The Reversible Residual Network

Backpropagation without storing activations

Aidan N. Gomez*,[1] Mengye Ren*,[1,2,3] Raquel Urtasun[1,2,3] Roger B. Grosse[1,2]

[1] Department of Computer Science, University of Toronto, [2] Vector Institute for Artificial Intelligence, [3] Uber Advanced Technologies Group

* - Equal contribution

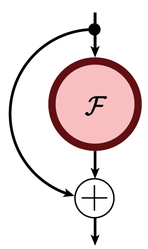
Motivation

- Residual neural networks have rapidly pushed forward the state-of-the-art in multiple domains.
- Task performance continually increases as networks grow.
- Memory consumption is the primary bottleneck
- We present Reversible Residual Networks (RevNets) as a variant of ResNets which overcome the activation storage problem.
- RevNets have constant activation storage with no loss in performance.

Background

Residual Layers:

$$y = x + \mathcal{F}(x)$$



- Residual layers are a standard component of neural networks.
- They appear in state-of-the-art architectures in vision, natural language and audio tasks.
- They resist the problem of vanishing gradients.

Backpropagation:

- Nearly all modern neural nets are trained using **backprop**.
- Backprop requires storing activations in memory, leading to cost proportional to the number of layers in a network.
- In convolutional neural networks the bulk of memory is dedicated to activation storage, while parameter storage is relatively cheap.

Method

The reversible residual unit takes the following form:

$$y_1 = x_1 + \mathcal{F}(x_2)$$
$$y_2 = x_2 + \mathcal{G}(y_1)$$

And can be reversed exactly:

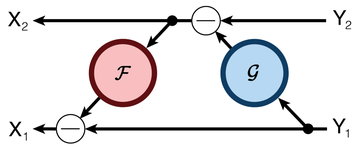
$$x_2 = y_2 - \mathcal{G}(y_1)$$
$$x_1 = y_1 - \mathcal{F}(x_2)$$

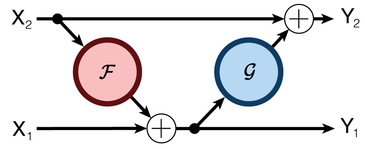
Method (cont.)

We can reconstruct activations during backprop, like so:

Algorithm 1 Reversible Residual Block Backprop

function BLOCKREVERSE($(y_1, y_2), (\overline{y}_1, \overline{y}_2)$)
  $z_1 \leftarrow y_1$
  $x_2 \leftarrow y_2 - \mathcal{G}(z_1)$
  $x_1 \leftarrow z_1 - \mathcal{F}(x_2)$
  $\overline{z}_1 \leftarrow \overline{y}_1 + \left(\frac{\partial \mathcal{G}}{\partial z_1}\right)^\top \overline{y}_2$ ▷ ordinary backprop
  $\overline{x}_2 \leftarrow \overline{y}_2 + \left(\frac{\partial \mathcal{F}}{\partial x_2}\right)^\top \overline{z}_1$ ▷ ordinary backprop
  $\overline{x}_1 \leftarrow \overline{z}_1$
  $\overline{w}_\mathcal{F} \leftarrow \left(\frac{\partial \mathcal{F}}{\partial w_\mathcal{F}}\right)^\top \overline{z}_1$ ▷ ordinary backprop
  $\overline{w}_\mathcal{G} \leftarrow \left(\frac{\partial \mathcal{G}}{\partial w_\mathcal{G}}\right)^\top \overline{y}_2$ ▷ ordinary backprop
  return (x_1, x_2) and $(\overline{x}_1, \overline{x}_2)$ and $(\overline{w}_\mathcal{F}, \overline{w}_\mathcal{G})$
end function





- In general, for a network with N layers: the forward pass requires ~N add-multiply ops while the backward requires ~2N.
- For RevNet, the residual functions each must be recomputed. Therefore, the number of operations required for reversible backprop is approximately 4N , or roughly 33% more than ordinary backprop.

Technique	Spatial Complexity (Activations)	Computational Complexity
Naive	$\mathcal{O}(L)$	$\mathcal{O}(L)$
Checkpointing [22]	$\mathcal{O}(\sqrt{L})$	$\mathcal{O}(L)$
Recursive Checkpointing [5]	$\mathcal{O}(\log L)$	$\mathcal{O}(L \log L)$
Reversible Networks (Ours)	$\mathcal{O}(1)$	$\mathcal{O}(L)$

Our method has **constant** space complexity and linear computational time.

Task	Parameter Cost	Activation Cost
ResNet-101	$\sim$ 178MB	$\sim$ 5250MB
RevNet-104	$\sim$ 180MB	$\sim$ 1440MB

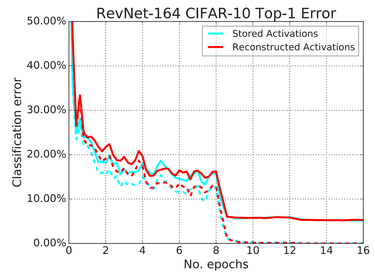
activations dominate storage requirements for a standard architecture; RevNets save 72.6%.

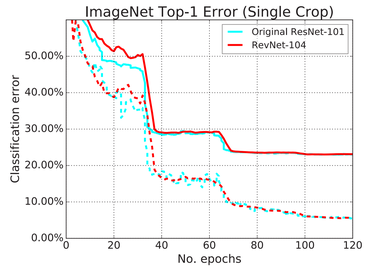
Results

- Our RevNet architectures are chosen to roughly match computational depth and number of parameters.
- RevNets matched the performance of ResNets, while saving >70% memory requirements.
- As shown in figures below: no per-iteration slowdown was observed in RevNets.
- RevNets allow for 4x larger batch size on ImageNet.
- We were able to fit a 600+ layer RevNet on one GPU vs. 100 layer for ResNet.

Architecture	CIFAR-10 [17]		CIFAR-100 [17]	
	ResNet	RevNet	ResNet	RevNet
32 (38)	**7.14%**	7.24%	29.95%	**28.96%**
110	**5.74%**	5.76%	26.44%	**25.40%**
164	5.24%	**5.17%**	**23.37%**	23.69%

ImageNet	
ResNet-101	RevNet-104
23.01%	23.10%





Reconstruction Error:

- Reconstructing activations introduces error in floating point.
- Extremely deep networks may suffer from the accumulation of these errors.
- Figure right: we measure the empirical error in our deepest network and observe that it remains extremely small throughout training.
- Despite numerical error, training efficiency and task performance are nearly indistinguishable.