

# Deep Learning

## Lecture 9: Flow models and implicit networks

---

Chris G. Willcocks

Durham University



## 1 Flow models

---

- definition
- the determinant
- the change of variables theorem

## 2 Normalising flows

---

- definition
- triangular Jacobians
- normalising flow layers

## 3 Implicit representation networks

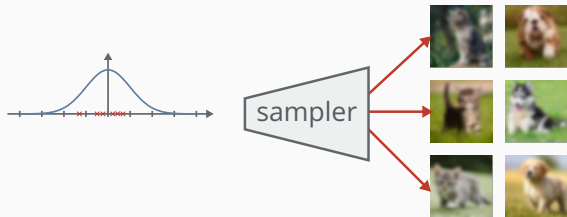
---

- definition
- SIREN
- GONs

**Definition:** Generative models learn a joint distribution over the entire dataset. They are mostly used for sampling applications or density estimation:

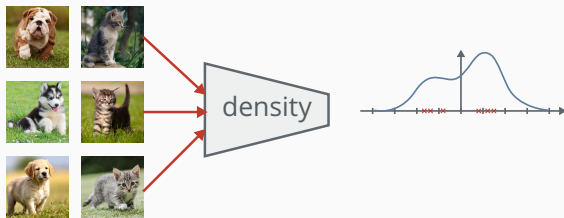
## Inference: sampling

A generative model learns to fit a model distribution over observations so we can sample novel data from the model distribution,  $\mathbf{x}_{\text{new}} \sim p_{\text{model}}(\mathbf{x})$



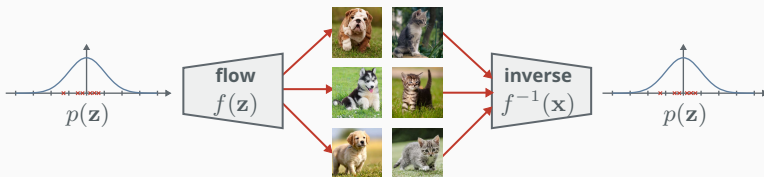
## Inference: density estimation

Density estimation is estimating the probability of observations. Given a datapoint  $\mathbf{x}$ , what is the probability assigned by the model,  $p_{\text{model}}(\mathbf{x})$ ?



## Definition: flow models

Flow models restrict our function to be a chain of invertible functions, called a flow, therefore the whole function is invertible.





## Definition: the Jacobian matrix

The collection of all first-order partial derivatives of a vector-valued function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$

$$\mathbf{J}_f = \nabla_{\mathbf{x}} f = \frac{df}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix},$$

$$\mathbf{J}_f(i, j) = \frac{\partial f_i}{\partial x_j}$$

## Definition: the determinant

The determinant of an  $n \times n$  square matrix  $M$  is a scalar value that determines the factor of how much a given region of space increases or decreases by the linear transformation of  $M$ :

$$\det M = \det \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \sum_{j_1 j_2 \dots j_n} (-1)^{\tau(j_1 j_2 \dots j_n)} a_{1j_1} a_{2j_2} \dots a_{nj_n}$$

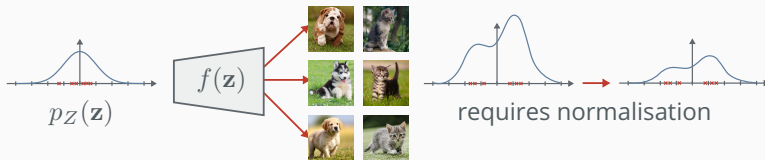
**Watch a 3Blue1Brown's video here** [↗](#)

PyTorch: `torch.det(M)`, for example: `torch.det(torch.eye(3,3))` returns 1.0  
and `torch.det(torch.tensor([[3.,2.],[0.,2.]])` returns 6.0

## Definition: the change of variables theorem

Given  $p_Z(\mathbf{z})$  where  $\mathbf{x} = f(\mathbf{z})$  and  $\mathbf{z} = f^{-1}(\mathbf{x})$  we ask what is  $p_X(\mathbf{x})$ ?

$$p_X(\mathbf{x}) = p_Z(f^{-1}(\mathbf{x})) \left| \det \left( \frac{\partial f^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

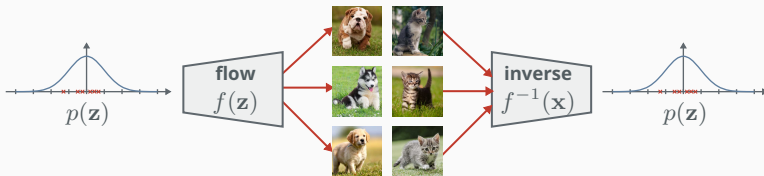


## Definition: normalising flows

Normalising flows  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$  transform and renormalise a sample  $\mathbf{z} \sim p_\theta(\mathbf{z})$  through a chain of bijective transformations  $f$ , where:

$$\mathbf{x} = f_\theta(\mathbf{z}) = f_K \circ \dots \circ f_2 \circ f_1(\mathbf{z})$$

$$\log p_\theta(\mathbf{x}) = \log p_\theta(\mathbf{z}) + \sum_{i=1}^K \log \left| \det \left( \frac{\partial f_i^{-1}}{\partial \mathbf{z}_i} \right) \right|$$





## Easy to compute determinants

We have a sequence of high-dimensional bijective functions, where we need to compute the Jacobian determinants.

Computing the determinants can be expensive, so most of the literature focuses on restricting the function  $f^{-1}$  to those with easy-to-compute Jacobian determinants.

This is done by ensuring the Jacobian matrix of the functions is triangular.

## Definition: triangular Jacobian

If the Jacobian is lower triangular:

$$J = \begin{bmatrix} a_{1,1} & & & & & 0 \\ a_{2,1} & a_{2,2} & & & & \\ a_{3,1} & a_{3,2} & \ddots & & & \\ \vdots & \vdots & \ddots & \ddots & & \\ a_{n,1} & a_{n,2} & \dots & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

then the determinant is simply the product of its **diagonals**.



# Normalising flows normalising flow layers

Description	Function	Log-Determinant
Additive Coupling [1]	$\mathbf{y}^{(1:d)} = \mathbf{x}^{(1:d)}$ $\mathbf{y}^{(d+1:D)} = \mathbf{x}^{(d+1:D)} + f(\mathbf{x}^{(1:d)})$	0
Planar [2]	$\mathbf{y} = \mathbf{x} + \mathbf{u}h(\mathbf{w}^T \mathbf{z} + b)$ <p>With <math>\mathbf{w} \in \mathbb{R}^D, \mathbf{u} \in \mathbb{R}^D, \mathbf{b} \in \mathbb{R}</math></p>	$\ln  1 + \mathbf{u}^T h'(\mathbf{w}^T \mathbf{z} + b) \mathbf{w} $
Affine Coupling [3]	$\mathbf{y}^{(1:d)} = \mathbf{x}^{(1:d)}$ $\mathbf{y}^{(d+1:D)} = \mathbf{x}^{(d+1:D)} \odot f_\sigma(\mathbf{x}^{(1:d)}) + f_\mu(\mathbf{x}^{(1:d)})$	$\sum_1^d \ln  f_\sigma(x^{(i)}) $
Batch Normalization [3]	$\mathbf{y} = \frac{\mathbf{x} - \tilde{\boldsymbol{\mu}}}{\sqrt{\tilde{\sigma}^2 + \epsilon}}$	$-\frac{1}{2} \sum_i \ln(\tilde{\sigma}_i^2 + \epsilon)$
1x1 Convolution [4]	<p>With <math>h \times w \times c</math> tensor <math>\mathbf{x}</math> &amp; <math>c \times c</math> tensor <math>\mathbf{W}</math></p> $\forall i, j : \mathbf{y}_{i,j} = \mathbf{W} \mathbf{x}_{i,j}$	$h \cdot w \cdot \ln  \det \mathbf{W} $
i-ResNet [5]	$\mathbf{y} = \mathbf{x} + f(\mathbf{x})$ <p>where <math>\ f\ _L &lt; 1</math></p>	$\text{tr}(\ln(\mathbf{I} + \nabla_{\mathbf{x}} f)) =$ $\sum_{k=1}^{\infty} (-1)^{k+1} \frac{\text{tr}((\nabla_{\mathbf{x}} f)^k)}{k}$
Emerging Convolutions [6]	$\mathbf{k} = \mathbf{w}_1 \odot \mathbf{m}_1, \quad \mathbf{g} = \mathbf{w}_2 \odot \mathbf{m}_2$ $\mathbf{y} = \mathbf{k} \star_l (\mathbf{g} \star_l \mathbf{x})$	$\sum_c \ln  \mathbf{k}_{c,c,m_y,m_x} \mathbf{g}_{c,c,m_y,m_x} $

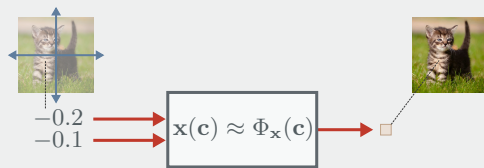
## Definition: implicit networks

Consider data  $\Phi: \mathbb{R}^m \rightarrow \mathbb{R}^n$ , like a single image, as a function of coordinates  $\mathbf{c} \in \mathbb{R}^m$ . The aim is to learn a neural approximation of  $\Phi$  that satisfies an implicit equation:

$$R(\mathbf{c}, \Phi, \nabla_{\Phi}, \nabla_{\Phi}^2, \dots) = 0, \quad \Phi: \mathbf{c} \mapsto \Phi(\mathbf{c}).$$

Equations with this structure arise in a myriad of fields, namely 3D modelling, image, video, and audio representation.

## Example: implicit network

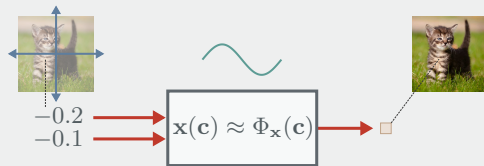


## Definition: SIREN

Sinusoidal REpresentation Networks (SIREN) are a simple implicit representation network with fully connected layers, but use `sin` (with clever initialisation to scale it appropriately) as their choice of non-linearity [7].

`sin` is periodic, so it allows to capture patterns over all of the coordinate space (it's translation invariant, like convolutions).

## Example: SIREN (implicit network)



[Link to project page](#) 

## Definition: gradient origin networks

Gradient origin networks (GON) treat the derivative of the decoder as an encoder [8]. This allows us to compute the latents:

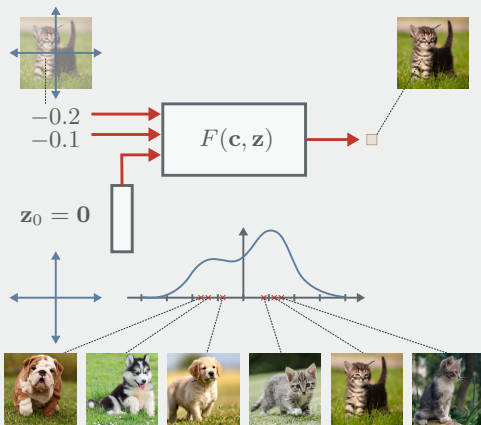
$$\mathbf{z} = -\nabla_{\mathbf{z}_0} \mathcal{L}(\mathbf{x}, F(\mathbf{z}_0))$$

which are then goingly optimised, giving the GON objective:

$$G_{\mathbf{x}} = \mathcal{L}(\mathbf{x}, F(-\nabla_{\mathbf{z}_0} \mathcal{L}(\mathbf{x}, F(\mathbf{z}_0)))).$$

[Link to project page](#) 

## Example: implicit GON





- [1] Laurent Dinh, David Krueger, and Yoshua Bengio. “Nice: Non-linear independent components estimation”. In: arXiv preprint arXiv:1410.8516 (2014).
- [2] Danilo Jimenez Rezende and Shakir Mohamed. “Variational inference with normalizing flows”. In: arXiv preprint arXiv:1505.05770 (2015).
- [3] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using real nvp”. In: arXiv preprint arXiv:1605.08803 (2016).
- [4] Durk P Kingma and Prafulla Dhariwal. “Glow: Generative flow with invertible 1x1 convolutions”. In: Advances in neural information processing systems. 2018, pp. 10215–10224.
- [5] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. “Invertible residual networks”. In: International Conference on Machine Learning. 2019, pp. 573–582.
- [6] Emiel Hoogetboom, Rianne van den Berg, and Max Welling. “Emerging convolutions for generative normalizing flows”. In: arXiv preprint arXiv:1901.11137 (2019).



- [7] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. "Implicit neural representations with periodic activation functions". In: Advances in Neural Information Processing Systems 33 (2020).
- [8] Sam Bond-Taylor and Chris G Willcocks. "Gradient Origin Networks". In: arXiv preprint arXiv:2007.02798 (2020).