

1 Generative modelling

Learn  $p_{\text{model}} \approx p_{\text{data}}$ , sample from  $p_{\text{model}}$ .

- Explicit density:
  - Approximate:
  - \* Variational: VAE, Diffusion
  - \* Markov Chain: Boltzmann machine
  - Tractable:
  - \* Autoregressive: FVSBN/NADE/MADE, Pixel(C/R)NN, WaveNet/TCN, Autor. Transf.,
  - \* Normalizing Flows
- Implicit density:
  - Direct: Generative Adversarial Networks
  - MC: Generative Stochastic Networks

Autoencoder:  $X \rightarrow Z \rightarrow X$ ,  $g \circ f \approx \text{id}$ ,  $f$  and  $g$  are NNs. Optimal linear autoencoder is PCA. Undercomplete:  $|Z| < |X|$ , else overcomplete. Overcomp. is for denoising, inpainting. Latent space should be continuous and interpolable. Autoencoder spaces are neither, so they are only good for reconstruction.

2 Variational AutoEncoder (VAE)

Sample  $z$  from prior  $p_\theta(z)$ , to decode use conditional  $p_\theta(x | z)$  defined by a NN.

$D_{\text{KL}}(P||Q) := \int_x p(x) \log \frac{p(x)}{q(x)} dx$ : KL divergence, measure similarity of prob. distr.  $D_{\text{KL}}(P||Q) \neq D_{\text{KL}}(Q||P)$ ,  $D_{\text{KL}}(P||Q) \geq 0$  Likelihood  $p_\theta(x) = \int_z p_\theta(x | z) p_\theta(z) dz$  is hard to max., let enc. NN be  $q_\phi(z | x)$ ,  $\log p_\theta(x^i) = \mathbb{E}_z [\log p_\theta(x^i | z)] - D_{\text{KL}}(q_\phi(z | x^i) || p_\theta(z)) + D_{\text{KL}}(q_\phi(z | x^i) || p_\theta(z | x^i))$ . Red is intractable, use  $\geq 0$  to ignore it; Orange is reconstruction loss, clusters similar samples; Purple makes posterior close to prior, adds cont. and interp. Orange - Purple is ELBO, maximize it.

$x \xrightarrow{\text{enc}} \mu_{z|x}, \Sigma_{z|x} \xrightarrow{\text{sample}} z \xrightarrow{\text{dec}} \mu_{x|z}, \Sigma_{x|z} \xrightarrow{\text{sample}} \hat{x}$  Backprop through sample by reparametr.:  $z = \mu + \sigma \epsilon$ . For inference, use  $\mu$  directly.

Disentanglement: features should correspond to distinct factors of variation. Can be done with semi-supervised learning by making  $z$  conditionally independent of given features  $y$ .

2.1  $\beta$ -VAE

Disentangle by  $\max_{\theta, \phi} \mathbb{E}_x [\mathbb{E}_{z \sim q_\phi} \log p_\theta(x | z)]$  s.t.  $D_{\text{KL}}(q_\phi(z | x) || p_\theta(z)) < \delta$ , with KKT:  $\max \text{Orange} - \beta \text{Purple}$ .

3 Autoregressive generative models

Autoregression: use data from the same input variable at previous time steps

Discriminative:  $P(Y | X)$ , generative:  $P(X, Y)$ ,

maybe with  $Y$  missing. Sequence models are generative: from  $x_i \dots x_{i+k}$  predict  $x_{i+k+1}$ .

Tabular approach:  $p(\mathbf{x}) = \prod_i p(x_i | \mathbf{x}_{<i})$ , needs  $2^{i-1}$  params. Independence assumption is too strong. Let  $p_{\theta_i}(x_i | \mathbf{x}_{<i}) = \text{Bern}(f_i(\mathbf{x}_{<i}))$ , where  $f_i$  is a NN. **Fully Visible Sigmoid Belief Networks**:  $f_i = \sigma(\alpha_0^{(i)} + \alpha^{(i)} \mathbf{x}_{<i}^T)$ , complexity  $n^2$ , but model is linear.

**Neural Autoregressive Density Estimator**: add hidden layer.  $\mathbf{h}_i = \sigma(\mathbf{b} + \mathbf{W}_{\cdot, <i} \mathbf{x}_{<i})$ ,  $\hat{x}_i = \sigma(c_i + \mathbf{V}_i \cdot \mathbf{h}_i)$ . Order of  $\mathbf{x}$  can be arbitrary but fixed. Train by max log-likelihood in  $O(TD)$ , can use 2nd order optimizers, can use **teacher forcing**: feed GT as previous output.

Extensions: Convolutional; Real-valued: conditionals by mixture of gaussians; Order-less and deep: one DNN predicts  $p(x_k | x_{i_1} \dots x_{i_j})$ .

**Masked Autoencoder Distribution Estimator**: mask out weights s.t. no information flows from  $x_d \dots$  to  $\hat{x}_d$ . Large hidden layers needed. Trains as fast as autoencoders, but sampling needs  $D$  forward passes.

**PixelRNN**: generate pixels from corner, dependency on previous pixels is by RNN (LSTM). **PixelCNN**: also from corner, but condition by CNN over context region (perceptive field)  $\Rightarrow$  parallelize. For conditionals use masked convolutions. Channels: model R from context, G from R + cont., B from G + R + cont. Training is parallel, but inference is sequential  $\Rightarrow$  slow. Use conv. stacks to mask correctly.

NLL is a natural metric for autoreg. models, hard to evaluate others.

**WaveNet**: audio is high-dimensional. Use dilated convolutions to increase perceptive field with multiple layers.

AR does not work for high res images/video, convert the images into a series of tokens with an AE: Vector-quantized VAE. The codebook is a set of vectors.  $x \rightarrow z \xrightarrow{\text{enc}} \text{codebook} \xrightarrow{\text{dec}} z_q \rightarrow \hat{x}$ .

We can run an AR model in the latent space.

3.1 Attention

$\mathbf{x}_t$  is a convex combination of the past steps, with access to all past steps. For  $X \in \mathbb{R}^{T \times D}$ :  $K = XW_K, V = XW_V, Q = XW_Q$ . Check pairwise similarity between query and keys via dot product: let attention weights be  $\alpha = \text{Softmax}(QK^T / \sqrt{D})$ ,  $\alpha \in \mathbb{R}^{1 \times T}$ . Adding mask

$M$  to avoid looking into the future:

$$X = \text{Softmax} \left( \frac{(XW_Q)(XW_K)^T}{\sqrt{D}} + M \right) (XW_V)$$

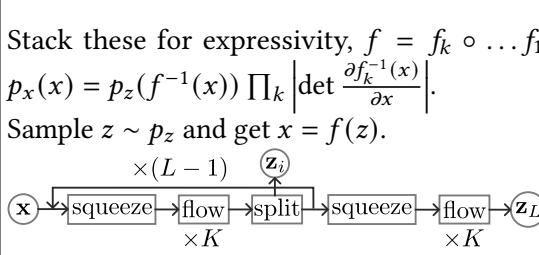
Multi-head attn. splits  $W$  into  $h$  heads, then concatenates them. Positional encoding injects information about the position of the token. Attn. is  $O(T^2D)$ .

4 Normalizing Flows

VAs dont have a tractable likelihood, AR models have no latent space. Want both. Change of variable for  $x = f(z)$ :  $p_x(x) = p_z(f^{-1}(x)) \left| \det \frac{\partial f^{-1}(x)}{\partial x} \right| = p_z(f^{-1}(x)) \left| \det \frac{\partial f(z)}{\partial z} \right|^{-1}$ . Map  $Z \rightarrow X$  with a deterministic invertible  $f_\theta$ . This can be a NN, but computing the determinant is  $O(n^3)$ . If the Jacobian is triangular, the determinant is  $O(n)$ . To do this, add a coupling layer:

$$\begin{pmatrix} y^A \\ y^B \end{pmatrix} = \begin{pmatrix} h(x^A, \beta(x^B)) \\ x^B \end{pmatrix}, \text{ where } \beta \text{ is any model, and } h \text{ is elementwise.}$$
$$\begin{pmatrix} x^A \\ x^B \end{pmatrix} = \begin{pmatrix} h^{-1}(y^A, \beta(y^B)) \\ y^B \end{pmatrix}, J = \begin{pmatrix} h' & h'\beta' \\ 0 & 1 \end{pmatrix}$$

Stack these for expressivity,  $f = f_k \circ \dots \circ f_1$ .  $p_x(x) = p_z(f^{-1}(x)) \prod_k \left| \det \frac{\partial f_k^{-1}(x)}{\partial x} \right|$ . Sample  $z \sim p_z$  and get  $x = f(z)$ .



- Squeeze: reshape, increase chan.
- ActNorm: batchnorm with init. s.t. output  $\sim \mathcal{N}(0, I)$ .  $y_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$ ,  $\mathbf{x}_{i,j} = (y_{i,j} - \mathbf{b})/\mathbf{s}$ ,  $\log \det = H \cdot W \cdot \sum_i \log |\mathbf{s}_i|$ : linear.
- $1 \times 1$  conv: permutation along channel dim. Init  $\mathbf{W}$  as rand. orthogonal  $\in \mathbb{R}^{C \times C}$  with  $\det \mathbf{W} = 1$ .  $\log \det = H \cdot W \cdot \log |\det \mathbf{W}|$ :  $O(C^3)$ . Faster:  $\mathbf{W} := \mathbf{P}(\mathbf{L} + \text{diag}(\mathbf{s}))$ , where  $\mathbf{P}$  is a random fixed permut. matrix,  $\mathbf{L}$  is lower triang. with 1s on diag.,  $\mathbf{U}$  is upper triang. with 0s on diag.,  $\mathbf{s}$  is a vector. Then  $\log \det = \sum_i \log |\mathbf{s}_i|$ :  $O(C)$
- Conditional coupling: add parameter  $\mathbf{w}$  to  $\beta$ .

**SRFlow**: use flows to generate many high-res images from a low-res one. Adds affine injector between conv. and coupling layers.  $\mathbf{h}^{n+1} = \exp(\beta_{\theta,s}^n(\mathbf{u})) \cdot \mathbf{h}^n + \beta_{\theta,b}(\mathbf{u})$ ,  $\mathbf{h}^n = \exp(-\beta_{\theta,s}^n(\mathbf{u})) \cdot (\mathbf{h}^{n+1} - \beta_{\theta,b}^n(\mathbf{u}))$ ,  $\log \det = \sum_{i,j,k} \beta_{\theta,s}^n(\mathbf{u}_{i,j,k})$ .

**StyleFlow**: Take StyleGAN and replace the network  $\mathbf{z} \rightarrow \mathbf{w}$  (aux. latent space) with a normalizing flow conditioned on attributes.

**C-Flow**: condition on other normalizing flows: multimodal flows. Encode original image  $\mathbf{x}_B^1$ :  $\mathbf{z}_B^1 = f_\phi^{-1}(\mathbf{x}_B^1 | \mathbf{x}_A^1)$ ; encode extra info (image, segm. map, etc.)  $\mathbf{x}_A^2$ :  $\mathbf{z}_A^2 = g_\theta^{-1}(\mathbf{x}_A^2)$ ; generate new image  $\mathbf{x}_B^2$ :  $\mathbf{x}_B^2 = f_\phi(\mathbf{z}_B^1 | \mathbf{z}_A^2)$ .

Flows are expensive for training and low res. The latent distr. of a flow needn't be  $\mathcal{N}$ .

5 Generative Adversarial Networks (GANs)

Log-likelihood is not a good metric. We can have high likelihood with poor quality by mixing in noise and not losing much likelihood; or low likelihood with good quality by remembering input data and having sharp peaks there.

**Generator**  $G : \mathbb{R}^Q \rightarrow \mathbb{R}^D$  maps noise  $z$  to data, **discriminator**  $D : \mathbb{R}^D \rightarrow [0, 1]$  tries to decide if data is real or fake, receiving both gen. outputs and training data. Train  $D$  for  $k$  steps for each step of  $G$ .

Training GANs is a min-max process, which are hard to optimize.  $V(G, D) = \mathbb{E}_{\mathbf{x} \sim p_d} \log(D(\mathbf{x})) + \mathbb{E}_{\hat{\mathbf{x}} \sim p_m} \log(1 - D(\hat{\mathbf{x}}))$

For  $G$  the opt.  $D^* = p_d(\mathbf{x}) / (p_d(\mathbf{x}) + p_m(\mathbf{x}))$ . Jensen-Shannon divergence (symmetric):  $D_{\text{JS}}(p||q) = \frac{1}{2} D_{\text{KL}}(p||\frac{p+q}{2}) + \frac{1}{2} D_{\text{KL}}(q||\frac{p+q}{2})$ . Global minimum of  $D_{\text{JS}}(p_d||p_m)$  is the glob. min. of  $V(G, D)$  and  $V(G, D^*) = -\log(4)$ .

If  $G$  and  $D$  have enough capacity, at each update step  $D$  reaches  $D^*$  and  $p_m$  improves  $V(p_m, D^*) \propto \sup_D \int_x p_m(\mathbf{x}) \log(-D(\mathbf{x})) d\mathbf{x}$ , then  $p_m \rightarrow p_d$  by convexity of  $V(p_m, D^*)$  wrt.  $p_m$ . These assumptions are too strong.

If  $D$  is too strong,  $G$  has near zero gradients and doesn't learn ( $\log'(1 - D(G(z))) \approx 0$ ). Use gradient ascent on  $\log(D(G(z)))$  instead.

Model collapse:  $G$  only produces one sample or one class of samples. Solution: **unrolling** - use  $k$  previous  $D$  for each  $G$  update.

DCGAN: pool  $\rightarrow$  strided convolution, batchnorm, no FC, ReLU for  $G$ , LeakyReLU for  $D$ .

Wasserstein GAN: different loss, gradients don't vanish. Adding gradient penalty for  $D$

stabilizes training. Hierarchical GAN: generate low-res image, then high-res during training. StyleGAN: learn intermediate latent space  $\mathcal{W}$  with FCs, batchnorm with scale and mean from  $\mathcal{W}$ , add noise at each layer.