PDF = "probability per unit volume"

- 1D: RV w/ PDF $p_x(x)$ $\Rightarrow$ $\Pr(X \in [x, x+\Delta x]) \approx p_x(x)\,\Delta x$
- nD: " $\Rightarrow$ $\Pr(X \in \text{tiny box around } x) \approx p_x(x) \cdot (\text{volume of that box})$

Jacobian determinant = local volume scaling

- For a smooth map $f: \mathbb{R}^d \to \mathbb{R}^d$, near a point x, the map is approximately linear:
  - ↳ $f(x+\delta) \simeq f(x) + J_f(x)\,\delta$ $\Rightarrow$ $\underline{(\text{new volume}) \approx |\det J_f(x)| \cdot (\text{old volume})}$ Δ of variables formula

Change-of-var. gives the flow likelihood:

Define $z = f_\theta(q; h)$, choose simple PDF for z: $p_z(z) = \mathcal{N}(0, I)$

↳ Take a tiny region A around q $\Rightarrow$ its img in z-space = $B = f(A)$

↳ ∵ this is reparameterizing the same outcomes: $\underline{\Pr(q \in A \mid h) = \Pr(z \in B \mid h)}$

Approximating both sides using $\Pr(X \in \text{tiny box around } x) \approx p_x(x) \cdot (\text{volume of that box})$: $p_\theta(q \mid h)\,\mathrm{Vol}(A) \approx p_z(z \mid h)\,\mathrm{Vol}(B)$

since $z = f(q; h)$: $\mathrm{Vol}(B) \approx |\det J_f(q; h)|\,\mathrm{Vol}(A)$

↓

∴ $p_\theta(q \mid h)\,\cancel{\mathrm{Vol}(A)} \approx p_z(f(q; h))\,|\det J_f(q; h)|\,\cancel{\mathrm{Vol}(A)}$

↳ Taking logs (base e) & using Gaussian p_z: $\overbrace{-\log p_\theta(q \mid h) = \underbrace{\tfrac{1}{2}\|z\|^2}_{\text{make } z \text{ look Gaussian}} - \underbrace{\log|\det J_f(q; h)|}_{\text{volume correction}} + \text{const.}}^{\text{flow training objective}}$

⚠ Generic neural nets can't model $f_\theta(q; h)$

↳ Problems:
1. computing $\det(J_f(q; h))$ is inefficient
2. f is not invertible

e.g. if using MLP

↓

Need special architecture for flow models $\Rightarrow$ e.g. coupling layers!

Coupling layers:

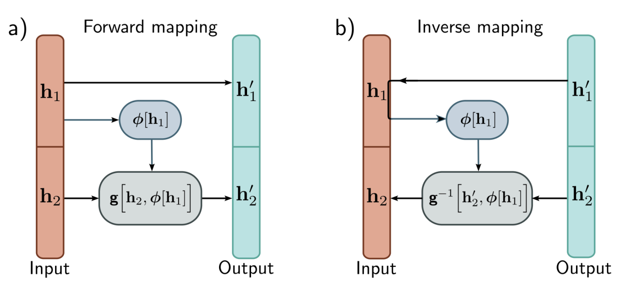

Figure 16.6 Coupling flows. a) The input (orange vector) is divided into $\mathbf{h}_1$ and $\mathbf{h}_2$. The first part $\mathbf{h}'_1$ of the output (cyan vector) is a copy of $\mathbf{h}_1$. The output $\mathbf{h}'_2$ is created by applying an invertible transformation $\mathbf{g}[\bullet, \phi]$ to $\mathbf{h}_2$, where the parameters ϕ are themselves a (not necessarily invertible) function of $\mathbf{h}_1$. b) In the inverse mapping, $\mathbf{h}_1 = \mathbf{h}'_1$. This allows us to calculate the parameters $\phi[\mathbf{h}_1]$ and then apply the inverse $\mathbf{g}^{-1}[\mathbf{h}'_2, \phi]$ to retrieve $\mathbf{h}_2$.

$\Rightarrow$ e.g. let $x \in \mathbb{R}^d$ $\Rightarrow$ split into $x = (x_a, x_b)$, $x_a \in \mathbb{R}^{d_a}$, $x_b \in \mathbb{R}^{d_b}$

- **Additive coupling**: define $y = g(x; c)$ $\begin{cases} y_a = x_a \\ y_b = x_b + t_\theta(x_a, c) \end{cases}$
  - ↳ where t_θ = any NN outputting a vector $\in \mathbb{R}^{d_b}$
  - > Invertible: ✓ can recover x from y ↓
    $\left.\begin{aligned} x_a &= y_a \\ x_b &= y_b - t_\theta(y_a, c) \end{aligned}\right\}$ g^{-1} exists & is easy to compute
  - > Jacobian w/ easy det: ✓
    $$J = \frac{\partial y}{\partial x} = \begin{bmatrix} \partial y_a/\partial x_a & \partial y_a/\partial x_b \\ \partial y_b/\partial x_a & \partial y_b/\partial x_b \end{bmatrix} = \begin{bmatrix} I & 0 \\ \partial t/\partial x_a & I \end{bmatrix}$$
    det(triangular mtx) = $\prod$ diag. entries
    ∴ $\det(J) = \det(I) \cdot \det(I) = 1$

- Additive coupling is invertible & $\det(J)$ efficient to compute, but it can't expand / contract volume
  ↳ ∴ limited expressiveness

★ FrEIA AllInOneBlock combines affine coupling, permutation, ... etc. common normalizing flow ops.

- **Affine coupling:** $y = g(x; c)$ $\begin{cases} y_a = x_a \\ y_b = x_b \odot \exp(s_\theta(x_a, c)) + t_\theta(x_a, c) \end{cases}$ elementwise mult. ($y = ax + b \Rightarrow$ "affine")
  ↳ s_θ = scale network $\Rightarrow s_\theta(x_a, c) = (s_1, \dots, s_{d_b})$

> **Invertible:** ✓ $x_a = y_a$
$$x_b = (y_b - t_\theta(x_a, c)) \odot \frac{1}{\exp(s_\theta(x_a, c))} = (y_b - t_\theta(x_a, c)) \odot \exp(-s_\theta(x_a, c))$$

> **Jacobian w/ easy det:** $(y_b)_k = (x_b)_k \cdot \exp(s_k(x_a, c)) + t_k(x_a, c)$

when differentiating w.r.t. x_b: $\frac{\partial (y_b)_k}{\partial (x_b)_k} = \exp(s_k(x_a, c))$, for $j \neq k$: $\frac{\partial (y_b)_k}{\partial (x_b)_j} = 0$

$$\therefore \frac{\partial y_b}{\partial x_b} = \mathrm{diag}(\exp(s_1), \dots, \exp(s_{d_b}))$$

$$\Rightarrow J = \begin{bmatrix} I & 0 \\ * & \mathrm{diag}(\exp(s)) \end{bmatrix} \Rightarrow \det(J) = \det(I) \cdot \det(\mathrm{diag}(\exp(s))) = \prod_{k=1}^{d_b} \exp(s_k) = \exp\left(\sum_{k=1}^{d_b} s_k\right)$$

$$\Rightarrow \log|\det J| = \sum_{k=1}^{d_b} s_k$$ ✓ easy to compute ◡̈

⚠ But half the vars don't change ?!

↓

In one coupling layer, x_a stays exactly the same ($y_a = x_a$) $\Rightarrow$ only x_b changes

↳ **Fix:** after each coupling layer, permute s.t. different subset of x becomes x_a next layer
$\Rightarrow$ over multiple layers, every coord. gets to be in the "changed group" multiple times

Conditioning preserves invertibility: [Recall: coupling layer requires generating s & t from NN]
↳ $y_b = x_b \odot \exp(s) + t$

- unconditional case: $(s, t) = NN_\theta(x_a)$
- **conditional case:** $(s, t) = NN_\theta(x_a, c)$

$\Rightarrow$ c is like a hyperparam: we only need $x \mapsto y = g_c(x)$ to be invertible for each fixed c
↳ c is given @ both forward & reverse time

↓

e.g. affine coupling w/ conditioning: $y_a = x_a$, $(s, t) = NN_\theta(x_a, c)$, $y_b = x_b \odot \exp(s) + t$

↓

Showing this is invertible:
1. $x_a = y_a$
2. $(s, t) = NN_\theta(y_a, c)$
3. Solve for x_b: $y_b = x_b \odot \exp(s) + t$

Intuition for applying flow models to our reachability problem:

↳ Goal post-training: for given $H = c$, sampling different $z \sim \mathcal{N}(0, I)$ gives different valid Q

↓

Flow model learns a geometry-aware warp s.t. the solution distr. becomes simple in latent space

Concrete cINN example w/ our setup: e.g. choose a split for one layer: $x_a = (x, y)$ $x_b = (\cos\theta, \sin\theta)$

↳ e.g. affine coupling layer: $y_a = x_a$, $(s, t) = NN_\theta(x_a, H)$ where $s, t \in \mathbb{R}^2$, $y_b = x_b \odot \exp(s) + t$

aka. given position + target, you rescale/shift angle features

↳ next layer can permute to $x_a = (\cos\theta, \sin\theta)$, $x_b = (x, y)$ $\Rightarrow$ angle features control how you rescale/shift position features

$\Downarrow$

Stack enough of these & we get a rich coupling btwn parts of Q relative to H

clamp exists in FrEIA ∵ affine coupling uses exp(s) → if s gets big, exp(s) blows up

↳ Implementations squash s into a bounded range

Why cINN is potentially good for our problem:

- Multi-modal: different modes correspond to different z-values

  ↓

  Training is max. likelihood: penalizes missing prob. mass (dropping modes)

Dimensions in our simplified setup: (maps to code implementation)

- Input to flow: $x = Q_{fear} \in \mathbb{R}^4$
- condition: $c = H \in \mathbb{R}^2$
- output (forward): $z \in \mathbb{R}^4$
- sampling (reverse): sample $z \sim \mathcal{N}(0, I)$ ⇒ compute $x = f^{-1}(z; c)$
- convert $x = (x, y, \underline{\cos\theta, \sin\theta})$ back to $Q = (x, y, \theta)$ w/ $\theta = \text{atan2}(\sin\theta, \cos\theta)$

  normalize to unit len. before atan2 for stability