Simplified Reversible Residual Networks (Simplified RevNets)

Erland Brandser Olsson

November 2026

Backpropagation - Standard AI Training

Nearly all neural networks are trained with backpropagation (a.k.a. backprop) [1]. Backprop consists of a forward pass,

$$\boldsymbol{x}^{(t+1)} = F(\boldsymbol{x}^{(t)}), \tag{1}$$

and a backward pass,

$$\begin{aligned} \overline{\boldsymbol{w}}^{(t)} &= \left(\frac{\partial F(\boldsymbol{x}^{(t)})}{\partial \boldsymbol{w}^{(t)}}\right)^{T} \overline{\boldsymbol{x}}^{(t+1)} \\ \overline{\boldsymbol{x}}^{(t)} &= \left(\frac{\partial F(\boldsymbol{x}^{(t)})}{\partial \boldsymbol{x}^{(t)}}\right)^{T} \overline{\boldsymbol{x}}^{(t+1)}, \end{aligned} \tag{2}$$

where $\overline{\boldsymbol{x}}^{(t)} = \partial \mathcal{L}/\partial \boldsymbol{x}^{(t)}$ and $\overline{\boldsymbol{w}}^{(t)} = \partial \mathcal{L}/\partial \boldsymbol{w}^{(t)}$ are the gradients of the loss with respect to the input and weights of layer t. Backprop is efficient—it reuses $\overline{\boldsymbol{x}}^{(t+1)}$ layer by layer—but **backprop must store every input activation** to form the required Jacobians: memory, not computation, is the bottleneck.

Residual Networks (ResNets)-Key Architectural Innovation

ResNets [2] are a key innovation enabling deeper networks: they give activations and gradients an explicit identity path that bypasses the layer,

$$\text{ResNet Forward:} \quad \boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + F(\boldsymbol{x}^{(t)}) \qquad \text{ResNet Backward:} \quad \overline{\boldsymbol{x}}^{(t)} = \left(\boldsymbol{I} + \left(J_F^{(t+1)}\right)^T\right) \overline{\boldsymbol{x}}^{(t+1)}, \tag{3}$$

where $J_F^{(t+1)} = \partial F(\boldsymbol{x}^{(t)})/\partial \boldsymbol{x}^{(t)}$. The update is purely additive, so the upstream gradient $\overline{\boldsymbol{x}}^{(t+1)}$ flows through the identity path without being multiplied through J_F (with $F = 0$ it passes unchanged). But deeper networks mean more activations to store, making **GPU memory a critical bottleneck**. ResNets simplified the weighted shortcut connection used in Highway Networks [3],

$$\text{Highway Forward:} \quad \boldsymbol{x}^{(t+1)} = \gamma \boldsymbol{x}^{(t)} + (1-\gamma) F(\boldsymbol{x}^{(t)}) \qquad \text{Highway Backward:} \quad \overline{\boldsymbol{x}}^{(t)} = \left(\gamma \boldsymbol{I} + (1-\gamma)\left(J_F^{(t+1)}\right)^T\right) \overline{\boldsymbol{x}}^{(t+1)}, \tag{4}$$

where $\gamma \in [0, 1]$. A follow-up analysis [4] shows that ResNet's success stems from dropping these weights: unrolled over L layers, the γ factors accumulate as $\propto \gamma^L$ and $\propto (1-\gamma)^L$, causing vanishing or exploding gradients. That is why ResNet is now the default building block, e.g., in Transformers, where F alternates between attention and feed-forward

Reversible Residual Networks (RevNets) - Recomputing Inputs

Key idea: ResNet's forward pass is mostly a series of additions, so we can run it in reverse—each subtraction undoes an addition, recovering activations without storing them. However, a ResNet layer $\boldsymbol{y} = \boldsymbol{x} + F(\boldsymbol{x})$ is not invertible: $\boldsymbol{x}$ cannot be recovered from $\boldsymbol{y}$ alone. RevNet [5] solves this with a two-step update,

$$\begin{aligned} \boldsymbol{y}_1 &= \boldsymbol{x}_1 + F(\boldsymbol{x}_2) \\ \boldsymbol{y}_2 &= \boldsymbol{x}_2 + G(\boldsymbol{y}_1), \end{aligned} \tag{5}$$

so that during the backward pass we can recover the inputs by subtraction,

$$\begin{aligned} \boldsymbol{x}_2 &= \boldsymbol{y}_2 - G(\boldsymbol{y}_1) \\ \boldsymbol{x}_1 &= \boldsymbol{y}_1 - F(\boldsymbol{x}_2). \end{aligned} \tag{6}$$

Major practical limitation: the input $\boldsymbol{x}$ must be split into $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and the single ResNet function replaced by a two-step F/G coupling, which lacks strong information-theoretic justification and obscures ResNet's original design goals. RevNet is therefore not a drop-in replacement: adopting it requires a major architectural redesign, typically based on heuristic design choices.

Simplified RevNets (SRevNets)

Goal: make a ResNet layer $\boldsymbol{y} = \boldsymbol{x} + F(\boldsymbol{x})$ reversible without modifying F.

$$\begin{array}{ll} \text{SRevNet v1 Forward:} & \text{SRevNet v2 Forward:} \\ \boldsymbol{y}_a = \boldsymbol{x}_a + F(\boldsymbol{x}) & \boldsymbol{y}_a = \boldsymbol{x}_a + \boldsymbol{x} \\ \boldsymbol{y} = \boldsymbol{x} + \boldsymbol{y}_a & \boldsymbol{y} = \boldsymbol{x} + G(\boldsymbol{y}_a) \end{array} \tag{7}$$

SRevNet v1 sets $G = I$ (the identity) and v2 sets $F = I$, and we introduce an auxiliary accumulator variable $\boldsymbol{x}_a$ to avoid input splitting so that we can rewrite $\boldsymbol{x}_a = \boldsymbol{x}_1$ and $\boldsymbol{x} = \boldsymbol{x}_2$ in RevNet's formulation. Inverse maps are

$$\begin{array}{ll} \text{SRevNet v1 Reverse:} & \text{SRevNet v2 Reverse:} \\ \boldsymbol{x} = \boldsymbol{y} - \boldsymbol{y}_a & \boldsymbol{x} = \boldsymbol{y} - G(\boldsymbol{y}_a) \\ \boldsymbol{x}_a = \boldsymbol{y}_a - F(\boldsymbol{x}) & \boldsymbol{x}_a = \boldsymbol{y}_a - \boldsymbol{x} \end{array} \tag{8}$$

Both are standard ResNet layers—module (F in v1, G in v2) **unmodified**:

$$\text{v1:} \quad \boldsymbol{y} = \boldsymbol{x} + \widetilde{F}(\boldsymbol{x}), \qquad \widetilde{F}(\boldsymbol{x}) = F(\boldsymbol{x}) + \boldsymbol{x}_a \tag{9}$$

$$\text{v2:} \quad \boldsymbol{y} = \boldsymbol{x} + \widetilde{G}(\boldsymbol{x}), \qquad \widetilde{G}(\boldsymbol{x}) = G(\boldsymbol{x} + \boldsymbol{x}_a) \tag{10}$$

v2 keeps the ResNet paradigm output = input + transform most directly. Therefore, SRevNets are **drop-in replacements** for ResNets.

SRevNet v1 ($G = I$) vs v2 ($F = I$)

$\boldsymbol{x}_a$ and its gradient $\overline{\boldsymbol{x}}_a$ are initialized to zero at the start of the forward and backward passes, so we can express $\boldsymbol{x}_a$ as $\boldsymbol{x}$-terms unrolled at layer t. The forward pass updates at layer t are

SRevNet v1 $\boldsymbol{x}_a$ update:

$$\begin{aligned} \boldsymbol{y}_a &= \boldsymbol{x}_a + F(\boldsymbol{x}) \Rightarrow \boldsymbol{y}_a = \textstyle\sum_{i=1}^{t} F(\boldsymbol{x}^{(i)}) \\ \boldsymbol{y} &= \boldsymbol{x} + \boldsymbol{y}_a \end{aligned}$$

SRevNet v2 $\boldsymbol{x}_a$ update:

$$\begin{aligned} \boldsymbol{y}_a &= \boldsymbol{x}_a + \boldsymbol{x} \Rightarrow \boldsymbol{y}_a = \textstyle\sum_{i=1}^{t} \boldsymbol{x}^{(i)} \\ \boldsymbol{y} &= \boldsymbol{x} + G(\boldsymbol{y}_a). \end{aligned} \tag{11}$$

v1 accumulates residuals, while v2 accumulates input features, creating a dense connectivity pattern analogous to DenseNet [6].

SRevNet v1 ($G = I$) vs v2 ($F = I$)

The classifier only uses the last-layer $\boldsymbol{x}$, not the auxiliary $\boldsymbol{x}_a$, so we can rewrite $\boldsymbol{x}_a$ as $\boldsymbol{x}$-terms in the backward pass. Input gradients at layer t are

$$\text{v1:} \quad \overline{\boldsymbol{x}}^{(t)} = \overline{\boldsymbol{x}}^{(t+1)} + \left(J_F^{(t+1)}\right)^T \left(\sum_{i=t}^{L} \overline{\boldsymbol{x}}^{(i+1)}\right), \tag{12}$$

$$\text{v2:} \quad \overline{\boldsymbol{x}}^{(t)} = \overline{\boldsymbol{x}}^{(t+1)} + \sum_{i=t}^{L} \left(J_G^{(i+1)}\right)^T \overline{\boldsymbol{x}}^{(i+1)}, \tag{13}$$

where $J_F^{(t+1)} = \partial F(\boldsymbol{x}^{(t)})/\partial \boldsymbol{x}^{(t)}$ and $J_G^{(i+1)} = \partial G(\boldsymbol{x}_a^{(i+1)})/\partial \boldsymbol{x}_a^{(i+1)}$. Both versions achieve the ResNet effect $\overline{\boldsymbol{x}}^{(t)} = \overline{\boldsymbol{x}}^{(t+1)} + \left(J_F^{(t+1)}\right)^T \overline{\boldsymbol{x}}^{(t+1)}$, but only v2 enables a **parallel ultra-highway effect** in which each layer t is connected to all deeper layers $i > t$. The same idea is used in DenseNet,

$$\begin{array}{cc} \text{DenseNet Forward:} & \text{DenseNet Backward:} \\ \boldsymbol{x}^{(t+1)} = F\left(\text{Concat}([\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}])\right) & \overline{\boldsymbol{x}}^{(t)} = \sum_{i=t}^{L} \left(J_x^{(i+1)}\right)^T \overline{\boldsymbol{x}}^{(i+1)}, \end{array} \tag{14}$$

where $J_x^{(i+1)} = \partial \boldsymbol{x}^{(i+1)}/\partial \boldsymbol{x}^{(t)}$. v2 achieves this using $\boldsymbol{x}_a$ summation rather than concatenation, reducing memory from $\mathcal{O}(t)$ to $\mathcal{O}(1)$. ResNet bridges only adjacent layers; DenseNet connects every layer to all deeper ones. v1 does not enable the same dense connectivity; $J_F^{(t+1)}$ is placed outside of the sum, whereas in v2 $J_G^{(i+1)}$ is placed inside, creating an asymmetry in v1 that distorts signals from deeper layers.

Simplified RevNet (SRevNet) vs Momentum ResNet (MResNet)

MResNet [7] is a neural-ODE-inspired second-order lift of ResNet, iterating

$$\boldsymbol{v}^{(t+1)} = \gamma \boldsymbol{v}^{(t)} + (1-\gamma) F(\boldsymbol{x}^{(t)}), \quad \boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \boldsymbol{v}^{(t+1)}, \tag{15}$$

and is reversible. Yet it aims to model richer second-order ODE dynamics, not merely to fix RevNet—a specialized ODE model. Its branching is similar to SRevNet v1 ($G = I$, asymmetric Jacobian scaling), but adds momentum $\gamma \in [0, 1]$. This momentum creates nasty depth-dependent multiplicative weighting, a direct parallel to the ResNet-vs-Highway debate: the input gradient

$$\overline{\boldsymbol{x}}^{(t)} = \overline{\boldsymbol{x}}^{(t+1)} + (1-\gamma)\left(J_F^{(t+1)}\right)^T \sum_{i=t}^{L} \gamma^{i-t}\, \overline{\boldsymbol{x}}^{(i+1)} \tag{16}$$

is γ-weighted with depth, unlike v1/v2's clean, unscaled additive form—creating vanishing/exploding gradients in deep networks. Reversal also loses information at each γ multiplication/division (reconstruction errors); the authors' fix needs an extra high-precision bit buffer—very expensive in fp16 mixed-precision training.

Performance

Table 1 Accuracy on ImageNet with ResNet101

Method	Val Top-1 (acc.)
ResNet	78.2
RevNet	76.7
MResNet $\gamma = 0.1$	0.1
$-\ \gamma = 0.25$	0.1
$-\ \gamma = 0.5$	0.1
$-\ \gamma = 0.75$	77.2
$-\ \gamma = 0.9$	77.3
$-\ \gamma = 0.95$	76.9
$-\ \gamma = 0.99$	76.0
SRevNet v1	77.0
SRevNet v2	77.3

Table 2 Perplexity on OpenWebText with Llama

Method	Val ppl (val.)
Llama	16.4
RevNet	16.5
MResNet $\gamma = 0.1$	921.3
$-\ \gamma = 0.25$	168.9
$-\ \gamma = 0.5$	128.4
$-\ \gamma = 0.75$	51.5
$-\ \gamma = 0.9$	16.8
$-\ \gamma = 0.95$	17.0
$-\ \gamma = 0.99$	18.3
SRevNet v1	17.5
SRevNet v2	16.8

Reversible networks match standard ResNet performance. SRevNet vs MResNet is a direct parallel to ResNet vs Highway: γ makes MResNet a family of models, not one. SRevNet matches the best γ via another branching ($F = I$), with no failed γ runs: same lesson from the ResNet follow-up: ResNet $\approx$ best-tuned Highway variant, with no failed γ runs.

Memory Usage and Training Time

Table 3 Train stats from ImageNet runs.

Method	Memory (GiB)	Network (MiB)	FLOPs (TFLOPs)	Train time (hours)
ResNet	32.4	42.5	23.9	20.3
RevNet	21.6	43.3	30.5	27.8
MResNet	18.6	42.5	30.3	29.6
MResNet γ-alg.	33.2	42.5	30.3	-
SRevNet v1	18.3	42.5	30.3	26.5
SRevNet v2	18.3	42.5	30.3	26.4

Table 4 Train stats from Llama runs.

Method	Memory (GiB)	Network (GiB)	FLOPs (TFLOPs)	Train time (hours)
Llama	64.8	1.1	63.3	17.2
RevNet	17.4	1.3	82.3	21.8
MResNet	17.0	1.1	82.3	22.7
SRevNet v1	17.0	1.1	82.3	22.3
SRevNet v2	17.0	1.1	82.3	22.2

Comparison with Activation Checkpointing

Table 5 GPU memory usage at different depths.

Layers	100	200	300	400
ResNet	4.9 GiB	9.7 GiB	14.5 GiB	19.3 GiB
ResNet AC	2.6 GiB	5.0 GiB	7.4 GiB	9.8 GiB
SRevNet v1	0.3 GiB	0.3 GiB	0.3 GiB	0.3 GiB
SRevNet v2	0.3 GiB	0.3 GiB	0.3 GiB	0.3 GiB
Network	0.9 MiB	1.8 MiB	2.7 MiB	3.6 MiB

Table 6 GFLOPs at various depths, backward pass (first number) and forward pass (second number).

Layers	100	200	300	400
ResNet	739.9 (369.9)	1479.8 (739.9)	2219.6 (1109.8)	2959.5 (1479.8)
ResNet AC	1109.8 (369.9)	2219.6 (739.9)	3329.4 (1109.8)	4439.3 (1479.8)
SRevNet v1	1109.8 (369.9)	2219.6 (739.9)	3329.4 (1109.8)	4439.3 (1479.8)
SRevNet v2	1109.8 (369.9)	2219.6 (739.9)	3329.4 (1109.8)	4439.3 (1479.8)

Summary and Future Work

- SRevNets are reversible **drop-in replacements** for ResNets.
- SRevNet v2 generalizes ResNet from adjacent-layer to dense connectivity (DenseNet-style).
- Reversible networks are a natural extension of activation checkpointing: free memory reduction at the same training time—we expect broader adoption in the future.
- Future work: reversible linear self-attention via an IO-aware SRevNet formula.

[1] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[2] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[3] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks". In: *arXiv preprint arXiv:1505.00387* (2015).

[4] Kaiming He et al. "Identity mappings in deep residual networks". In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

[5] Aidan N Gomez et al. "The Reversible Residual Network: Backpropagation Without Storing Activations". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https:

//proceedings.neurips.cc/paper_files/paper/2017/file/f9be311e65d81a9ad8150a60844bb94c-Paper.pdf.

[6] Gao Huang et al. "Densely connected convolutional networks". In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*. Ieee. 2017, pp. 2261–2269.

[7] Michael E Sander et al. "Momentum residual neural networks". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9276–9287.