Invertible Convolutional Flow

M. Karami, J. Sohl-Dickstein, D. Schuurmans, L. Dinh, D. Duckworth

University of Alberta, Google

Two ways to improve expressivity of normalizing flow:

- **Invertible convolution filter**

- **Invertible nonlinear gates**

Circular Convolution

- Linear convolution of two sequences when one is padded cyclically

$$\boldsymbol{y}(i) := \sum_{n=0}^{N-1} \boldsymbol{x}(n)\boldsymbol{w}(i-n)_{\text{ mod } N}$$

- Jacobian of this convolution forms a *circulant matrix*

- Its eigenvalues are equal to the DFT of w, so

$$\log|\det \boldsymbol{J}_y| = \sum_{n=0}^{N-1} \log|\boldsymbol{w}_{\mathcal{F}}(n)|$$

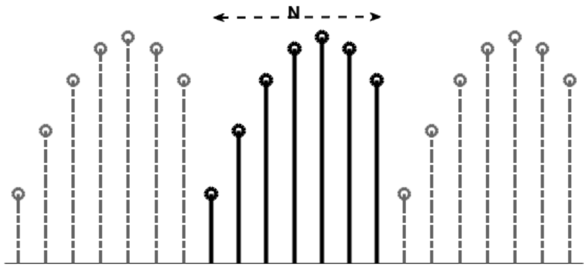
- The circular convolution-multiplication property

$$\boldsymbol{y}_{\mathcal{F}}(k) = \boldsymbol{w}_{\mathcal{F}}(k)\, \boldsymbol{x}_{\mathcal{F}}(k)$$

- Inverse operation (deconvolution)

$$\boldsymbol{x}_{\mathcal{F}}(n) = \boldsymbol{w}_{\mathcal{F}}^{-1}(n)\; \boldsymbol{y}_{\mathcal{F}}(n)$$

- These can be evaluated in *O(N logN)* time in the frequency domain, using FFT algorithms.



$$\boldsymbol{J}_C = \begin{bmatrix} w_0 & w_{N-1} & \cdots & w_2 & w_1 \\ w_1 & w_0 & \ddots & \ddots & w_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ w_{N-2} & \ddots & \ddots & w_0 & w_{N-1} \\ w_{N-1} & w_{N-2} & \cdots & w_1 & w_0 \end{bmatrix}$$

Symmetric Convolution

- Using *even-symmetric expansion*

$$\hat{\boldsymbol{x}}(n) = \varepsilon\{\boldsymbol{x}(n)\} := \begin{cases} \boldsymbol{x}(n) & n = 0, 1, ..., N-1 \\ \boldsymbol{x}(-n-1) & n = -N, ..., -1 \end{cases}$$

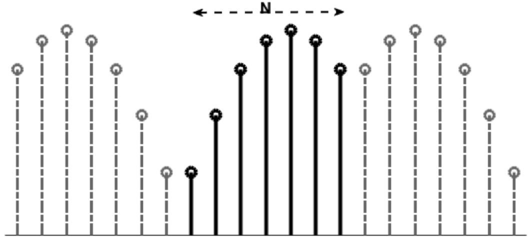
- The symmetric convolution can be defined as

$$\boldsymbol{y} = \boldsymbol{w} \ *_s \ \boldsymbol{x} := \mathcal{R}\{\hat{\boldsymbol{x}} \circledast \hat{\boldsymbol{w}}\}$$

- The *convolution-multiplication property* holds for DCT of operands

$$\mathcal{F}_{dct}\{\boldsymbol{y}\} = \mathcal{F}_{dct}\{\boldsymbol{w}\} \odot \mathcal{F}_{dct}\{\boldsymbol{x}\}$$

- The convolution, its Jacobian-determinant and inversion (deconvolution) can be performed efficiently in *O(N logN)*.



$$\boldsymbol{J}_S = \begin{bmatrix} w_0 & w_0 & \cdots & w_{N-3} & w_{N-2} \\ w_1 & w_0 & \ddots & w_{N-4} & w_{N-3} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ w_{N-2} & w_{N-3} & \ddots & w_0 & w_0 \\ w_{N-1} & w_{N-2} & \cdots & w_1 & w_0 \end{bmatrix} + \begin{bmatrix} w_1 & w_2 & \cdots & w_{N-1} & w_{N-1} \\ w_2 & w_3 & \iddots & w_{N-1} & w_{N-2} \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ w_{N-1} & w_{N-1} & \iddots & w_2 & w_1 \\ w_{N-1} & w_{N-2} & \cdots & w_1 & w_0 \end{bmatrix}$$

data-adaptive invertible convolution flow

- Let x_1 and x_2 are the disjoint parts of the input x.
- A **data-adaptive convolution** is defined by convolving x_2 with an arbitrary function of x_1

$$f_*(\boldsymbol{x}_2; \boldsymbol{x}_1) = \boldsymbol{w}(\boldsymbol{x}_1) * \boldsymbol{x}_2$$

- Using any of the invertible convolutions, this transform is invertible with cheap inversion and cheap log-det-Jacobian computation

Pointwise nonlinear bijectors

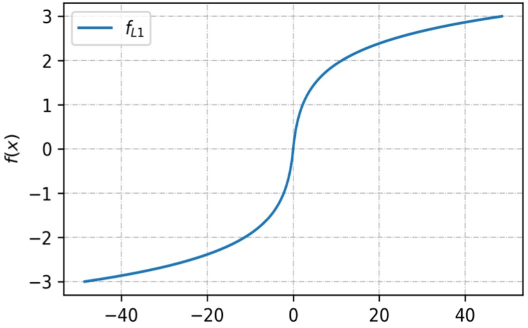
- log-det-Jacobian term in the log-likelihood equation can be interpreted as a regularizer.
- If we would like to encourage some desirable statistical properties, formulated by a regularizer *γ(y)*, in intermediate layers of a flow-based model, we can do so by carefully designing nonlinearities *y=f(x)*.

- *f(x)* is obtained by solving the differential equation

$$|\frac{\partial f^{-1}}{\partial y}| = |\frac{\partial g}{\partial y}| = e^{\gamma(y)}$$

- For *l1* regularization, inducing sparsity, this leads to the **S-Log** gate defined as

$$f_\alpha(x) = \frac{\text{sign}(x)}{\alpha} \ln(\alpha|x| + 1)$$
$$f_\alpha^{-1}(y) = \frac{\text{sign}(y)}{\alpha} (e^{\alpha|y|} - 1)$$

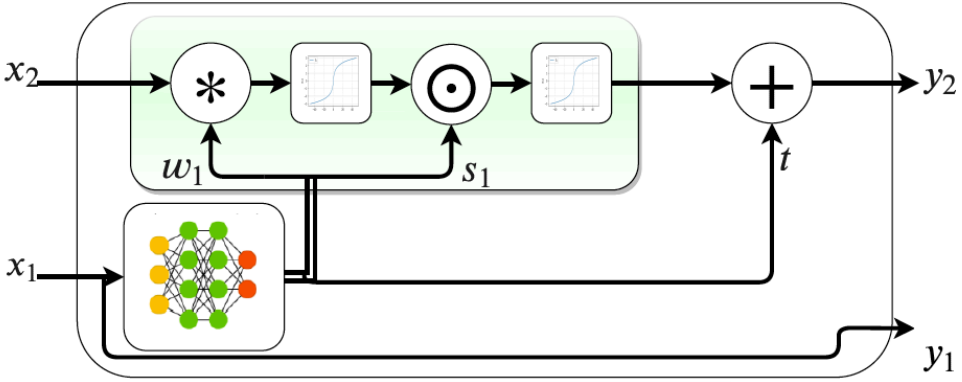


S-Log gate which is differentiable and has unbounded domain and range by construction

Convolutional coupling flow (CONF)

- Combining the invertible convolution, element-wise multiplication and nonlinear bijectors, we achieve a more expressive flow in the coupling form:

$$\begin{cases} \boldsymbol{y}_1 = \boldsymbol{x}_1 \\ \boldsymbol{y}_2 = f_{\alpha'}\big(\, \boldsymbol{s}(\boldsymbol{x}_1) \odot f_{\alpha}(\boldsymbol{w}(\boldsymbol{x}_1) * \boldsymbol{x}_2)\,\big) + \boldsymbol{t}(\boldsymbol{x}_1) \end{cases}$$



POSTER 3011