Key Properties for Tractability

#1 Compositionality

#2 Link Between Chain Rule and Triangular Jacobians

#3 Free to Choose Directionality

#2 Chain Rule and Triangular Jacobians

Autoregressive flows:

$$x_d = \tau(u_d; \boxed{c(\mathbf{u}_{<d})})$$

Invertible *transformer*

Arbitrary *conditioner* (may not be invertible)

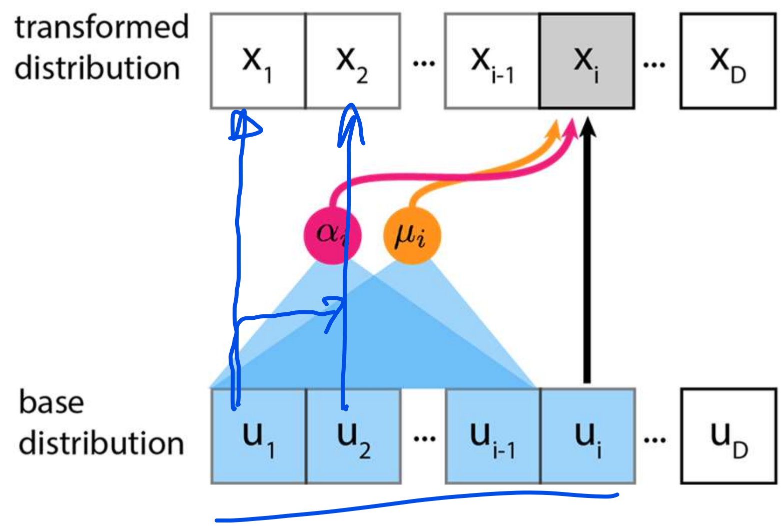


Diagram taken from Eric Jang's blog: https://blog.evjang.com/2018/01/nf2.html

$x_1 = T_1(\vec{u}) = \tau(u_1)$

$x_2 = T_2(\vec{u}) = \tau(u_2; c(u_1))$

$x_3 = \tau(u_3; c(u_1, u_2))$

$u_1 = \tau^{-1}(x_1)$; $u_2 = \tau^{-1}(x_2; c(u_1))$

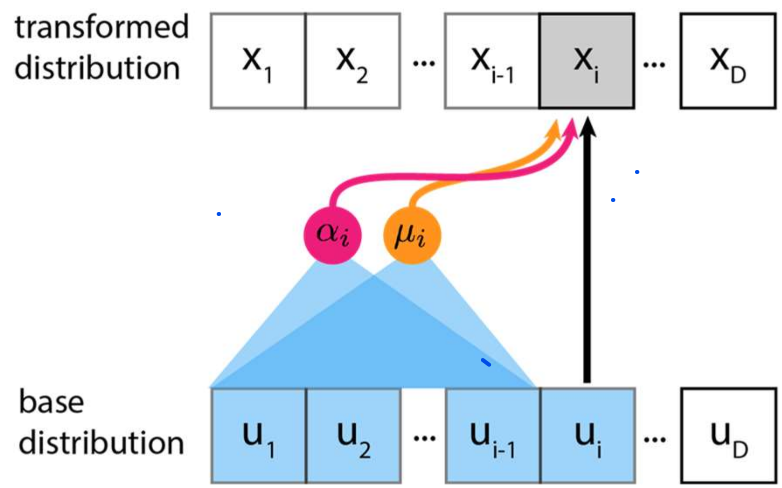
$u_3 = \tau^{-1}(x_3; c(u_1, u_2))$

$T_i(\vec{u}) = T_i(u_1, \ldots u_i)$

#2 Chain Rule and Triangular Jacobians

Autoregressive flows:

$$x_d = \tau(u_d; c(\mathbf{u}_{<d}))$$



Can be shown to be *universally expressive* due to their connection to the chain rule of probability.

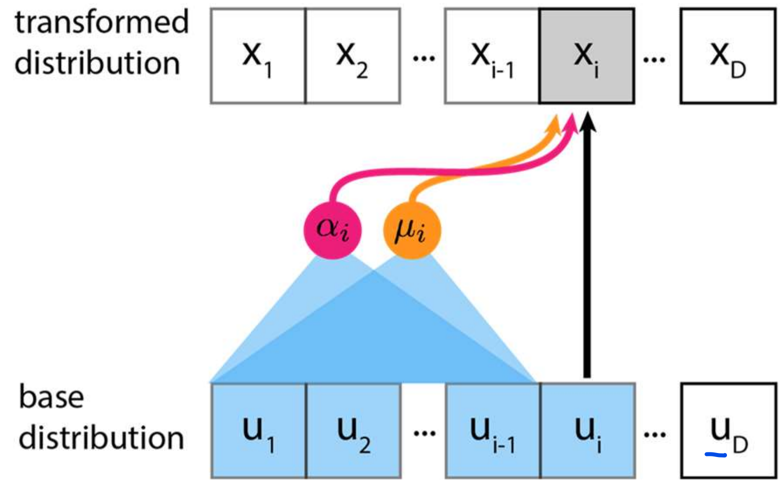
$$p(\mathbf{x}) = \prod_d p(x_d | \mathbf{x}_{<d})$$

Diagram taken from Eric Jang's blog: https://blog.evjang.com/2018/01/nf2.html

#2 Chain Rule and Triangular Jacobians

Autoregressive flows:

$$x_d = \tau(u_d; c(\mathbf{u}_{<d}))$$



Are efficient to implement due to having a triangular Jacobian ($O(D)$ determinant).

$$\begin{bmatrix} \frac{\partial \tau}{\partial \underset{u_1}{\cancel{\mathrm{z}_1}}}(\overset{u_1}{\cancel{\mathrm{z}_1}}; \boldsymbol{h}_1) & & \mathbf{0} \\ & \ddots & \\ \mathbf{L}(\mathbf{z}) & & \frac{\partial \tau}{\partial \underset{u_D}{\cancel{\mathrm{z}_D}}}(\overset{u_D}{\cancel{\mathrm{z}_D}}; \boldsymbol{h}_D) \end{bmatrix}$$

Computing determinant of AR flows

$$J_T = \begin{bmatrix} \frac{\partial T_1}{\partial u_1} & & & \\ & \frac{\partial T_2}{\partial u_2} & \frac{\partial T_1}{\partial u_i} = 0 & \\ & & \ddots & \\ & & & \frac{\partial T_D}{\partial u_D} \end{bmatrix}$$

$i > 1$

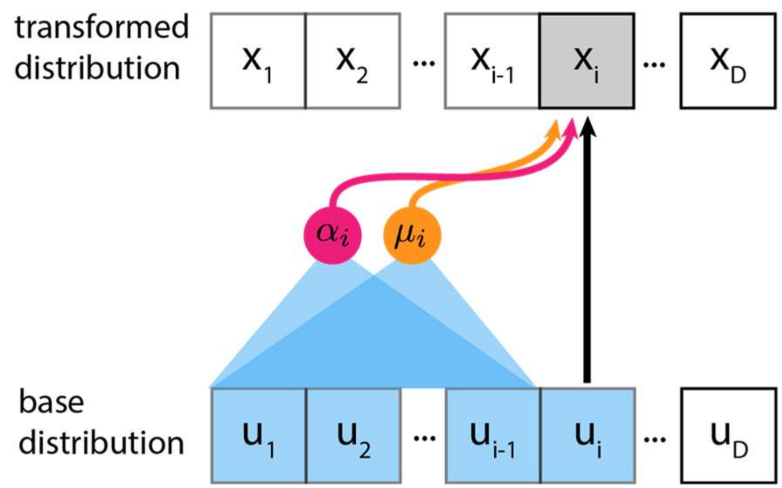
Lower triangular matrix.

Determinant? $\prod_{i=1}^{D} \frac{\partial T_i}{\partial u_i}$

#2 Chain Rule and Triangular Jacobians

Autoregressive flows:

$$x_d = \tau(u_d; c(\mathbf{u}_{<d}))$$



The autoregressive structure yields *both* **expressivity and practicality**.

Diagram taken from Eric Jang's blog:
https://blog.evjang.com/2018/01/nf2.html

Types of Auto-regressive flows

- Affine functions

$$x_i = \tau(u_i, c(u_{<i})) = \alpha_i u_i + \beta_i$$

$$u_i = \tau^{-1}(x_i, c(u_{<i})) = \frac{x_i - \beta_i}{\alpha_i}$$

$$\log |J_T| = \sum_{i=1}^{D} \log \frac{\partial T_i}{\partial u_i} = \sum_{i=1}^{D} \log|\alpha_i| = \sum_{i=1}^{D} \tilde{\alpha}_i$$

$\alpha_i \neq 0$

$$\alpha_i = \boxed{C_1(u_1, u_2 \ldots u_{i-1})}$$

$$\beta_i = C_2(u_1, u_2 \ldots u_{i-1})$$

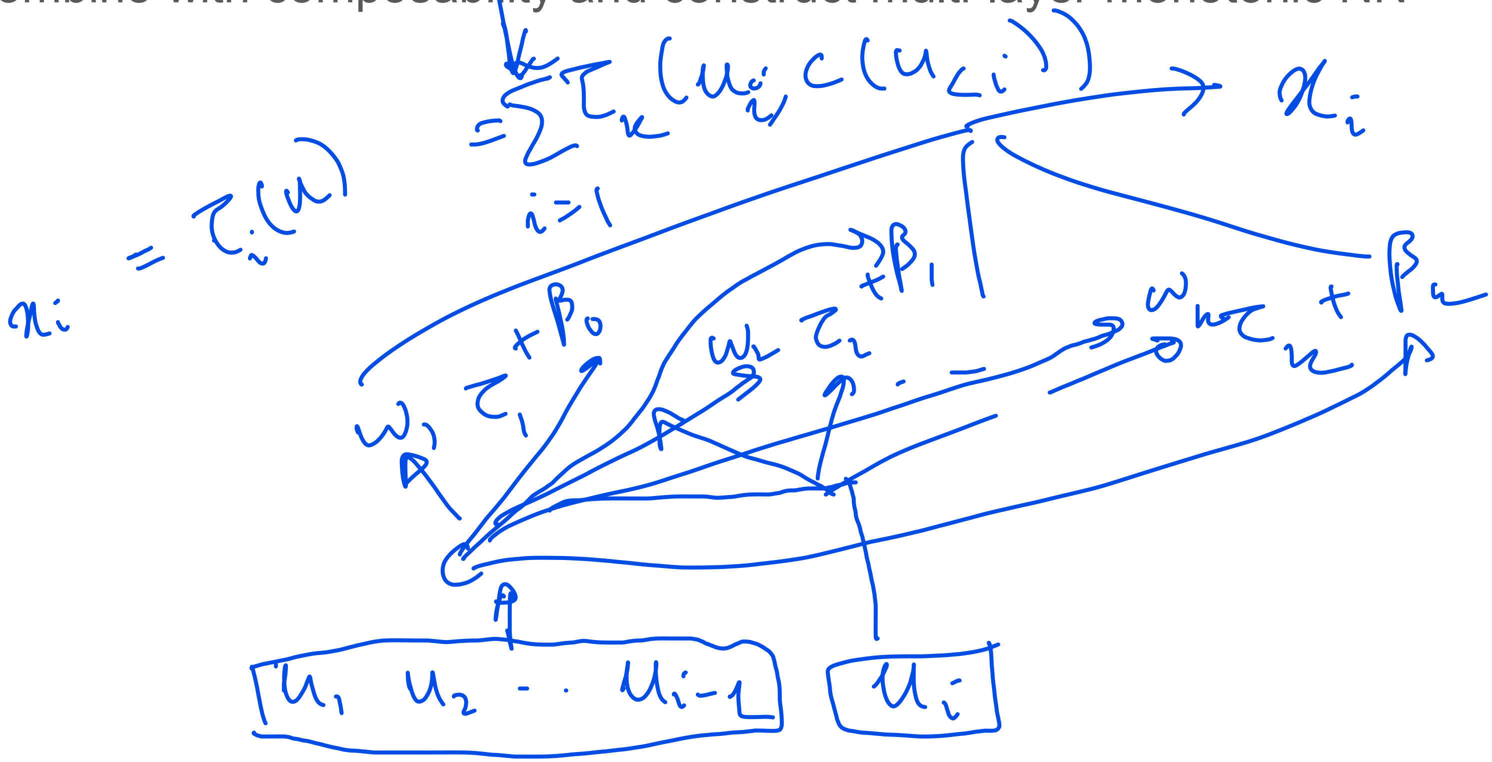
$$\alpha_i = e^{\tilde{\alpha}_i}$$

$$\tilde{\alpha}_i = \tilde{C}_1(u_1 \ldots u_{i-1})$$

Combination-based transformers

- Monotonic differentiable functions $\tau_1, \dots, \tau_K$, example sigmoid, tanh, leaky ReLU
- Positive weights $w_1, \dots w_K$
- Then, $\tau(u) = \sum_k w_k \tau_k(u)$ is also bijective.
- Combine with composability and construct multi-layer monotonic NN

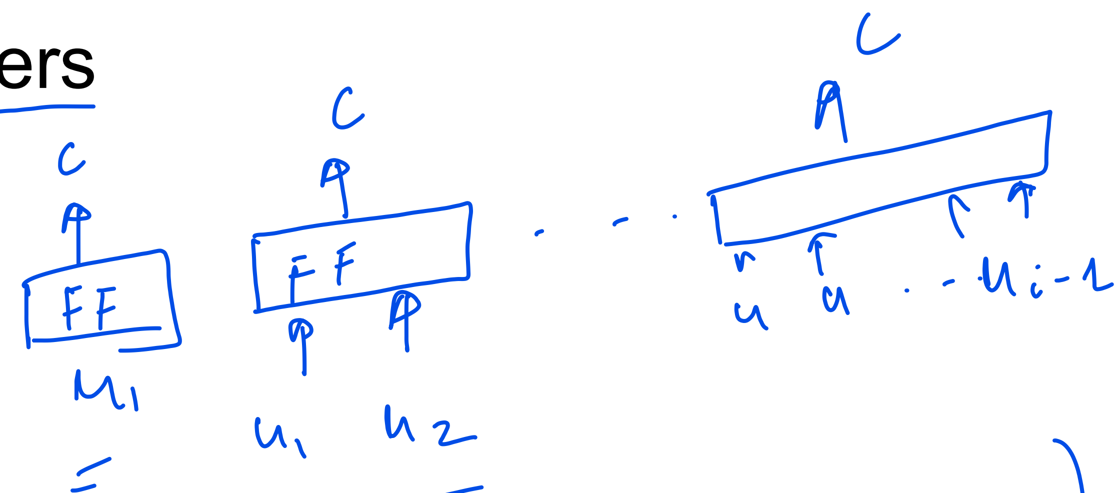
$u \in \mathbb{R}$



Implementing Conditioners

- Separate networks for each i

D is small $\quad c(u_1)$



- Recurrent conditioners

$c(u_1 \dots u_i) = h_i \leftarrow$ RNN state $(u_1, \dots u_{i-1})$

$RNN(h_i; u_i) \rightarrow h_{i+1} \rightarrow c_{i+1}$

- Masked conditioners

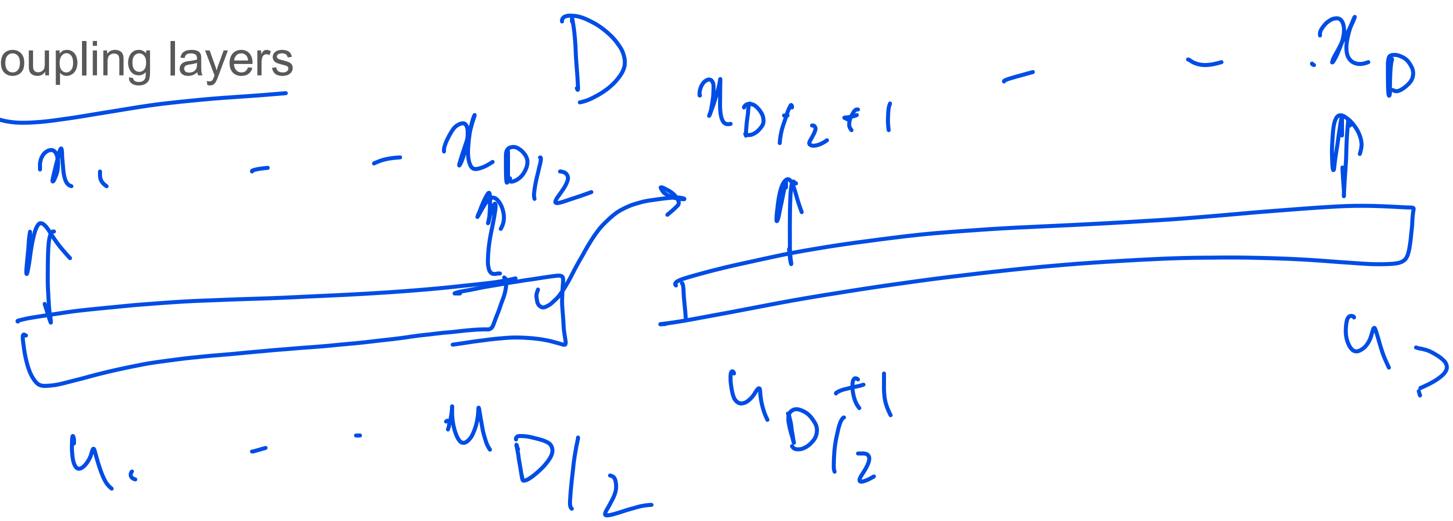
D

$c_i = T(u_1 \dots u_{i-1})$ Transformers

$c_i = (u_1, \dots u_D) \odot$ (Masking)

$\underbrace{1\ 1 \dots 1}_{i-1}\ 0 \dots 0$

- Coupling layers



- Composition with permutations

permutation

$x_1 \cdots x_D$

$z_1 \cdots z_D$

$u_1 \cdots u_D$

Key Properties for Tractability

#1 Compositionality

#2 Link Between Chain Rule and Triangular Jacobians

#3 Free to Choose Directionality