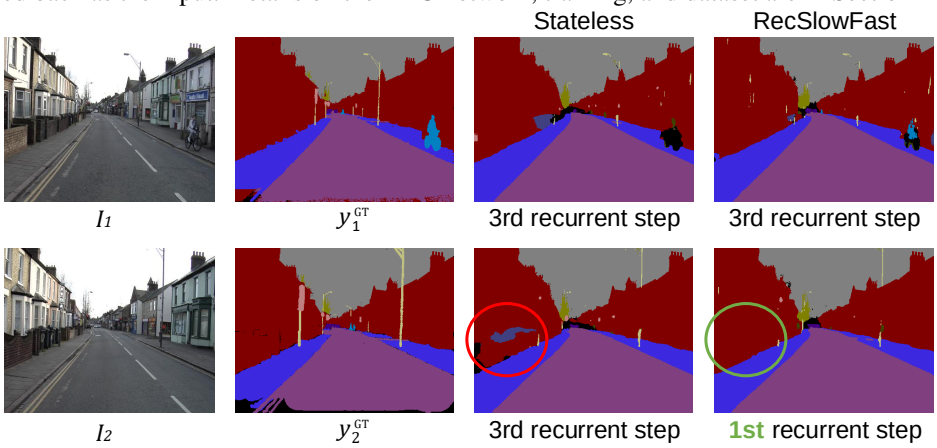


Figure 5: Results of the RecSlowFast-DRU on CamVid semantic segmentation. The leftmost column is the input image pair and the second column contains their ground truths. The 3rd and 4th columns are predictions from Stateless-DRU-s3f3 and RecSlowFast-DRU-s3f1 respectively.

When comparing RecSlowFast-DRU with 3 recurrent steps, the test mIoU is better than with 1 recurrent step (0.63 vs. 0.6). When we further finetune the 3-step checkpoint with s3f1 schedule, the averaged mIoU became 0.62 while saving 33% computation cost compare to s3f3 on the test pair. We also show qualitatively in Figure 5, that RecSlowFast-DRU is able to out perform the Stateless version with just 1 additional recurrent step on the second input frame $\mathcal{I}_2$ and successfully solved ambiguous area in $\mathcal{I}_2$ (marked with green circle) while Stateless DRU failed with 3 recurrent steps (marked with red circle).

7 Conclusion

In this work, we propose the RecSlowFast framework to show the usefulness of exploiting hidden representation similarities from temporally correlated input frames with recurrence for accelerating temporal visual processing. We constructed a new dataset *T-Pathfinder*, which requires tracking of the longest contour in a sequence of frames. The number of recurrent steps could be reduced for later frames in the sequence, thus cutting down the computation cost, with the help of representation similarities between two consecutive frames. Moreover, such cross-input hidden states reuse enables the model to continuously improve the prediction and boost task accuracy by a large margin on the *T-Pathfinder* dataset. We also analyze the correlation between input feature distances and hidden states distances, providing empirical support for the feasibility of saving computation on temporally correlated inputs. Based on this correlation, we propose early halting criteria based on the hidden state distance ratio, which enables dynamic scheduling without presetting a fixed inference schedule or sophisticated auxiliary scheduling networks. RecSlowFast also maintains high parameter efficiency via the layer reuse property of recurrence. We perceive the potential for implementing RecSlowFast on hardware with constrained resources. In such contexts, the reuse of parameters through a recurrent structure becomes particularly advantageous.

Acknowledgments and Disclosure of Funding

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 899287.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. doi: 10.1109/TPAMI.2016.2644615. URL https://ieeexplore.ieee.org/document/7803544.

[2] S. Bai, Z. Geng, Y. Savani, and J. Kolter. Deep equilibrium optical flow estimation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 610–620, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society. doi: 10.1109/CVPR52688.2022.00070. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.00070.

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/01386bd6d8e091c2ab4c7c7de644d37b-Paper.pdf.

[4] Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to Ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021. URL https://openreview.net/forum?id=1EuxRTe0WN.

[5] Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Extrapolation without overthinking. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20232–20242. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/7f70331dbe58ad59d83941dfa7d975aa-Paper-Conference.pdf.

[6] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *European Conference on Computer Vision (ECCV)*, pp. 44–57, 2008. URL https://link.springer.com/chapter/10.1007/978-3-540-88682-2_5.

[7] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[9] Can Ufuk Ertenli, Emre Akbaş, and Ramazan Gökberk Cinbiş. Streaming multiscale deep equilibrium models. 17th European Conference on Computer Vision, ECCV 2022, 2022. URL https://www.ecva.net/papers/eccv_2022/papers_ECCV/papers/136710189-supp.pdf.

[10] C. Feichtenhofer, H. Fan, J. Malik, and K. He. SlowFast networks for video recognition. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6201–6210, Los Alamitos, CA, USA, nov 2019. IEEE Computer Society. doi: 10.1109/ICCV.2019.00630. URL https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00630.

[11] Alex Graves. Adaptive computation time for recurrent neural networks, 2017. URL https://arxiv.org/abs/1603.08983.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, pp. 770–778. IEEE, June 2016. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459.

[13] Roos Houtkamp and Pieter R. Roelfsema. Parallel and serial grouping of image elements in visual perception. *Journal of Experimental Psychology: Human Perception and Performance*, 36:1443–1459, 12 2010. ISSN 00961523. doi: 10.1037/A0020248. URL https://www.researchgate.net/publication/45819659_Parallel_and_Serial_Grouping_of_Image_Elements_in_Visual_Perception.

[14] Kohitij Kar and James J. DiCarlo. Fast recurrent processing via ventrolateral prefrontal cortex is needed by the primate ventral stream for robust core visual object recognition. *Neuron*, 109(1):164–176.e5, 2021. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2020.09.035. URL https://www.sciencedirect.com/science/article/pii/S0896627320307595.

[15] Kohitij Kar, Jonas Kubilius, Kailyn Schmidt, Elias B. Issa, and James J. DiCarlo. Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. *Nature Neuroscience 2019 22:6*, 22:974–983, 4 2019. ISSN 1546-1726. doi: 10.1038/s41593-019-0392-5. URL https://www.nature.com/articles/s41593-019-0392-5.

[16] Tim C. Kietzmann, Courtney J. Spoerer, Lynn K.A. Sörensen, Radoslaw M. Cichy, Olaf Hauk, and Nikolaus Kriegeskorte. Recurrence is required to capture the representational dynamics of the human visual system. *Proceedings of the National Academy of Sciences of the United States of America*, 116:21854–21863, 10 2019. ISSN 10916490. doi: 10.1073/PNAS.1905544116/SUPPL_FILE/PNAS.1905544116.SM06.MP4. URL https://www.pnas.org/doi/abs/10.1073/pnas.1905544116.

[17] Junkyung Kim*, Drew Linsley*, Kalpit Thakkar, and Thomas Serre. Disentangling neural mechanisms for perceptual grouping. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJxrVA4FDS.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[19] Jonas Kubilius, Martin Schrimpf, Aran Nayebi, Daniel Bear, Daniel L. K. Yamins, and James J. DiCarlo. CORnet: Modeling the neural mechanisms of core object recognition. *bioRxiv*, 09/2018 2018. doi: https://doi.org/10.1101/408385. URL https://www.biorxiv.org/content/10.1101/408385v1.full.pdf.

[20] Jonas Kubilius, Martin Schrimpf, Ha Hong, Najib J. Majaj, Rishi Rajalingham, Elias B. Issa, Kohitij Kar, Pouya Bashivan, Jonathan Prescott-Roy, Kailyn Schmidt, Aran Nayebi, Daniel Bear, Daniel L. K. Yamins, and James J. DiCarlo. Brain-Like Object Recognition with High-Performing Shallow Recurrent ANNs. In *Neural Information Processing Systems (NeurIPS)*, pp. 12785–12796. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9441-brain-like-object-recognition-with-high-performing-shallow-recurrent-anns.

[21] Victor A.F. Lamme and Pieter R. Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in Neurosciences*, 23:571–579, 11 2000. ISSN 01662236. doi: 10.1016/S0166-2236(00)01657-X. URL https://pubmed.ncbi.nlm.nih.gov/11074267/.

[22] Sam Leroux, Pavlo Molchanov, Pieter Simoens, Bart Dhoedt, Thomas Breuel, and Jan Kautz. IamNN: Iterative and adaptive mobile neural network for efficient image classification. In *International Conference on Learning Representations Workshop*, 2018. URL https://openreview.net/forum?id=BkG3_ykDz.

[23] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020. doi: 10.1109/TPAMI.2018.2858826. URL https://ieeexplore.ieee.org/document/8417976.

[24] Drew Linsley, Junkyung Kim, Vijay Veerabadran, Charles Windolf, and Thomas Serre. Learning long-range spatial dependencies with horizontal gated recurrent units. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/ec8956637a99787bd197eacd77acce5e-Paper.pdf.

[25] Drew Linsley, Alekh Karkada Ashok, Lakshmi Narasimhan Govindarajan, Rex Liu, and Thomas Serre. Stable and expressive recurrent vision models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 10456–10467. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/766d856ef1a6b02f93d894415e6bfa0e-Paper.pdf.

[26] Mason Liu, Menglong Zhu, Marie White, Yinxiao Li, and Dmitry Kalenichenko. Looking Fast and Slow: Memory-guided mobile video object detection, 2019. URL https://arxiv.org/abs/1903.10172.

[27] P. Micaelli, A. Vahdat, H. Yin, J. Kautz, and P. Molchanov. Recurrence without recurrence: Stable video landmark detection with deep equilibrium models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22814–22825, Los Alamitos, CA, USA, jun 2023. IEEE Computer Society. doi: 10.1109/CVPR52729.2023.02185. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.02185.

[28] Aran Nayebi, Javier Sagastuy-Brena, Daniel M. Bear, Kohitij Kar, Jonas Kubilius, Surya Ganguli, David Sussillo, James J. DiCarlo, and Daniel L. K. Yamins. Recurrent Connections in the Primate Ventral Visual Stream Mediate a Trade-Off Between Task Performance and Network Size During Core Object Recognition. *Neural Computation*, 34(8):1652–1675, 07 2022. ISSN 0899-7667. doi: 10.1162/neco_a_01506. URL https://doi.org/10.1162/neco_a_01506.

[29] Randall O'Reilly, Dean Wyatte, Seth Herd, Brian Mingus, and David Jilk. Recurrent processing during object recognition. *Frontiers in Psychology*, 4, 2013. ISSN 1664-1078. doi: 10.3389/fpsyg.2013.00124. URL https://www.frontiersin.org/articles/10.3389/fpsyg.2013.00124.

[30] Pieter R. Roelfsema. Cortical algorithms for perceptual grouping. *Annual review of neuroscience*, 29: 203–227, 2006. ISSN 0147-006X. doi: 10.1146/ANNUREV.NEURO.29.051605.112939. URL https://pubmed.ncbi.nlm.nih.gov/16776584/.

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pp. 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28. URL https://doi.org/10.1007/978-3-319-24574-4_28.

[32] Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. Consistent accelerated inference via confident adaptive transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 4962–4979, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.406. URL https://aclanthology.org/2021.emnlp-main.406.

[33] Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Tsp2PL7-GQ.

[34] Junxing Shi, Haiguang Wen, Yizhen Zhang, Kuan Han, and Zhongming Liu. Deep recurrent neural network reveals a hierarchy of process memory during dynamic natural vision. *Human Brain Mapping*, 39, 02 2018. doi: 10.1002/hbm.24006. URL https://pubmed.ncbi.nlm.nih.gov/29436055/.

[35] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-Chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf.

[36] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/215a71a12769b056c3c32e7299f1c5ed-Paper.pdf.

[37] Hanlin Tang, Martin Schrimpf, William Lotter, Charlotte Moerman, Ana Paredes, Josue Ortega Caro, Walter Hardesty, David Cox, and Gabriel Kreiman. Recurrent computations for visual pattern completion. *Proceedings of the National Academy of Sciences of the United States of America*, 115:8835–8840, 8 2018. ISSN 10916490. doi: 10.1073/PNAS.1719397115/-/DCSUPPLEMENTAL. URL www.pnas.org/cgi/doi/10.1073/pnas.1719397115.

[38] Wei Wang, Kaicheng Yu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Recurrent U-Net for resource-constrained segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2142–2151, 2019. URL https://www.computer.org/csdl/proceedings-article/iccv/2019/480300c142/1hQqyqNAQsU.

[39] Y. Xu, T. Fu, H. Yang, and C. Lee. Dynamic video segmentation network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6556–6565, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society. doi: 10.1109/CVPR.2018.00686. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00686.

[40] J. Yang, Y. Bhalgat, S. Chang, F. Porikli, and N. Kwak. Dynamic iterative refinement for efficient 3d hand pose estimation. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2703–2713, Los Alamitos, CA, USA, jan 2022. IEEE Computer Society. doi: 10.1109/WACV51458.2022.00276. URL https://doi.ieeecomputersociety.org/10.1109/WACV51458.2022.00276.

[41] Aston Zhang, Yi Tay, Yikang Shen, Alvin Chan, and Shuai Zhang. Self-instantiated recurrent units with dynamic soft recursion. In *Advances in Neural Information Processing Systems*, volume 34, pp. 6503–6514. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/3341f6f048384ec73a7ba2e77d2db48b-Paper.pdf.

[42] M. Zhu and M. Liu. Mobile video object detection with temporally-aware feature maps. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5686–5695, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society. doi: 10.1109/CVPR.2018.00596. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00596.

[43] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. Flow-guided feature aggregation for video object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 408–417, Los Alamitos, CA, USA, oct 2017. IEEE Computer Society. doi: 10.1109/ICCV.2017.52. URL https://doi.ieeecomputersociety.org/10.1109/ICCV.2017.52.