The streaming rollout of deep networks

Volker Fischer, Jan Köhler, Thomas Pfeil

Bosch Center for Artificial Intelligence

Network rollouts are used for training and inference of recurrent networks. Different network rollouts exist not only for recurrent but any neural network. Different rollouts lead to different behavior and level of model-parallelism for the same network.

We introduce a formal description of network rollouts and their degree of model-parallelism. We prove that the streaming rollout maximizes model-parallelism. We empirically compare different rollouts w.r.t. their behavior and model-parallelism. We present an experimental open-source toolbox to investigate the streaming rollout.

Graph Formulation for Network Rollouts

Definition (network): A **network** is a directed graph $N = (V, E)$ with a set of vertices V, a non-empty set of edges E, and every node is connected to the input I_N:

$$v \in V$$
$$e = (e_{\text{src}}, e_{\text{tgt}}) \in E \subset V \times V$$
$$I_N \coloneqq \{v \in V \mid \nexists u \in V : (u, v) \in E\}$$

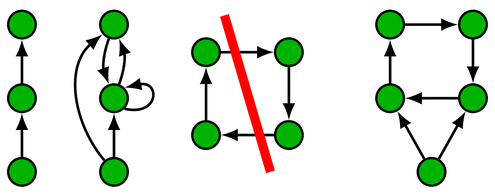

Figure 1: Network examples. According to our definition, the third graph is not a network, because is has no input nodes.

Definition (rollout pattern & window): A **rollout pattern** of a network $N = (V, E)$ is a mapping $R : E \to \{0, 1\}$. The **rollout window** of size $W \in \mathbb{N}$ for R is the directed graph $R_W = (V_W, E_W)$, with:

$$V_W \coloneqq \{0, \dots, W\} \times V, \quad \overline{v} = (i, v) \in V_W$$
$$E_W \coloneqq \{((i, u), (j, v)) \in V_W \times V_W \mid (u, v) \in E \ \wedge \ j = i + R((u, v))\} \tag{1}$$

A rollout pattern is **valid** iff R_W is acyclic. $\mathcal{R}_N$ is the set of all valid rollout patterns and the rollout pattern $R \equiv 1$ is the **streaming rollout** $R^{\text{stream}} \in \mathcal{R}_N$. Two rollout patterns R and R' are **equally model-parallel** iff $R(e) = R'(e)$ for all edges $e = (u, v)$, with $u \notin I_N$.

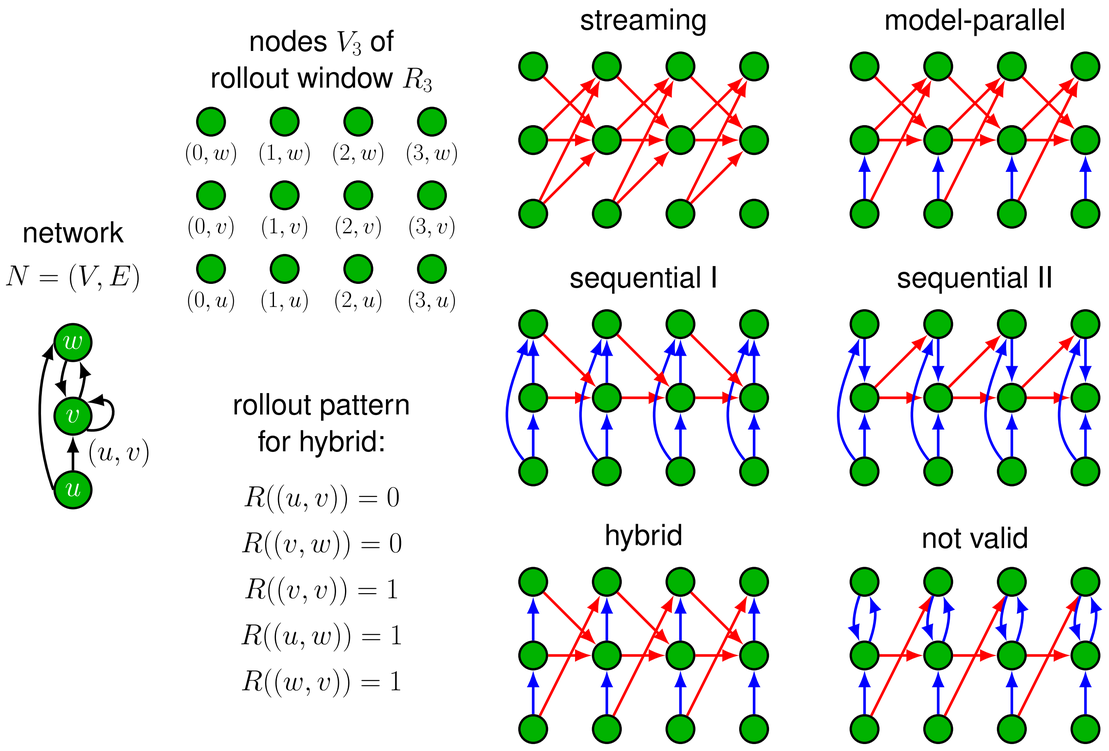

Figure 2: Six out of the 32 rollouts (right) of the network on the left.

Definition (update state): A **state** S of a rollout window $R_W = (V_W, E_W)$ is a mapping $S : V_W \to \{0, 1\}$. Σ_W denotes the set of all states. The **full state** S_{full} and **initial state** S_{init} are defined as:

$$S_{\text{full}} \equiv 1; \qquad S_{\text{init}}((i, v)) = 1 \iff v \in I_N \vee i = 0. \tag{2}$$

Figure 3: The full state S_{full} (left) and initial state S_{init} (right) for some rollout window R_3. The number inside a node (i, v) is the state $S((i, v))$ of that node.

Definition (update step): Given a state S, the **update step** function U yields an updated state $U(S)$:

$$U : \Sigma_W \to \Sigma_W; \qquad U(S) : V_W \to \{0, 1\} \tag{3}$$

$$U(S)(\overline{v}) \coloneqq \begin{cases} 1 \text{ if } S(\overline{v}) = 1 \text{ or if for all } (\overline{u}, \overline{v}) \in E_W \ : \ S(\overline{u}) = 1 \\ 0 \text{ otherwise} \end{cases}$$

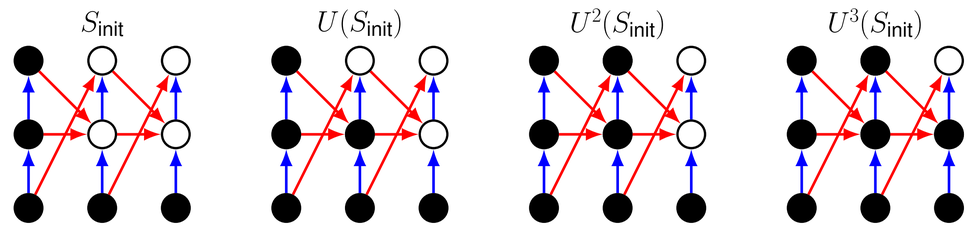

Figure 4: The update step U is successively applied to the initial state S three times.

Definition (inference tableau & factor): The mapping $T : V_W \to \mathbb{N}$ is the **inference tableau**:

$$T(\overline{v}) \ \coloneqq \ \max_{p \in P_{\overline{v}}} |p| \ = \ \operatorname*{argmin}_{n \in \mathbb{N}} \{U^n(S_{\text{init}})(\overline{v}) = 1\} \tag{4}$$

where $P_{\overline{v}}$ contains all paths in R_W to $\overline{v}$ and with $p(1)_{\text{tgt}} \notin \{0\} \times V$. The maximal value of T over the rollout window of size 1 is the rollout pattern's **inference factor**:

$$F(R) \coloneqq \max_{\overline{v} \in V_1} T_{R_1}(\overline{v}). \tag{5}$$

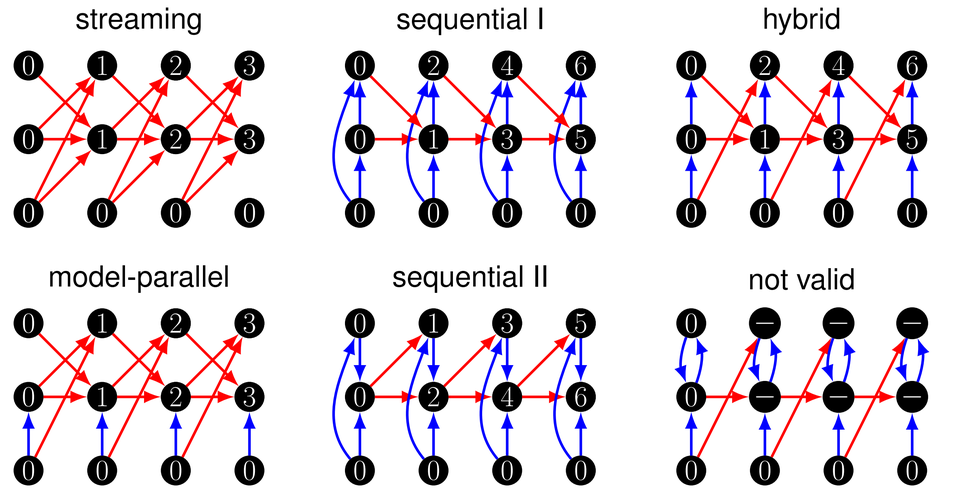

Figure 5: Tableau values $T_R(\overline{v})$ for the nodes of the six rollouts depicted in Fig. 2.

- Ignoring edges inside the 0-th frame, rollout windows of window size W have the same number of edges $W * |E|$, independent of the chosen R.
- Maximal path lengths in rollout windows differ between rollout patterns.

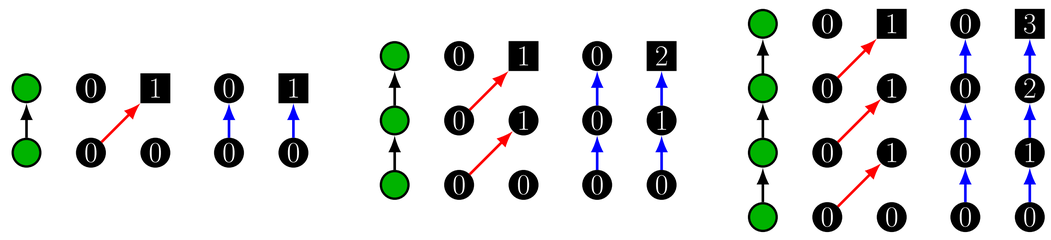

Figure 6: Illustration of inference factor $F(R)$. For three simple feed-forward networks with increasing depth, the rollout windows of size $W = 1$ are shown for the streaming (left) and sequential (right) rollouts.

Lemma 1: $|\mathcal{R}_N|$ is bounded by:

$$1 \leq n \leq |\mathcal{R}_N| \leq 2^{|E| - |E_{\text{rec}}|}, \tag{6}$$

E_{rec} being the set of all self-connecting edges $E_{\text{rec}} \coloneqq \{(u, v) \in E \mid u = v\}$, and n either:

- $n = 2^{|E_{\text{forward}}|}$, with E_{forward} the set of edges not contained in any cycle, or
- $n = \prod_{p \in C} (2^{|p|} - 1)$, $C \subset C_N$ any set of minimal, pair-wise disjunct cycles.

$\implies$ Number of valid rollouts increases with network complexity.

- Towards fully model-parallel execution

Theorem 1: Let R be a valid rollout pattern for a network $N = (V, E)$, then the following statements are equivalent:

a) R and R^{stream} are equally model-parallel.

b) The first frame is updated after the first update step: $F(R) = 1$.

c) For $W \in \mathbb{N}$, the i-th frame of R_W is updated at the i-th update step:

$$\forall (i, v) \in V_W \ : \ T((i, v)) \leq i.$$

d) Inference tableau values of R_W are minimal everywhere and over all rollout patterns:

$$\forall \overline{v} \in V_W \ : \ T_{R_W}(\overline{v}) = \min_{R' \in \mathcal{R}_N} T_{R'_W}(\overline{v}).$$

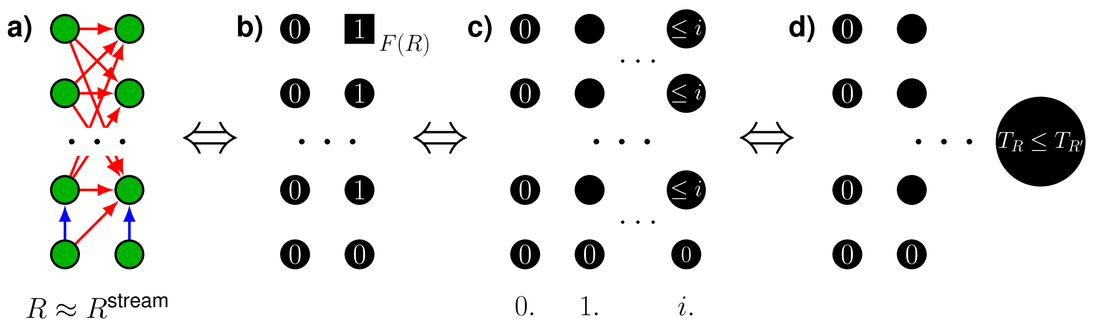

Figure 7: Illustration of the four equivalent statements from Theorem 1.

Network Training & Experiments

Training loss is computed as mean over single-frame classification losses.

The **number of input / loss frames** is independent from the rollout pattern.

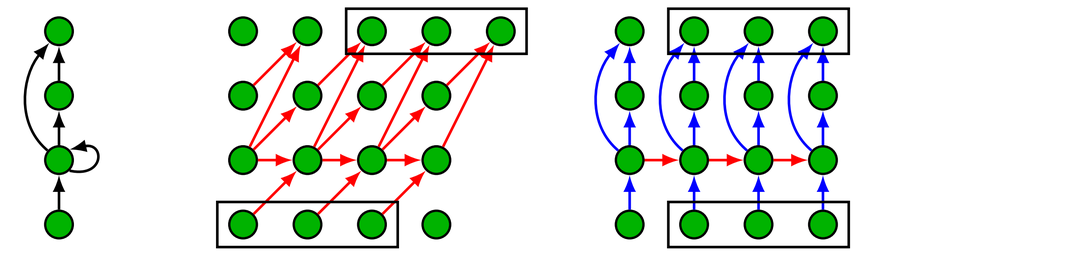

Figure 8: Illustration of network training for streaming and sequential rollout pattern.

Rollouts (streaming vs. sequential) were compared on a **response time task**.

Network input is a sequence of images on which networks have to respond as quickly as possible with the correct class (cf. Fig. 9).

Time is measured in number of update steps U, assuming inference with stateful rollout windows of size $W = 1$.

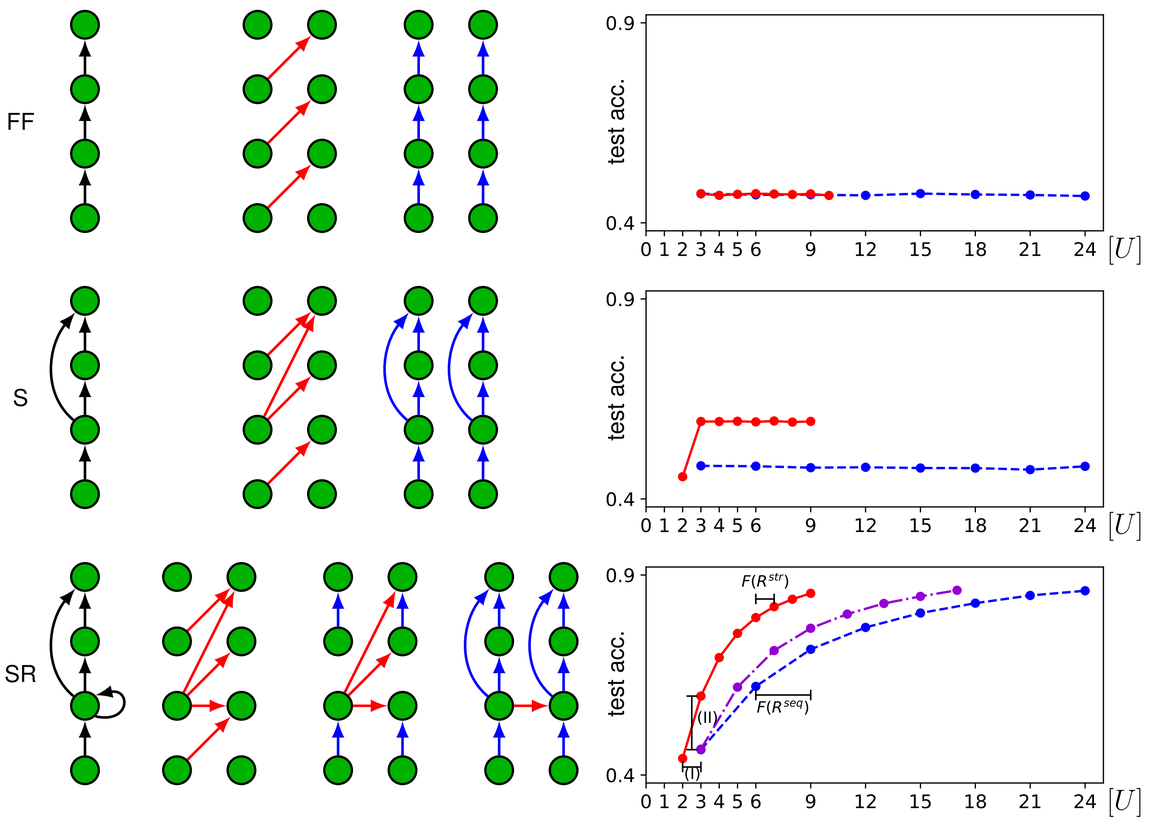

Figure 9: Experimental results for MNIST. For three networks (first column, one network per row: FF, S, SR), the streaming and sequential, and for the SR network, additionally one of its hybrid rollout patterns (middle column) are shown. On the right side, the test accuracy over the number of update steps U is shown for each rollout pattern.

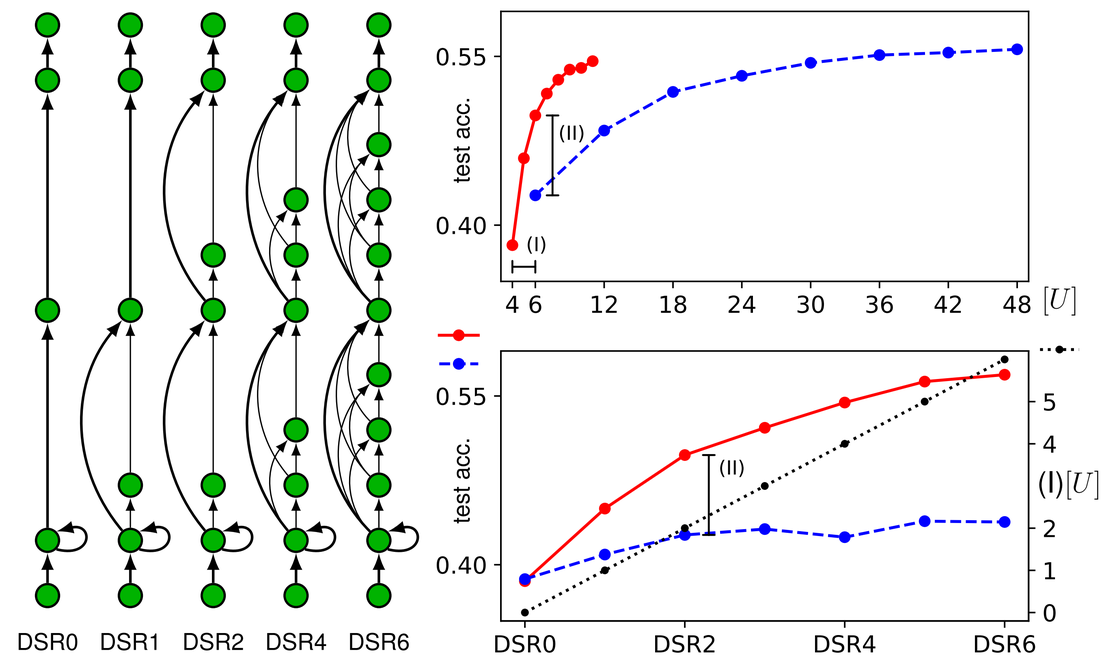

Figure 10: For seven networks (DSR0 - DSR6, shown left), streaming and sequential rollout patterns were compared on the CIFAR10 dataset. For the DSR2 network, test accuracy over update steps is shown in the top right corner. In the bottom right corner, the according test accuracies (streaming in red, sequential in blue) at time of first response for the sequential rollout (cf. (II) top right) are shown for all seven models. Further, the black plot shows the delay between first responses of streaming and sequential rollouts (cf. (I) top right).

Discussion & Conclusion

The presented **theory** for rollouts is **generically applicable** to a vast variety of deep neural networks.

Restriction of rollout patterns $R(e) \in \{0, 1\}$ generalizes to $R(e) > 1$ via copy-nodes. For $R(e) < 0$, backward connections use future information and, hence loose real-time capability.

Different rollout patterns may lead to **different behavior**, hence parameters between rollout patterns might be incompatible and training has to consider the rollout pattern.

For **inference** with streaming rollouts, we use **stateful shallow rollouts** (e.g., $W = 1$), enabling infinite memory for recurrent networks and minimizing memory footprint.

We **assumed equal update time** for every single node update. In general, inference of a single frame for streaming rollouts takes the runtime of the most expensive node update, inference for sequential rollouts takes the summed up runtime of all node updates. The streaming rollout favors networks with many nodes of approximately equal update times.

Software & hardware have to support model-parallelism in order to leverage advantages of the streaming rollout.

Synchronization of layers is important for the streaming rollout and mechanisms to control information flow is subject of future research.

In a Nutshell

For one network, several rollout patterns exist.

Different rollout patterns yield different

- **levels of model-parallelism and**
- **network behavior.**

The streaming rollout

- **is maximally model-parallel and**
- **all nodes can be updated in parallel.**

Scripts to reproduce experimental results, as well as an experimental toolbox to study the streaming rollout pattern are available as open-source code under:

https://github.com/boschresearch/statestream