

# Invertible Convolutional Networks

Marc Finzi\* Pavel Izmailov\* Wesley Maddox\* Polina Kirichenko\* Andrew Gordon Wilson  
Cornell University

## Overview

- Directly invert ordinary convolutional networks by adding circular padding, bijective activations, and invertible downsampling.
- Use lifting into a higher dimensional space to compute inverses and log determinants via real valued linear algebra.
- We apply these methods to define a simple, fully convolutional normalizing flow.

## Inverse of Convolutional Layers

Convolutions can be expressed in the Fourier domain as

$$\text{Conv}_W(x)_i = \sum_j W_{ij} \star x_j = \sum_j \mathcal{F}^{-1}(\mathcal{F}(W_{ij})^* \circ \mathcal{F}(x_j)).$$

In the Fourier basis, the (linear) Conv operation is block diagonal, where the blocks extend over channels but not spatially.

$$\begin{pmatrix} c, h, w \\ \begin{matrix} A_1 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & A_{hw} \end{matrix} \\ c, h, w \end{pmatrix}^{-1} = \begin{pmatrix} c, h, w \\ \begin{matrix} A_1^{-1} & 0 & 0 & 0 \\ 0 & A_2^{-1} & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & A_{hw}^{-1} \end{matrix} \\ c, h, w \end{pmatrix}$$

**Figure 1:** The overall  $chw \times chw$  linear operation performed by  $(X_j)_{j=1}^c \rightarrow (\sum_j ((\mathcal{F}(W)^*)_{ij} \circ X_j))_{i=1}^c$  is block diagonal, with  $h, w$  many blocks of size  $c \times c$ .

$$\text{Conv}_W^{-1}(y)_k = \sum_i \mathcal{F}^{-1} \left( (\mathcal{F}(W)^*)_{ki}^{-1} \circ \mathcal{F}(y_i) \right) \quad (1)$$

- This representation reduces an intractable naive  $O(h^3w^3c^3)$  computation time to  $O(c^2hw \log(hw) + c^3hw)$ .
- The existence of the inverse  $(\mathcal{F}(W)^*)_{ki}^{-1}$  is guaranteed since at initialization the set of singular matrices has measure 0.

## Fast Inverses and Logdets

Log determinants can also be efficiently computed, in the Fourier domain the computation decomposes into the sum over spatial locations,

$$\log |\det(\text{Conv}_W)| = \sum_{h,w} \log |\det(\mathcal{F}(W)_{:, :, h,w}^*)|. \quad (2)$$

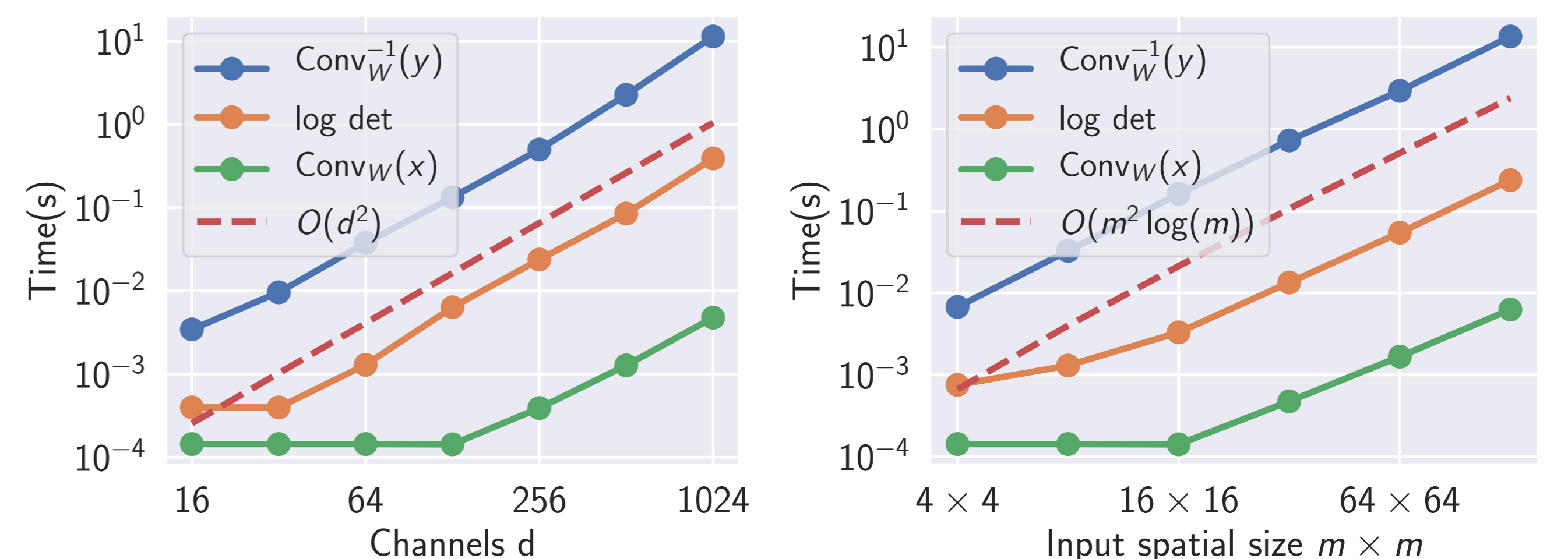
- Both (1) and (2) require linear algebra routines for the collection of complex matrices  $\mathcal{F}(W)_{:, :, h,w}^*$ .
- Many deep learning frameworks (PyTorch, MxNet, Chainer, Theano) do not support complex linear algebra.
- We compute these log determinants and inverses using only real valued linear algebra routines by lifting to a higher dimensional space.

Consider the injective ring homomorphism,  $\phi: \mathbb{C}^{n \times n} \rightarrow \mathbb{R}^{2n \times 2n}$ :

$$\phi(A + iB) = I \otimes A + J \otimes B = \begin{pmatrix} A & -B \\ B & A \end{pmatrix}, \quad J = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad J^2 = -I.$$

Given a pair of complex matrices  $C, V \in \mathbb{C}^{n \times n}$ ,  $\phi(C)\phi(V) = \phi(CV)$ . Additionally, it can be shown that

$$\log |\det(C)| = \frac{1}{2} \log |\det(\phi(C))|$$



**Figure 2:** Runtime of Conv operations for  $bs, c, h, w = 32 \times 64 \times 8 \times 8$  sized image with a  $3 \times 3$  filter as the number of channels  $d = c$  and the spatial size  $h, w = m$ ,  $m$  is varied. Log-determinant and  $\text{Conv}_W(\cdot)$  operations are accelerated on an Nvidia 1080ti, and the  $\text{Conv}_W^{-1}(\cdot)$  is computed on the CPU.

## Smooth Bijective Activation Function

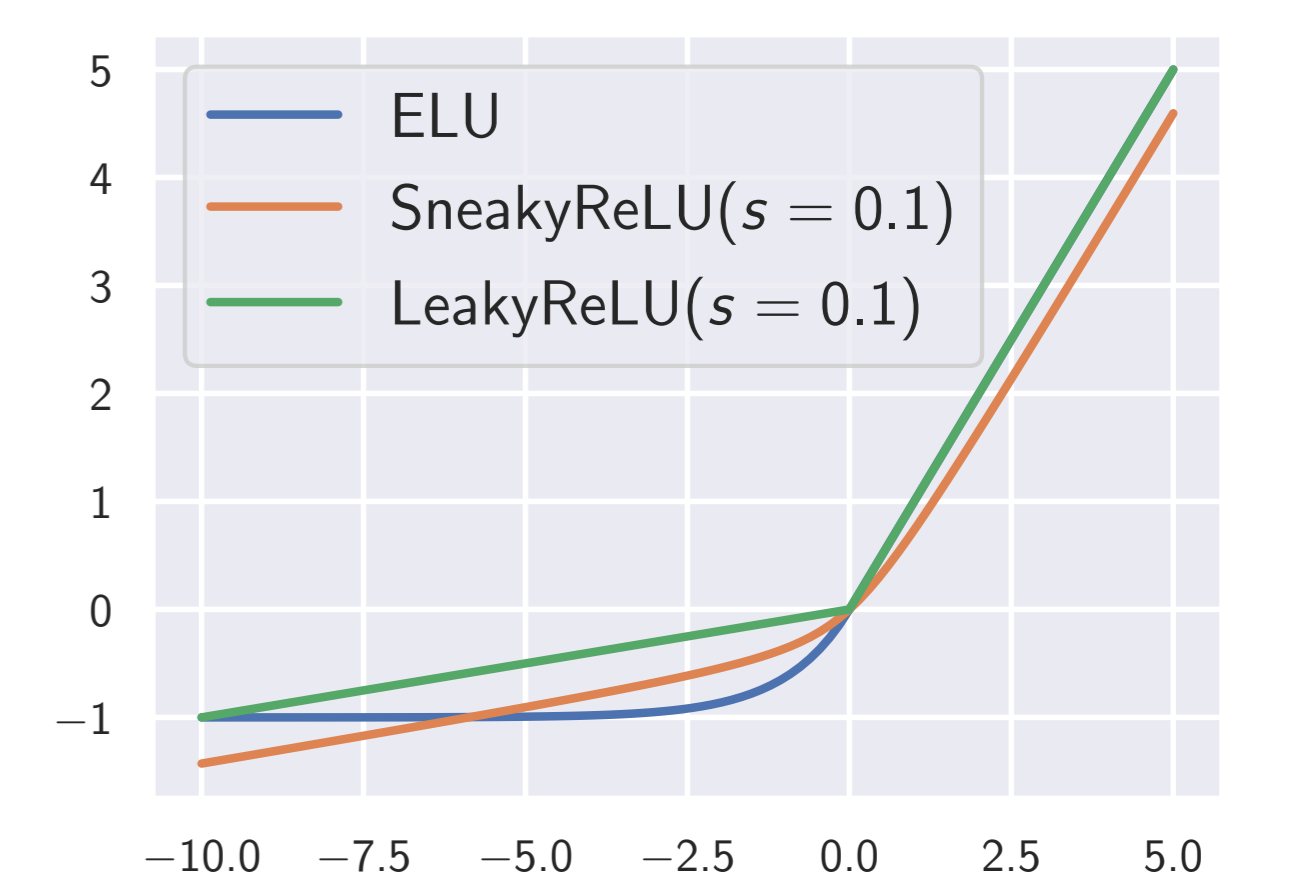
Since ReLU is not differentiable, we replace it with a smoothed modification of LeakyReLU, SneakyReLU.

$$\sigma(x) = (x + \alpha(\sqrt{1+x^2} - 1)) / (1 + \alpha),$$

$\alpha = \frac{1-s}{1+s}$  where  $s$  is the slope as  $x \rightarrow -\infty$ , and the function has a closed form inverse.

Change	Accuracy	Diff
Base Convnet (4M params, 9 layers)	94.6	-
Zero Padding $\rightarrow$ Circular Padding	94.6	0
ReLU $\rightarrow$ SneakyReLU	93.0	-1.6
MaxPool2d $\rightarrow$ Checkerboard DS	92.0	-1.0
MaxPool2d $\rightarrow$ iINN	92.5	-0.5
MaxPool2d $\rightarrow$ iAvgPool2d	92.6	-0.4

**Table 1:** Path to Invertibility (CIFAR10 classification performance).



**Figure 3:** SneakyReLU.

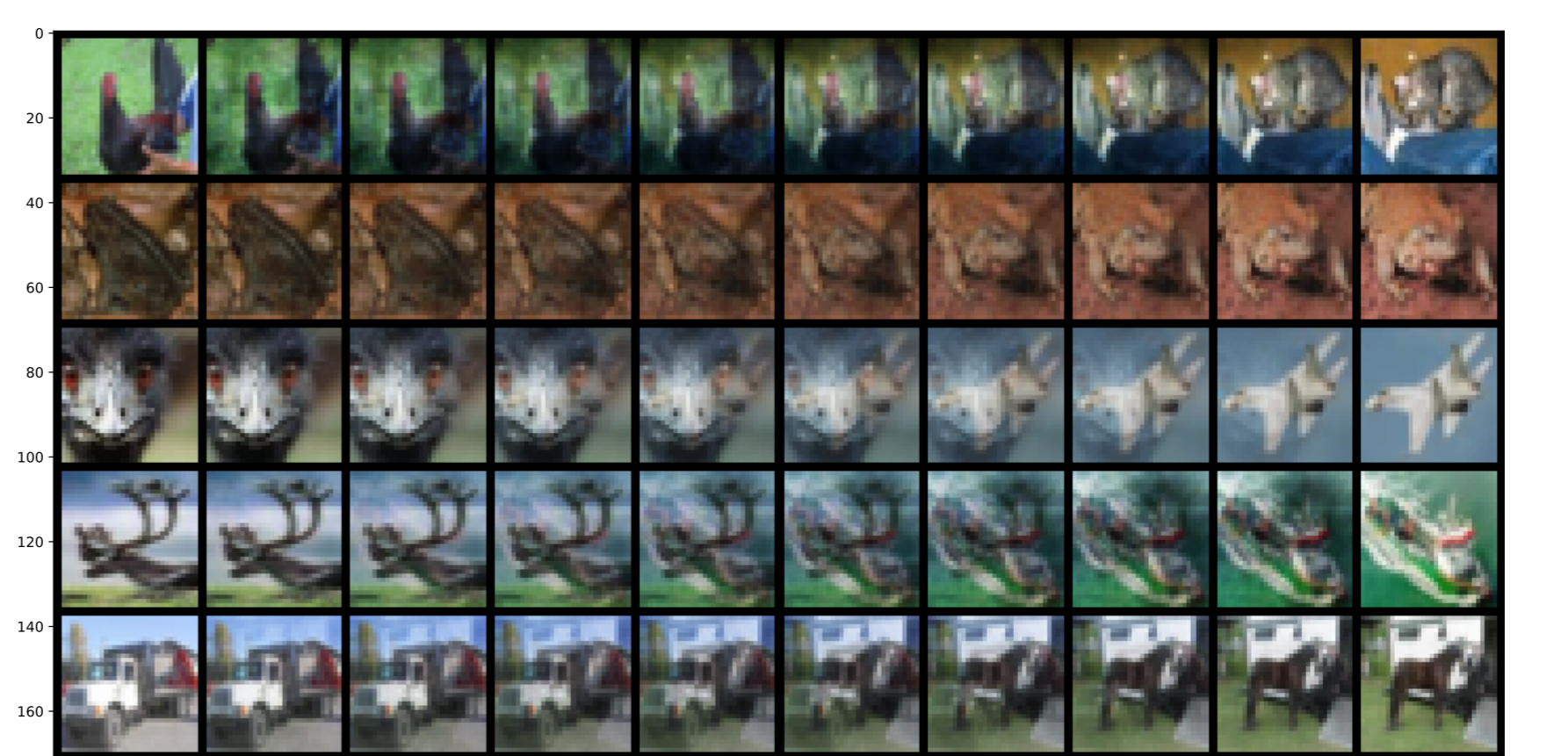
## Fully Convolutional Normalizing Flow

Our adapted CNN is invertible and allows for fast log-determinants, so we can train it as a normalizing flow,

$$\log |\det(\text{Df})| = \sum_{i=1}^L (\log |\det(\text{Conv}_W^{(i)})| + \sum_{j,h,w} \log \sigma'(a_{j,h,w}^{(i)})).$$

Method (CIFAR-10)	BPD
MADE (Germain et al)	5.67
RealNVP (Dinh et al)	3.49
Glow (Kingma et al)	3.35
i-ResNet (Behrmann et al)	3.45
i-ConvNet	4.61

**Table 2:** Bits per dimension



**Figure 4:** Latent space interpolations.