Reducing the Computational Cost of Deep Generative Models with Binary Neural Networks

Computer Science
UCL

Thomas Bird, Friso H. Kingma, David Barber

Motivation

- Generative models have high compute requirements
- Research into ML compute efficiency has instead focused on supervised
- Can we apply these techniques to generative models?

Binary neural networks

- Reduce space
- Increase speed
- Train through the straight-through estimator + clipping:

	Weights	Activations
Forward pass:	$w_{\mathbb{B}} = \text{sign}(w_{\mathbb{R}})$	$\alpha_{\mathbb{B}} = \text{sign}(\alpha_{\mathbb{R}})$
Backward pass:	$\frac{\partial \mathcal{L}}{\partial w_{\mathbb{R}}} := \frac{\partial \mathcal{L}}{\partial w_{\mathbb{B}}}$	$\frac{\partial \mathcal{L}}{\partial \alpha_{\mathbb{R}}} := \frac{\partial \mathcal{L}}{\partial \alpha_{\mathbb{B}}} * 1_{\lvert \alpha_{\mathbb{R}} \rvert \leq 1}$
After update:	$w_{\mathbb{R}} \leftarrow \max(-1, \min(1, w_{\mathbb{R}}))$	–

Binary weight norm

- Normalisation is important for BNNs, since weights are large in magnitude.
- Prefer to use weight norm over batch norm for generative models.
- Weight norm admits a simplification with binary weights, we call *binary weight norm:*

Weight norm:

$$\mathbf{w}_{\mathbb{R}} = \mathbf{v}_{\mathbb{R}} \cdot \frac{g}{||\mathbf{v}_{\mathbb{R}}||}$$

Binary weight norm:

$$\mathbf{w}_{\mathbb{R}} = \mathbf{v}_{\mathbb{B}} \cdot \frac{g}{\sqrt{n}}$$

$$\mathcal{F}(\mathbf{x}, \mathbf{v}_{\mathbb{B}} \cdot \alpha) = \mathcal{F}(\mathbf{x}, \mathbf{v}_{\mathbb{B}}) \cdot \alpha$$

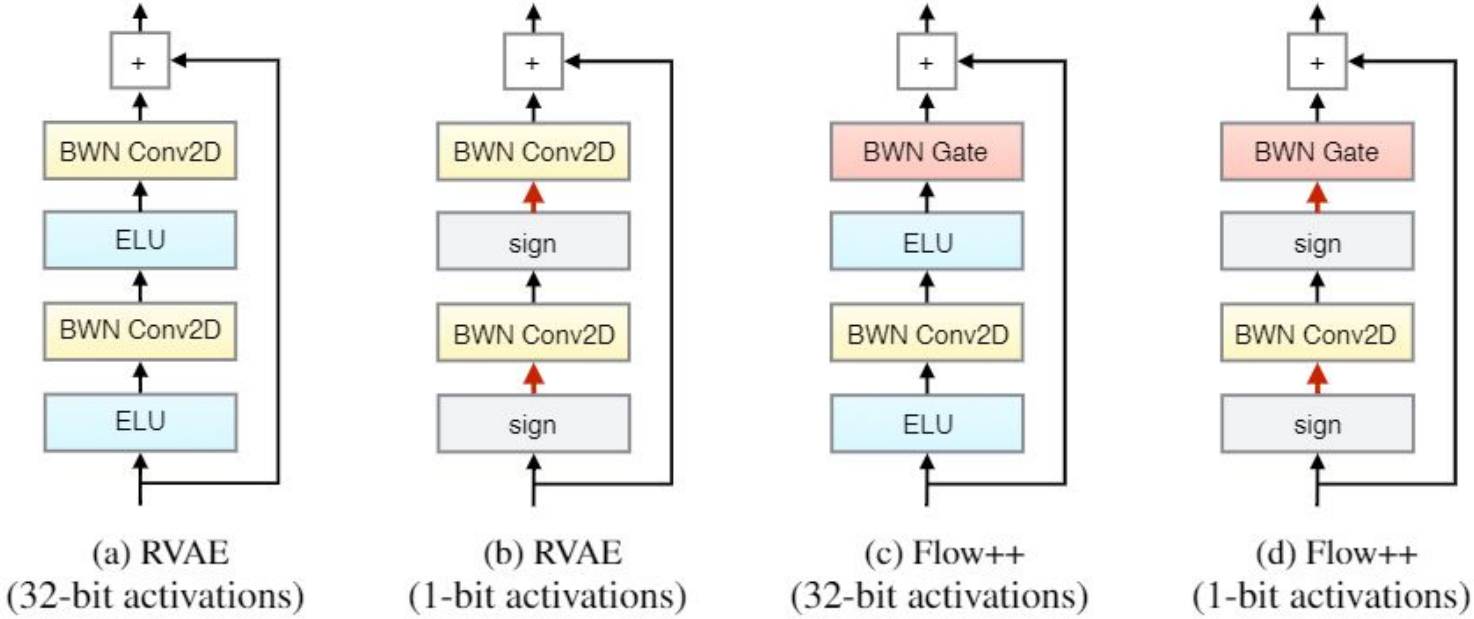
Commutes with linear ops
-> still use fast binary operations

Binarising residual layers

- Which parts of generative models should we binarise?
- Binarising full model doesn’t work well.
- Residual layers are natural candidates, since they are motivated by avoiding the degradation problem, which is a primary concern for binary layers.
- Residual layers also usually contain most network parameters.

$$\mathbf{g}_{\text{res}}(\mathbf{x}) = \mathcal{T}(\mathbf{x}) + \mathbf{x}$$

Binarised residual blocks



(a) RVAE (32-bit activations)

(b) RVAE (1-bit activations)

(c) Flow++ (32-bit activations)

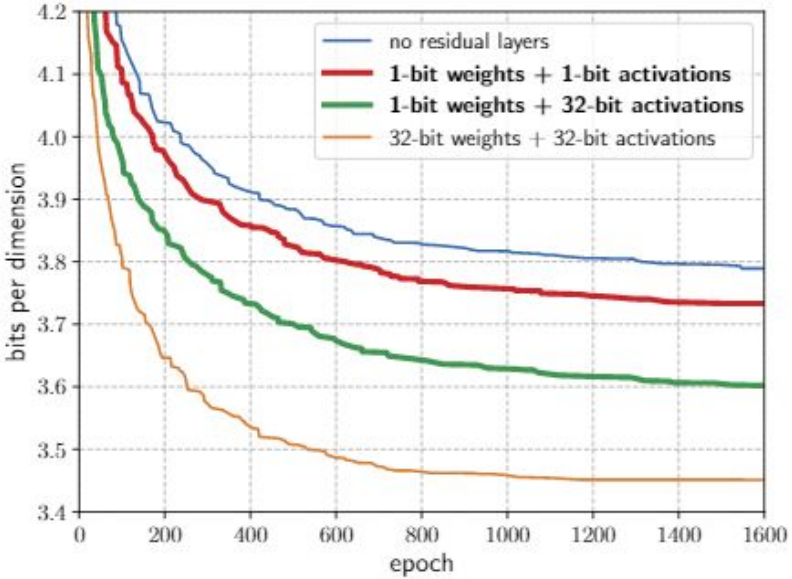
(d) Flow++ (1-bit activations)

Results

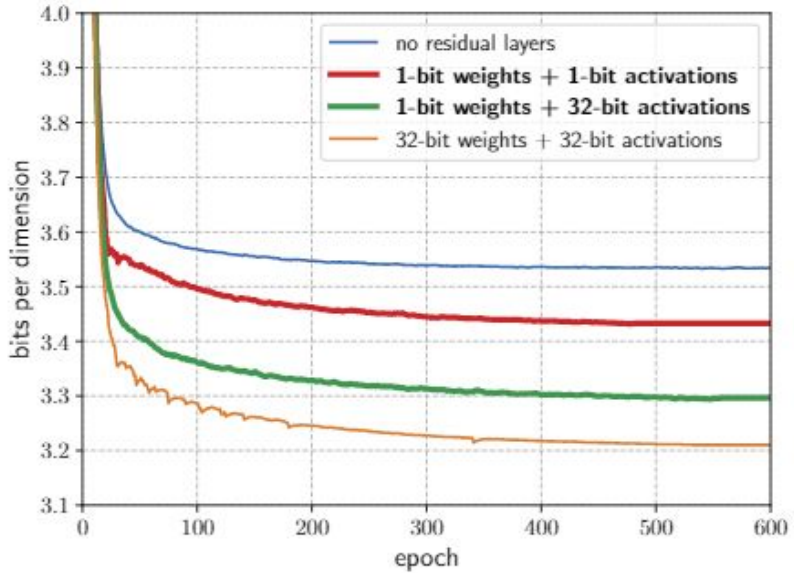
Model size made **90-94%** smaller, with only a small sacrifice in performance.

	Precision		Modelling loss		# Parameters	% Binary	Memory cost
	Weights	Activations	CIFAR	ImageNet (32×32)			
ResNet VAE	32-bit	32-bit	3.45	4.25	56M	0%	255 MB
	1-bit	32-bit	3.60	4.47	56M	97.1%	13 MB
	1-bit	1-bit	3.73	4.58	56M	97.1%	13 MB
increased width	1-bit	32-bit	3.56	-	96M	97.7%	20 MB
	1-bit	1-bit	3.68	-	96M	97.7%	20 MB
no residual	N.A.	N.A.	3.78	-	1.6M	0%	6 MB
Flow++	32-bit	32-bit	3.21	4.05	34M	0%	129 MB
	1-bit	32-bit	3.29	4.18	34M	90.1%	14 MB
	1-bit	1-bit	3.43	4.30	34M	90.1%	14 MB
no residual	N.A.	N.A.	3.54	-	2.2M	0%	9 MB

Training curves

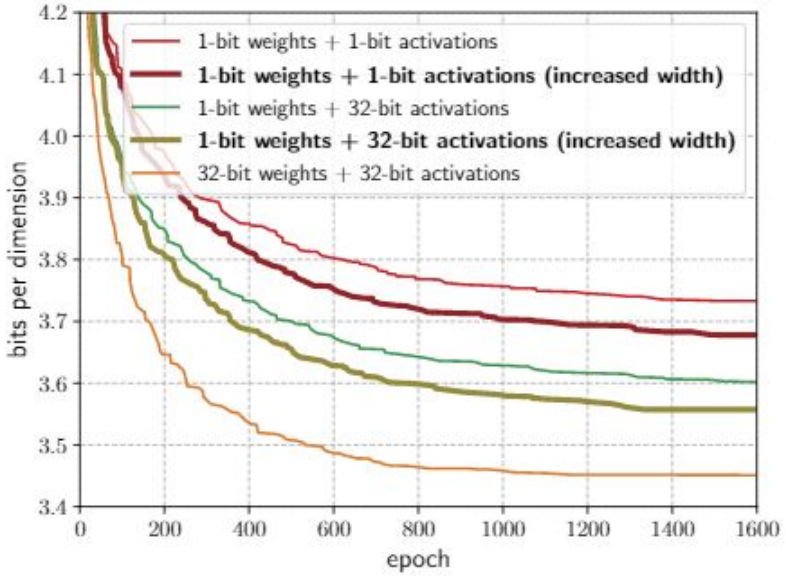


(a) **ResNet VAE**

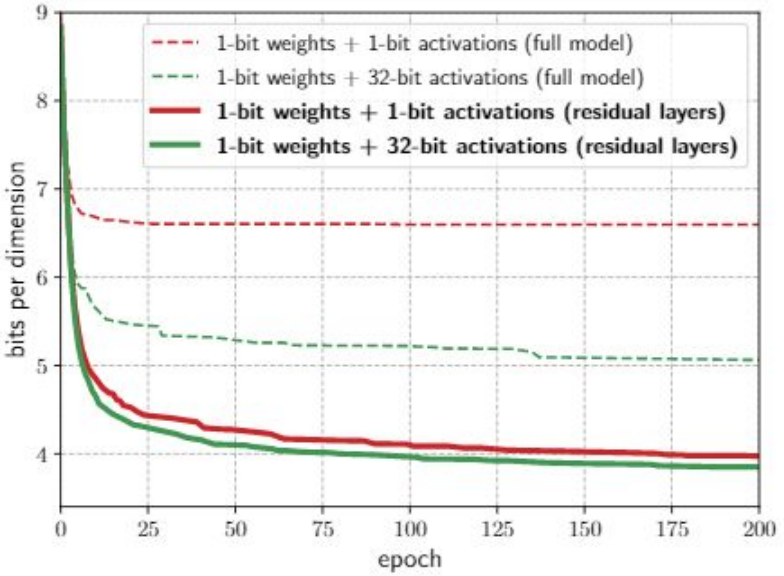


(b) **Flow++**

Ablations



(c) **ResNet VAE** (increased channel width)



(d) **ResNet VAE** (ablations)