CNN

Linear $T(\alpha\vec{u}+\beta\vec{v})=T(\alpha\vec{u})+T(\beta\vec{v})$ *

Invariant to f $T(f(u))=T(u)$

Equivariant to f $T(f(u))=f(T(u))$ *

☆ $I'(i,j)=\sum_{m=-k}^{k}\sum_{n=-k}^{k}K(m,n)I(i+m,j+n)$

* $I'(i,j)=\sum\sum I(i-m,j-n)K(m,n)$ ♡

* **Commutative** $(I*K)(i,j)=\boxed{(K*I)(i,j)}$

☆ ≡ * iff $K(m,n)=K(-m,-n)$

Discrete Convolution as Matrix Mult $I*K=\begin{bmatrix}K_1 & 0 & \cdots & 0\\ K_2 & K_1 & & \vdots\\ K_3 & K_2 & & \\ \vdots & K_3 & & \\ 0 & \vdots & \cdots & K_m\end{bmatrix}\begin{bmatrix}I_1\\ I_2\\ \vdots\\ I_n\end{bmatrix}$

Fwd $Z^{[l]}_{i,j}=\sum_m\sum_n W^{[l]}_{m,n}Z^{[l-1]}_{i-m,j-n}+b$

Bwd $\delta^{[l-1]}_{i,j}=\partial\mathcal{L}/\partial Z^{[l-1]}_{i,j}\overset{\leftarrow}{=}$ θ update w.r.t. $W^{[l]}_{m,n}$

$\sum_{i'}\sum_{j'}\left(\partial\mathcal{L}/\partial Z^{[l]}_{i',j'}\right)\left(\partial Z^{[l]}_{i',j'}/\partial Z^{[l-1]}_{i,j}\right)$

$=\sum_{i'}\sum_{j'}\delta^{[l]}_{i',j'}W^{[l]}(m,n)$

$=\delta^{[l]}*ROT_{180}(W^{[l]})$

CNN Backprop Example Remember flip K for *

$\underset{X}{\begin{bmatrix}0&0&1&1&0\\0&1&0&1&1\\1&1&1&1&0\\1&0&1&1&0\\1&1&0&0&1\end{bmatrix}}\quad\underset{K}{\begin{bmatrix}1&0&1\\0&1&0\\0&1&0\end{bmatrix}}\quad\underset{Y=ReLU(X*K)}{\begin{bmatrix}3&3&3\\4&2&3\\2&3&3\end{bmatrix}}\rightarrow\underset{Pool(Y)}{\begin{bmatrix}4\end{bmatrix}}$

$\underset{\partial E/\partial Y'}{\begin{bmatrix}1\end{bmatrix}}\rightarrow\underset{\partial E/\partial Y}{\begin{bmatrix}0&0&0\\1&0&0\\0&0&0\end{bmatrix}}\rightarrow\underset{\partial E/\partial K}{\begin{bmatrix}1&0&1\\1&1&1\\0&1&0\end{bmatrix}}\quad\underset{\partial E/\partial X}{\begin{bmatrix}0&0&0&0&0\\0&1&0&0&0\\0&1&0&0&0\\1&0&1&0&0\\0&0&0&0&0\end{bmatrix}}$

Use θ-update and Backward Rule where $\partial E/\partial X=\partial E/\partial Y * ROT_{180}(K)$ $=\partial E/\partial Y \star K$. Slide K over $\partial E/\partial Y$

For $\partial E/\partial K=\frac{\partial E}{\partial Y}\star X=ROT_{180}(X*\partial E/\partial Y)$ ROT_{180}(Slide $\partial E/\partial Y$ over X)

Max-Pooling Fwd: $Z^{[l]}_{i,j}=\max\{Z^{[l-1]}_{i}\}$

Bwd: $\delta^{[l-1]}=\{\delta^{[l]}\}_{i^*,j^*}$ since $\partial Z^{[l]}/\partial Z^{[l-1]}=\mathbb{I}\{i=i^*\}$ No learnable param

$\begin{bmatrix}1&1&2&4\\5&6&7&8\\3&2&1&0\\1&2&3&4\end{bmatrix}\rightarrow\begin{bmatrix}6&8\\3&4\end{bmatrix}$ Max pool, 2x2 filter, Stride 2

Semantic Seg. Semantic Class label $\boxed{\forall \text{pixels}}$

Upsampling Increase Spatial feature size

Nearest Neighbor, Bed of Nails, Max-Unpool

Transposed Convolutions 1. $Z=S-1$; $p'=k-p-1$

2. Insert Z rows between rows & columns

3. Add p' number of zeros around image

4. Perform convolution, Remember Flip K!

U-Net FCNN, Combine global and local feature maps by Copying corresponding tensors from earlier stages in each upsampling Stage. → Captures global, local Ctxt, → acc↑

Res.Conn. Help maintain local features as imgs not completly downsampled every Stage

Parameters Let I: length of input vol size, F: length of filter size, P: amount of zero-PAD, S: Stride, then O: output size

$$O=\left\lfloor\frac{I-F+P_{start}+P_{end}}{S}\right\rfloor+1$$

often $P_{start}=P_{end}\triangleq P$

Complexity	In	Out	Params
CONV: $F\times F\times C\times K$	$I\times I\times C$	$O\times O\times K$	$(F\times F\times C+1)\cdot K$
POOL: F max	$I\times I\times C$	$O\times O\times C$	ZERO
FCNN	N_{in}	N_{out}	$(N_{in}+1)\times N_{out}$

K: how many filters

RNN

$\vec{o}_t$ $\vec{a}_t$

$\hat{y}_t=\mathrm{Softmax}(Vh_t+c)$, $\vec{h}_t=\tanh(Wh_{t-1}+Ux_t+b)$

$\mathcal{L}_t=-\sum_k y_{t,k}\log\hat{y}_{t,k}$ $\quad\mathcal{L}=\sum_t\mathcal{L}_t$

$\frac{\partial\mathcal{L}}{\partial o_t}=-\sum_k\frac{\partial\mathcal{L}_t}{\partial\hat{y}_{t,k}}\frac{\partial\hat{y}_{t,k}}{\partial o_t}$ Compute element-Wise drop t for brevity

$\frac{\partial\hat{y}_k}{\partial o_i}=\frac{\partial}{\partial o_i}\left(\frac{\exp(o_k)}{\sum_j^k\exp(o_j)}\right)$, $[i=k]=\hat{y}_i(1-\hat{y}_i)$, $[i\neq k]=-\hat{y}_i\hat{y}_k$

$=\frac{y_i}{\hat{y}_i}\hat{y}_i(1-\hat{y}_i)-\sum_{i\neq k}\frac{y_k}{\hat{y}_k}\hat{y}_k\hat{y}_i=\hat{y}_i-y_i$

$\frac{\partial\mathcal{L}}{\partial h_t}=\sum_{t'}^{T}\frac{\partial\mathcal{L}_{t'}}{\partial h_{t'}}=\frac{\partial\mathcal{L}_t}{\partial h_t}+\sum_{t'=t+1}^{T}\frac{\partial\mathcal{L}_{t'}}{\partial h_t}$

$\frac{\partial\mathcal{L}_{t'}}{\partial h_t}=\frac{\partial\mathcal{L}_{t'}}{\partial h_{t'}}\prod_{k=t}^{t'-1}\frac{\partial h_{k+1}}{\partial h_k}\Leftrightarrow \mathrm{diag}(1-\tanh^2(a_{k+1}))W$

$\frac{\partial\mathcal{L}_t}{\partial V}=\sum_t\frac{\partial\mathcal{L}_t}{\partial\vec{o}_t}\frac{\partial\vec{o}_t}{\partial V}\leftrightarrow\sum_i\frac{\partial\mathcal{L}_t}{\partial o_{t,i}}\frac{\partial o_{t,i}}{\partial V}$

$o_{t,i}=v_{i1}\cdot h_{t,1}+\cdots+v_{im}\cdot h_{t,m}+\vec{c}$

$\Rightarrow\partial o_{t,i}/\partial v_{kl}=h_{tl}\,\mathbb{I}\{i=k\}=M$

$K\times m$ zero except $M_{i,:}$ so $=\vec{h}_t\,\partial\mathcal{L}_t/\partial\vec{o}_t$

$\frac{\partial\mathcal{L}_t}{\partial W}=\frac{\partial\mathcal{L}_t}{\partial o_t}\frac{\partial o_t}{\partial h_t}\frac{\partial h_t}{\partial W}\leftrightarrow\sum_{k=1}^{t}\frac{\partial h_t}{\partial h_k}\frac{\partial^+ h_k}{\partial W}$

$\frac{\partial h_t}{\partial h_k}=\prod_{t'=k+1}^{t}\frac{\partial h_{t'}}{\partial h_{t'-1}}=\prod_{t'=k+1}^{t}\mathrm{diag}(1-h_{t'}^2)W$

Vanishing / Exploding Gradient $W=Q\Lambda Q^T$

$(W^T)^{t-k-1}=(Q^T\Lambda Q)^{t-k-1}=Q^T\Lambda^{t-k-1}Q$

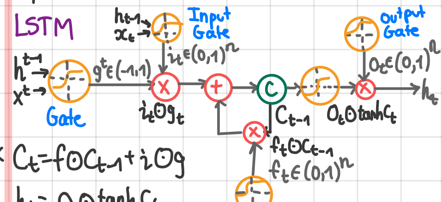
$\Rightarrow\|\mathrm{diag}(f'(h_{i-1}))\|<\gamma$ $\quad f=\sigma$ OR tanh

$\lambda_1<\frac{1}{\gamma}$ vanish, $\lambda_1>\frac{1}{\gamma}$ explodes as $t\to\infty$

$\lambda_1<\frac{1}{\gamma}:\left\|\frac{\partial h_i}{\partial h_{i-1}}\right\|\leq\|W^T\|\|\mathrm{diag}(f'(h_{i-1}))\|<\frac{1}{\gamma}\gamma<1$

Let $\eta\in\mathbb{R}$ s.t. $\left\|\prod_{i=1}^{t}\frac{\partial h_i}{\partial_{i-1}}\right\|<\eta^{t-k}\to 0$ as $t\to\infty$ by induction

LSTM



$C_t=f\odot C_{t-1}+i\odot g$

$h_t=o\odot\tanh C_t$

$\begin{pmatrix}i\\f\\o\\g\end{pmatrix}=\begin{pmatrix}\mathrm{Sigm}\\\mathrm{Sigm}\\\mathrm{Sigm}\\\mathrm{Sigm}\end{pmatrix}W^{[l]}\begin{pmatrix}h_t^{[l-1]}\\h_{t-1}^{[l]}\end{pmatrix}$ helps with Vanishing ∇: not multiply with itself but element-wise and add.

Gradients can still explode in LSTM

Gradient Clipping $\hat{g}\leftarrow\partial\mathcal{L}/\partial\theta$ When $\hat{g}$ explodes

$\theta\leftarrow\theta-\lambda\hat{g}T/\|\hat{g}\|$ if $\|\hat{g}\|>T$

GENERATIVE MODELS

Implicit	Explicit
• MC (GSN)	• Tractable [Autoregressive, NF]
• Direct	• Approximate
-GAN	-MCMC (Boltzmann Machine)
	-Variational (VAE, Diffusion)

AUTOENCODERS

Z Efficient Data Representation, Desirable: Continuity & Interpolation

Original Space $X\underset{g}{\overset{f}{\rightleftarrows}}$ Latent Space Z

$[g\circ f]$ Approximates $\mathbb{I}(\cdot)$ on the data

Linear AE finds Linear embedding /code/ Z

NonLinear AE $\hat{\theta}_f,\hat{\theta}_g=\arg\min_{\theta_f,\theta_g}\mathcal{L}$ where $\mathcal{L}=\sum_{n=1}^{N}\|x_n-\hat{x}_n\|^2=\sum_{n=1}^{N}\|x_n-g(f(x_n))\|^2$

Undercomplete $\dim(z)<\dim(x)$ ⊕ to extract features to train, ⊖ Out of distribution Samples

OverComplete $\dim(z)>\dim(x)$ Perf Reconstr.

Denoising AE $\mathcal{L}$ evaluated based on Comp. w/ Clean img. Learns transform. to rm. noise

AE ⊖ Lack of Continuity in Z, Struggles to generate high quality samples. ⊕ Good for reconstructions.

VARIATIONAL AUTOENCODERS

Objective $p_\theta(x)=\int_z p(x|z)p(z)dz$. Intra.

Approx. posterior $p_\theta(z|x)$ with $q_\phi(z|x)$

$\log p_\theta(x)=\mathbb{E}_{z\sim q_\phi(z|x)}[\log p(x)]$

$=\mathbb{E}_z\left[\log\frac{p(x|z)p(z)}{p(z|x)}\right]=\mathbb{E}_z\left[\log\frac{p(x|z)p(z)q(z|x)}{p(z|x)q(z|x)}\right]$

$=\mathbb{E}_z[\log p(x|z)]-\mathbb{E}_z\left[\log\frac{q(z|x)}{p(z)}\right]+$ $\mathbb{E}_z[\log q(z|x)/p(z|x)]$ $\quad ELBO_{VAE}$

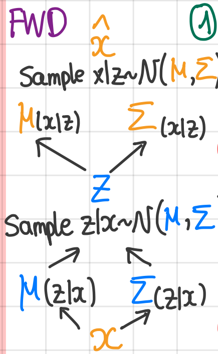
$\propto\underbrace{\mathbb{E}_z[\log p(x|z)]}_{1}-\underbrace{KL(q(x|z)\|p(z))}_{2}+$

Generate new Samples by max data likelihood

$\log p_\theta(x_i)\geq\mathcal{L}(x_i,\theta,\phi)$

$\theta^*,\phi^*=\arg\max_{\theta,\phi}\sum_{i=1}^{N}\mathcal{L}(x_i,\theta,\phi)$

FWD



① Maximize Reconstr. likelihood, encourages Samples of Same Category to be close in Z. $\boxed{0 \text{ Var}\equiv AE}$

② Make approx Similar to prior encourages encoder to project latent space repr. evenly around Center of latent space. At runtime, only use decoder network. $z\sim p(z)$.

$KL\geq 0$ $-KL=-\int p(x)\log\frac{p(x)}{q(x)}dx$ $=\int p(x)\log\frac{q(x)}{p(x)}dx\leq\log\int_x p(x)\frac{q(x)}{p(x)}=0$

Reparam Trick (To allow VAE backprop) Instead of $z\sim N(\mu,\sigma^2)$: $z=\mu+\sigma\varepsilon$ $\varepsilon\sim N(0,I)$

VAE-Backprop $\mathbb{E}_z[\log p(x|z)]-KL(q(z|x)\|p(z))=$

$\mathbb{E}_z[\log p(x|z)]+\mathbb{E}_z\left[\log\frac{p(z)}{q(z|x)}\right]$

$=\mathbb{E}_z\left[\log\frac{p_\theta(x,z)}{q_\phi(z|x)}\right]$ taking gradient w.r.t θ,φ

$=\nabla_{\theta,\phi}\mathbb{E}_{z\sim q_\phi}\left[\log\frac{p_\theta(x,f(x,\varepsilon,\theta))}{q_\phi(f(x,\varepsilon,\theta)|x)}\right]$

$=\mathbb{E}_\varepsilon\left[\nabla_{\theta,\phi}\log\frac{p_\theta(x,f(x,\varepsilon,\theta))}{q_\phi(f(x,\varepsilon,\theta)|x)}\right]$

$\overset{*}{\approx}\frac{1}{K}\sum_{i=1}^{K}\nabla_{\theta,\phi}\log(p_\theta(x,f)/q_\phi(f|x))$

*** Monte-Carlo Gradient Estimator**

$\nabla_\varphi\mathbb{E}_{p_\varphi(z)}[f(z)]=\nabla_\varphi\int p_\varphi(z)f(z)dz$

f differentiable. Given $z=g(\varepsilon,\varphi)$ Via Reparam Trick and Change of Variables $p(z)|dz|=p(\varepsilon)|d\varepsilon|$ so $\int p_\varphi(z)f(z)dz=\int p(\varepsilon)f(z)d\varepsilon=$ $p(\varepsilon)f(g(\varepsilon,\varphi))d\varepsilon$ Hence

$\mathbb{E}_{p(\varepsilon)}[\nabla_\varphi f(g(\varepsilon,\varphi))]=$

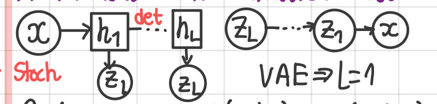
$\nabla_\varphi\int p(\varepsilon)f(g(\varepsilon,\varphi))d\varepsilon=\nabla_\varphi\mathbb{E}_{p(\varepsilon)}[f(g(\varepsilon,\varphi))]$

$=\frac{1}{S}\sum_{s=1}^{S}[\nabla_\varphi f(g(\varepsilon^{(s)},\varphi))]$ $\varepsilon^{(s)}\sim p(\varepsilon)$

Analytical KL $\forall p(z)=N(\mu_p,\sigma_p^2I)$ and $q(z)=N(\mu_q,\sigma_q^2I)$. $\int p(z)\log(q(z))dz$

$=-\frac{J}{2}\log(2\pi)-\frac{1}{2}\sum_{j=1}^{J}\log\sigma_{q,j}^2-$ $\frac{1}{2}\sum_{j=1}^{J}(\sigma_{p,j}^2+(\mu_{p,j}-\mu_{q,j})^2)/\sigma_{q,j}^2$

Hierachical Latent Variable Models



VAE ⇒ L=1

$q_\phi(z_1,\ldots,z_L|x)=q(z_1|x)\cdots q_\phi(z_L|x)$

$p_\theta(z_1,\ldots,z_L)=p_\theta(z_L)p_\theta(z_{L-1}|z_L)\cdots p_\theta(z_1|z_2)$

$\mathcal{L}(\theta,\phi;x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x)\|p_\theta(z))$

$= \mathbb{E}_{q_\phi(z_1,\dots,z_L|x)}[\log p_\theta(x|z_1,\dots,z_L)]$

$- KL(q_\phi(z_1,\dots,z_L|x)\|p_\theta(z_1,\dots,z_L))$

$= \mathbb{E}_{q_\phi(z_1|x)}[\log p_\theta(x|z_1)]$

$- KL(q_\phi(z_1,\dots,z_L|x)\|p_\theta(z_1,\dots,z_L))$

$= \langle$ Only looking at 2nd Term $\rangle + \int q_\phi(z_1,\dots,z_L|x)$ $\log(p_\theta(z_1,\dots,z_L)/q_\phi(z_1,\dots,z_L|x))dz_1,\dots,z_L$

$= \int \prod_{i=1}^L q_\phi(z_i|x)\log(\prod_{i=1}^L p_\theta(z_i|z_{i+1})/q_\phi(z_i|x))dz_1\cdots dz_L$

$= \sum_{i=1}^L \int q_\phi(z_{i+1}|x)\,q_\phi(z_i|x)\log\frac{p_\theta(z_i|z_{i+1})}{q_\phi(z_i|x)}dz_{i,i+1}$

$= KL(q_\phi(z_L|x)\|p_\theta(z_L)) - \sum_{i=1}^{L-1}$ $\mathbb{E}_{z_{i+1}\sim q_\phi(z_{i+1}|x)}[KL(q_\phi(z_i|x)\|p_\theta(z_i|z_{i+1}))]$

β-VAE Learn disentangled repr. unsup. (distinct factors of variation)

$p(\vec{x}|\vec{z}) \approx p(\vec{x}|\vec{v},\vec{w})$, Cond. Ind $V \in \mathbb{R}^K$

Cond. dep, $w \in \mathbb{R}^H$ $\log p(\vec{v}|\vec{x}) = \sum_k \log p(v_k|\vec{x})$

$z \in \mathbb{R}^M$ $M \geq K$. Objective

$\max_{\phi,\theta} \mathbb{E}_{x\sim D}[\mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]]$

Subject to $KL(q_\phi(z|x)\|p_\theta(z)) < \delta$

Via KKT: $\mathcal{F}(\theta,\phi,\beta) = \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(\vec{x}|\vec{z})]$

$-\beta[KL(q_\phi(\vec{z}|\vec{x})\|p_\theta(\vec{z})) - \delta]$ $\beta,\delta \geq 0$

$\geq \mathbb{E}_z[\log p_\theta(\vec{x}|\vec{z})] - \beta KL(q_\phi(\vec{z}|\vec{x})\|p_\theta(\vec{z}))$

$\mathcal{L}_{Beta}(\phi,\beta) = -\mathbb{E}_z[\log p_\theta(\vec{x}|\vec{z})]$ $\beta=1$: VAE

$+\beta KL(q_\phi(\vec{z}|\vec{x})\|p_\theta(\vec{z}))$ $\beta>1$ Strong Constr. on z

AUTOREGRESSIVE MODELS

Autoregression for timeseries $x_t = b_0 + b_1x_{t-1} + b_2x_{t-2}$ uses same input variable at previous timesteps.

Tabular $p(\vec{x}) = \prod_{i=1}^n p(x_i|\vec{x}_{<i})$

⊕ Represents any possible distribution of n RV ⊖ $\theta = (\sum_{i=1}^n 2^{i-1}) \in O(2^{n-1})$ param

Independence Assump. $p(\vec{x}) = p(x_1)\dots p(x_n)$

⊕ n params ⊖ "Random Samply unrelated"

Conditionals w/ fixed number of params

$\forall$ position i; Learn $f_i: \{0,1\}^{i-1} \mapsto [0,1]$

Params: $\sum_{i=1}^n |\theta_i|$, if we carefully set dimensions $\forall\{\theta_i\}_{i=1}^n$ controllable params.

Fully Visible Sigmoid Belief Network

$f_i(x_1,\dots,x_{i-1}) = \sigma(\alpha_0^i + \dots + \alpha_{i-1}^i x_{i-1})$ LR

Params $= \sum_{i=1}^n i = (n^2+n)/2$, stack layers

Neural Autoregressive Density Estimator

AE-like NN to learn $p(x_i = 1|\vec{x}_{<i})$

$\vec{h}_i = \sigma(\vec{b} + W[:,<i]\,\vec{x}_{<i})$ first $<i$ columns

$\hat{x}_i = \sigma(c_i + \vec{V}[i,:]\,\vec{h}_i)$, ith row

Each Conditional Modeled by same NN

Train maximizing $\frac{1}{T}\sum_{t=1}^T \log(p(\vec{x}_t))$

$= \frac{1}{T}\sum_{t=1}^T\sum_{i=1}^N \log p(x_i^{(t)}|\vec{x}_{<i}^{(t)})$

Ground Truth are used for conditioning

⊕ Compute $\in O(TD)$, could make use of 2nd order optimizers, extendable to other observ.

Masked AE Distribution Estimator

Constrain AE s.t. outputs can be used as conditionals $p(x_i|\vec{x}_{<i})$ ⊕ Training has same complexity as AE, criteria is NLLK for bin $\vec{x}$, $p(\vec{x})$ is just fwd.

(but sampling D fwd, very large h_t needed. Complexity at inference larger than AE)

PixelCNN Masked Convolutions for Conditionals

Train Maximizes likelihood of train imgs

$p(\vec{x}) = \prod_{i=1}^n p(x_i|x_1,\dots,x_{i-1})$

Predicts $\forall$ pixels, a distribution $0\to255$

Masking $p(x_i|\vec{x}_{<i}) = p(x_{i,R}|\vec{x}_{<i})\cdot$ $p(x_{i,G}|x_{i,R},\vec{x}_{<i})\cdot p(x_{i,B}|x_{i,R},x_{i,G},\vec{x}_{<i})$

Stacking Layers of masked convolutions creates a blindspot $\to$ Use 2 stacks of convolution [vertical, horizontal stack]

WaveNet Use dialated convolution to increase receptive field without increasing num parameters for Audio.

Self-Attention $\vec{x}_t = f(\vec{x}_{t-1},\vec{x}_{t-2},\dots,\vec{x}_1)$

$W_K, W_Q, W_V \in \mathbb{R}^{D\times D}$ $\vec{x} \in \mathbb{R}^D$ (learnable)

$K = XW_K$, $V = XW_V$ $Q = XW_Q$

$\vec{\alpha} = \text{Softmax}(QK^T/\sqrt{D}) \in \mathbb{R}^{1\times T}$

$X = \text{Softmax}(QK^T/\sqrt{D} + M)\,V$

Transformer MHSA Split attention into multiple heads s.t. each head calcul. a chunk of the representation.

P.E Inject sinusoidal encodings unique to indices to preserve input ordering. $= \sin(w_k\cdot t)$ if $i = 2k$ $\cos(w_k\cdot t)$ o/w, $w_k = 10000^{-2k/d}$

Encoder is Autoregressive

Complexity	Per layer	Seq ops	max path len
Self-Attention	$O(n^2d)$	$O(1)$	$O(1)$
Recurrent	$O(nd^2)$	$O(n)$	$O(n)$
Convolutional	$O(knd^2)$	$O(1)$	$O(\log_k n)$
SA Restricted	$O(rnd)$	$O(1)$	$O(n/r)$

VRNN $p_\theta(\vec{z}) = \prod_{t=1}^T p_\theta(z_t|\vec{z}_{<t},\vec{x}_{<t})$

$q_\phi(\vec{z}|\vec{x}) = \prod_{t=1}^T q_\phi(z_t|x_t,\vec{x}_{<t},\vec{z}_{<t})$

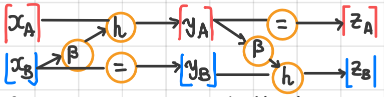
$h_t = f_\theta(\vec{x}_t,\vec{z}_t,h_{t-1})$, $p_\theta(\vec{x},\vec{z}) = p_\theta(\vec{x}|\vec{z})p_\theta(\vec{z})$

$= \prod_{t=1}^T p_\theta(z_t|\vec{z}_{<t},\vec{x}_{<t})\,p_\theta(x_t|z_t,\vec{z}_{<t},\vec{x}_{<t})$

$\mathcal{L}(\theta,\phi;x) = \mathbb{E}_{q_\phi(z_t|x_{\leq t},z_{<t})}[\log p_\theta(x_t|z_t,z_{<t},x_{<t})]$

$-KL(q_\phi(z_t|x_t,x_{<t},z_{<t})\|p_\theta(z_t|z_{<t},x_{<t}))$

NORMALIZING FLOWS



f: Invertible, differentiable, dim preserving

$f_\theta: \mathbb{R}^d \to \mathbb{R}^d$, $X = f_\theta(Z)$, $Z = f_\theta^{-1}(X)$

Model $P_X(\vec{x},\theta) = P_Z(f^{-1}(\vec{x}))\left|\det\left(\frac{\partial f^{-1}(\vec{x})}{\partial \vec{x}}\right)\right|$

Training Maximize exact Log-Likelihood $\log P_X(D) = \sum_{x\in D}(\log P_Z(f^{-1}(x)) + \sum_K \log|\det \partial f(z)/\partial z|^{-1}$

Inference to generate $\vec{x}$: $\vec{z}\sim p_{\vec{z}} \Rightarrow \vec{x} = f(\vec{z})$

Probability of $\vec{x}$: $\vec{z} = f^{-1}(\vec{x}) \Rightarrow p_{\vec{z}}(\vec{z})$

Planar NF $\vec{x} = f(\vec{z}) = \vec{z} + \vec{u}h(\vec{w}^T\vec{z} + \vec{b})$

$p_{\vec{z}}(f^{-1}(\vec{x})) = p_{\vec{z}}(\vec{z}) \propto \exp(-\vec{z}^2/2)$

$\det(\partial f(\vec{z})/\partial\vec{z}) = \det(I + h'\vec{u}\vec{w}^T)$

$= (1 + h'\vec{w}^T\vec{u}) \Rightarrow \log p(\vec{x}) = -\frac{\vec{z}^2}{2} - \log(1 + h'\vec{u}^T\vec{w})$

Conditional Coupling NF

$p(\vec{x}|c) = p(\vec{z}|c)|\det(\partial f(\vec{z})/\partial\vec{z})|^{-1}$

$f(\vec{z}) = \begin{bmatrix}\vec{x}_0\\ \vec{x}_1\end{bmatrix} = \begin{bmatrix}\vec{z}_0\\ \tau(\vec{z}_0) + \vec{z}_1\odot\sigma(A\vec{z}_0 + \vec{b})\end{bmatrix}$

$\vec{z} = [\vec{z}_0,\vec{z}_1]$, $\vec{x} = [\vec{x}_0,\vec{x}_1]$ $\vec{f} = [f_1,f_2]$

$\frac{\partial f(\vec{z})}{\partial\vec{z}} = J_f(\vec{z}) = \begin{bmatrix}\partial f_1(\vec{z})/\partial\vec{z}_0 & \partial f_1(\vec{z})/\partial\vec{z}_1\\ \partial f_2(\vec{z})/\partial\vec{z}_0 & \partial f_2(\vec{z})/\partial\vec{z}_1\end{bmatrix}$

$\det(\partial f(\vec{z})/\partial\vec{z}) = \prod_{j=1}^{D-d_0}\sigma(A\vec{z}_0 + \vec{b})_j$

Squeeze Reduce Spatial Dim, preserves dimensionality. One per level. $4\times4\times1 \to 2\times2\times4$

StepOfFlow 1. ActNorm 2. Invertible 1X1 Convolution 3. C. Coupling layer

1. Data-dependent Initialization

↳ Run one iter, init layer $\sim N(0,I)$

Fwd: $\forall i,j$, $\vec{y}_{ij} = \vec{s}\odot\vec{x}_{i,j} + \vec{b}$

Rev: $\forall i,j$, $\vec{x}_{ij} = (\vec{y}_{i,j} - \vec{b})/\vec{s}$

Log-det: $H\cdot W\cdot$ sum$(\log|\vec{s}|)$

2. Permutation in the Channel dimension

$\vec{h}\in\mathbb{R}^{H\times W\times C}$ W orthogonal $\det(\vec{W}) = 1 \in \mathbb{R}^{C\times C}$

Compute in $O(C^3)$ $\log|\det(\frac{\partial}{\partial h}\text{conv}(h;W)|$

$= H\cdot W\cdot\log|\det(W)|$ Parameterize $W = PL(U + \text{diag}(\vec{s}))$ L: diag$=1$ U: diag$=0$

in $O(C)$ $\log|\det(W)| = \text{sum}(\log(\vec{s}))$

SRFlow Conditioning on low-res image

1,2, Affine Injector A pre-trained CNN encodes a low-res img $\vec{u} = g_\theta(\vec{x})$

Activation map h_n modulated with spatial feature maps $\vec{u}$ $\forall$ channels, locations

Fwd: $h^{(n+1)} = \exp(\beta^n_{\theta,s}(\vec{u}))\cdot h^{(n)} + \beta_{\theta,b}(u)$

Inv: $h^{(n)} = \exp(-\beta^n_{\theta,s}(\vec{u}))\cdot(h^{(n-1)} - \beta^n_{\theta,b}(u))$

Log-Det: $\sum_{i,j,k}\beta^n_{\theta,s}(u)_{ijk}$

StyleFlow NF for Disentanglement

C-Flow Two Flows, One conditioned on other

$\to$ Multimodal img-img mapping, style transfer, img manip, 3D point cloud recon.

Flow B: Compute transform. params. | Flow A

GENERATIVE ADVERSERIAL NETWORKS

$\mathcal{L}(D) = -\frac{1}{2N}(\sum_{i=1}^N y^{(i)}\log D(x^{(i)}) + \sum_{N+1}^{2N}(1-y^{(i)})$ $\log(1 - D(x^{(i)})))$ Via BCE loss

$D^* = \arg\min_D -\frac{1}{2}(\mathbb{E}_{x\sim p_d}[\log D(x)]$ $+\mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))])$ for a fixed G

Objective $\min_G\max_D V(D,G)$ highlighted

$D^*_G(x) = \arg\max_D \mathbb{E}_{x\sim p_d}[\log D(x)] +$ $\mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))] =$

$\int_x p_d(x)\log(D(x))dx + \int_z p_z(z)\log(1 - D(G(z)))dz$

$= \int_x p_d(x)\log(D(x)) + p_g(x)\log(1 - D(x))dx$

$= p_d(x)/(p_d(x) + p_g(x))$ [optimal $D = 0.5$]

$V(G,D^*)$

$= \mathbb{E}_{x\sim p_d}\left[\log\frac{2\,p_d(x)}{2\,p_d(x) + p_g(x)}\right] + \mathbb{E}_{x\sim p_g}\left[\log\left(\frac{2\,p_g(x)}{2(p_d(x) + p_g(x))}\right)\right]$

$= KL(p_d(x)\|\frac{p_d(x) + p_g(x)}{2}) + KL(p_g(x)\|\frac{p_d(x) + p_g(x)}{2})$

$-\log4 = 2JS(p_d(x)\|p_m(x)) - \log4$

$G^* = \arg\min_G V(G,D^*) \wedge \forall x: JS(\cdot) \geq 0 = -\log(4)$

Convergence G, D enough Capacity $\wedge$ at each step D allowed to reach D^*, and p_g is updated to improve $\sup_D \int_x p_g(x)\log(1 - D(x))dx$

$\Rightarrow p_g$ converges to p_d (directly opt p_g not θ)

Training Alternate between Gradient-**Ascent** on D $\max_{\theta_d}\mathbb{E}_{x\sim p_d(x)}[\log D_{\theta_d}(x)]$ $+\mathbb{E}_{z\sim p_z(\cdot)}[\log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$

and Gradient **Ascent** on G $\max_{\theta_g}\mathbb{E}_{z\sim p_z}$ $[\log(D_{\theta_d}(G_{\theta_g}(z)))]$ $\ast$ $\log(1 - D(G(z)))$ Saturates in early stages.

ModeCollapse G $\to$ High Quality, Low Variability

Sol: Unrolled GANs: After k updates, G is optimized w.r.t. D after next k steps.

↳ Discourages G to exploit local min.

JS-Issues Correlates badly with sample quality (don't know when to stop train)

D tends to be optimal, JS saturates (when no overlap supports) → Bad Gradients

Sol: Wasserstein Distance

Gradient Penalty w.r.t gradients of D

$\lambda\|\nabla_x D(x)\|^2$ Stabilizes GAN training

DCGAN Replace pooling layers w/ strided conv [D], fractional-strided conv [G] batch Norm [D,G], Remove fully connected h for deeper architectures, ReLU expect Tanh on last layer [G], Leaky ReLU ∀L [D]

MenHat − Men NoHat + Women NoHat = WomenWHat

StyleGAN pass $\vec z$ through a series of FCNN. mapping $\vec z \to \vec w$, insert $\vec w$ multiple times w/ (adnorm). Learn params in first layer) layerwise style and noise injection

Pix2Pix $\mathcal{L}(G,D)=\mathcal{L}_{cGAN}(G,D)+\lambda\mathcal{L}_{L1}(G)$ where

$\mathcal{L}_{cGAN}(G,D)=\mathbb{E}_{x,y}[\log D(x,y)]+\mathbb{E}_{x,z}[1-\log D(x,G(x,z))]$

$\mathcal{L}_{L1}(G)=\mathbb{E}_{x,y,z}[\|y-G(x,z)\|_1]$ Img Translation

CycleGAN Unpaired Image Translation

$\mathcal{L}(G,F,D_x,D_y)=\mathcal{L}_{GAN}(G,D_y,X,Y)+\mathcal{L}_{GAN}(F,D_x,Y,X)$

$+\lambda\mathcal{L}_{cyc}(G,F)$ where $\mathcal{L}_{cyc}=\mathbb{E}_{x\sim p_d}[\|F(G(x))-x\|_1]$ $+\mathbb{E}_{y\sim p_d}[\|G(F(y))-y\|_1]$

DIFFUSION MODELS

$X_T\sim N(0,1)$

real img $X_0\sim q(X_0)$; $q(x_0)$ original data distr.

Diffusion Step $\boxed{q(x_t|x_{t-1})}=\mathcal{N}(x_t;\sqrt{1-\beta_t}\,x_{t-1};\beta_t I)$

$0<\beta_1<\dots<\beta_T<1$ $x_t=\sqrt{1-\beta_t}\,x_{t-1}+\sqrt{\beta_t}\,\varepsilon$ Iterative

Direct Let $\alpha_t=1-\beta_t$; $\bar\alpha_t=\prod_{i=1}^t\alpha_i$

$x_t=\sqrt{\alpha_t}\,x_{t-1}+\sqrt{1-\alpha_t}\,\varepsilon=\sqrt{\alpha_t\alpha_{t-1}}\,x_{t-2}+\sqrt{1-\alpha_t\alpha_{t-1}}\,\varepsilon$

$=\dots=\sqrt{\bar\alpha_t}\,x_0+\sqrt{1-\bar\alpha_t}\,\varepsilon$; $\boxed{q(x_t|x_0)}=\mathcal{N}(\sqrt{\bar\alpha_t}\,x_0,(1-\bar\alpha_t)I)$

linear schedular img noise too quick → use cosine

Denoising Step $q_\theta(x_t)=\int q(x_t|x_0)q(x_0)dx_0$ Intract.!

$p_\theta(x_{t-1}|x_t)=\mathcal{N}(x_{t-1};\mu_\theta(x_t,t),\sigma_t I)\approx q(x_{t-1}|x_t,x_0)$

Objective $\log p(x)=\log\int p(x_{0:T})dx_{1:T}$

$=\log\int p(x_{0:T})\frac{q(x_{1:T}|x_0)}{q(x_{1:T}|x_0)}dx_{1:T}=\log\mathbb{E}_q\left[\frac{p(x_{0:T})}{q(x_{1:T}|x_0)}\right]$

$\geq\mathbb{E}_q\left[\log\frac{p(x_{0:T})}{q(x_{1:T}|x_0)}\right]=\text{ELBO}=\underbrace{\mathbb{E}_q[\log p_\theta(x_{0:T})]}_{\text{Reconstruct } \mathcal{L}_0}-$ VAE $\mathcal{L}$

$\underbrace{KL(q(x_T|x_0)\|p(x_T))}_{\text{Prior } \mathcal{L}_T}-$

$\underbrace{\sum_{t=2}^{T}\mathbb{E}_q[KL(q(x_{t-1}|x_t,x_0)\|p_\theta(x_{t-1}|x_t))]}_{\text{Denoising } \mathcal{L}_t}$

$\arg\min_\theta\mathcal{L}_t=\arg\min_\theta KL(q(x_{t-1}|x_t,x_0)\|p_\theta(x_{t-1}|x_t))$

$=\arg\min_\theta KL(\mathcal{N}(\mu_q,\Sigma_q(t))\|\mathcal{N}(\mu_\theta,\Sigma_q(t)))$

$=\arg\min_\theta\frac12\left[\log\frac{|\Sigma_q(t)|}{|\Sigma_q(t)|}-d+tr(\Sigma_q(t)^{-1}\Sigma_q(t))\right.$

$\left.+(\mu_\theta-\mu_q)^T\Sigma_q(t)^{-1}(\mu_\theta-\mu_q)\right]=\arg\min_\theta\frac{1}{2\sigma_q^2(t)}\|\mu_\theta-\mu_q\|_2^2$

$\mu_q(x_t,x_0)=\frac{1}{\sqrt{\alpha_t}}x_t-\frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}\sqrt{\alpha_t}}\varepsilon_0$

$\mu_\theta(x_t,x_0)=\frac{1}{\sqrt{\alpha_t}}x_t-\frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}\sqrt{\alpha_t}}\underbrace{\hat\varepsilon_\theta(x_t,t)}_{\text{NN}}$ (1)

(1) Learns to predict total added noise $\varepsilon_0\sim\mathcal{N}(0,I)$ that was used from x_0 to x_t. Pred ε_0 better > x_0 or μ_q

$\|\varepsilon-\varepsilon_\theta(x_t,t)\|^2-\|\varepsilon-\varepsilon_\theta(\sqrt{\bar\alpha_t}x_0+\sqrt{(1-\bar\alpha_t)}\varepsilon,t)\|^2$

Train While ¬Conv. $[x_0\sim q(x_0); t\sim\mathcal{U}(\{1,\dots,T\})$

$t\sim\mathcal{N}(0,I)$; GD Step $\nabla_\theta\|\varepsilon-\varepsilon_\theta(\sqrt{\bar\alpha_t}x_0+\sqrt{1-\bar\alpha_t}\varepsilon x_0,t)\|^2$

Sampling $x_t\sim\mathcal{N}(0,I)$; for $(t=T,\dots,1)$ $[z\sim\mathcal{N}(0,1)$ if $t>1$ else 0; Given $\sigma_t=\beta_t^2$:

$X_{t-1}=\frac{1}{\sqrt{\alpha_t}}\left(x_t-\frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\underbrace{\varepsilon_\theta(x_t,t)}_{\text{Predicted Noise}}\right)+\sigma_t z$

Reverse Distribution $q(x_{t-1}|x_t,x_0)\propto$ Bayes

$\boxed{q(x_t|x_{t-1},x_0)}\boxed{q(x_{t-1}|x_0)}\propto$

$\exp\{-(x_t^2-2x_t\sqrt{1-\beta_t}\,x_{t-1}+(1-\beta_t)x_{t-1}^2/2\beta_t-$

$(x_{t-1}^2-2x_{t-1}\sqrt{\bar\alpha_t}\,x_0+\bar\alpha_t x_0^2)/2(1-\bar\alpha_t)\}$

$\sigma_q^2=(1-\beta_t)/\beta_t+1/(1-\bar\alpha_{t-1})$

$\mu_q=\sqrt{1-\beta_t}\,x_{t-1}\sigma_q^2/\beta_t+\sqrt{\bar\alpha_t}\,x_0\sigma_q^2/(1-\bar\alpha_t)$

Since $q(x_{t-1}|x_t,x_0)\sim\mathcal{N}(x_{t-1};\mu_q^2,\sigma_q^2)\propto$

$\exp\{(x_{t-1}-\mu_q)^2/2\sigma_q^2\}$

$=\exp\{(\underbrace{x_{t-1}^2}_{a^2}\underbrace{-2x_{t-1}\mu_q}_{-2ab}+\underbrace{\mu_q^2}_{b^2})/2\sigma_q^2\}$

NEURAL IMPLICIT REPRESENTAIONS

Voxels $O(n^3)$ Memory, Resolution Limited

Points discretize surface → 3D pts, No connectivity

Meshes discretize → Vertices, faces, approx error

Implicit functions Learns analytic f which represents the 3D-Surface. No approx error ⇒ Smooth Continuous. No graphical visual

hard to obtain high frequency details

Level-Set of a continuous function to repr. surfaces $f(x)=x_1^2+x_2^2+x_3^2-r^2$

$S=\{x|f(x)=0\}$ (prob inside S ↓)

Occupancy Networks $f_\theta:\mathbb{R}^3\times\mathcal{X}\mapsto[0,1]$

DeepSDF $f_\theta:\mathbb{R}^3\times\mathcal{X}\mapsto\mathbb{R}$ [−in, +out]

NIR Implicit ⇒ Arbitary topology, resolu. Low Memory footprint. Learning from:

Watertight Meshes 1. Randomly sample 3D points in space 2. Query GT occupancy or SDF from GT meshes 3. Train with CE.

Point Clouds (Given) $\mathcal{X}=\{x_i\}_{i\in I}\subset\mathbb{R}^3$

Compute θ, $f_\theta(x)$ SDF of $\mathcal{M}$ defined by $\mathcal{X}$ w/o any additional supervised data.

Weak Supervision ① want $\mathcal{L}$ vanish train pts

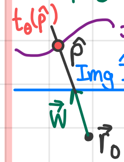
$\mathcal{L}(\theta)=\sum_{i\in I}\underbrace{|f_\theta(x_i)|^2}_{①}+\lambda\underbrace{\mathbb{E}_x[\|\nabla_x f_\theta(x)\|-1]^2}_{②}$

② Want Spatial gradient = 1 ⇒ interpret as geometric surface, encourages smoothness

Ray Marching FIRST zero-crossing of SDF

Secant Method $x_n=x_{n-1}+f(x_{n-1})\frac{x_{n-1}-x_{n-2}}{f(x_{n-1})-f(x_{n-2})}$

DVR fwd

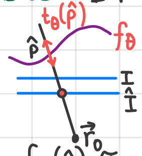


∀pixels u, find $\hat p$ along $\vec w$ Via ray marching & Root finding

Evaluate texture field $t_\theta(\hat p)$

Insert Colour $t_\theta(\hat p)$ at pixel u

Bwd $\hat I$: pred. $\mathcal{L}=\sum_u\|\hat I_u-I_u\|$



$\frac{\partial\mathcal{L}}{\partial\theta}=\frac{\partial\mathcal{L}}{\partial I_u}\frac{\partial I_u}{\partial\theta}$

$\frac{\partial\hat I_u}{\partial\theta}=\frac{\partial t_\theta(\hat p)}{\partial\theta}+\frac{\partial t_\theta(\hat p)}{\partial\hat p}\cdot\frac{\partial\hat p}{\partial\theta}$

$f_\theta(\hat p)=\tau$ with $\hat p=r_0+\hat d w$

$\frac{\partial f_\theta(\hat p)}{\partial\theta}+\frac{\partial f_\theta(\hat p)}{\partial\hat p}\frac{\partial\hat p}{\partial\theta}=0$ $\quad\frac{\partial\hat p}{\partial\theta}=w\frac{\partial\hat d}{\partial\theta}$

Solve for $\frac{\partial\hat d}{\partial\theta}$, then multiply w back.

$\frac{\partial\hat p}{\partial\theta}=-w\left(\frac{\partial f_\theta(\hat p)}{\partial\hat p}\cdot w\right)^{-1}\frac{\partial f_\theta(\hat p)}{\partial\theta}$ Analytic No storing

NEural RadienceField Can model transparency → worse geometry

Input x,y,z,θ,ϕ (Camera parameters)

Output σ, c (density, RGB value of point)

$x,y,z\to256\to\uparrow\to256\xrightarrow{\sigma}128\to$ RGB
(prevent washout) x,y,z $\quad\theta,\phi$

don't want σ to depend on θ, ϕ just x, y, z

Alpha Compositing $\delta_i=t_{i+1}-t_i$

$\alpha=1-\exp\{-\sigma_i\delta_i\}$

Transmittence $T_i=\prod_{j=1}^{i-1}(1-\alpha_j)$ $\quad C=\sum_{i=1}^{N}\underbrace{T_i\alpha_i}_{\text{weight}}C_i$

Treat w as probability for new samples

All points used for Volumetric rendering

Train $\min_\theta\sum_i\|render_i(F_\theta)-I_i\|^2$

Use PE to map continuous input coordinates into higher-dim space ⇒ MLP easily approx higher frequency function.

⊖ 50+ Callibrated views needed, slow rendering speed for high-res imgs, only models static scenes.

Sparse Views → Regularize depth by smoothness, image by likelihood.

FastNerF Replace MLP w/ feature grid + shallow MLP, represent grid w/ hash table for memory efficiency

Gaussian Splatting hard to model thin structures w/ isotropic shapes (sphere) → Non-isotropic parameterization (ellipsoid) 3D Gaussians to 2D Space Gaussians (splats)

1. ∀pixels (approx by tile-based sorting, sort all Gaussians according to their depth.

2. Calculate opacity of each Gaussian $\alpha=o\cdot\exp(-0.5(x-\mu')^T\Sigma'^{-1}(x-\mu'))$

3. Use Volume Rendering Equation $C=T_1\alpha_1C_1+T_2\alpha_2C_2+T_3\alpha_3C_3$ $\quad T_1=1$ $\quad T_2=1-\alpha_1$, $T_3=(1-\alpha_2)(1-\alpha_1)$ $\quad T_i=\prod_{j=1}^{i-1}(1-\alpha_j)$

Parameterized via Spherical harmonics

$Y_0^0(\theta,\phi)=\frac12\sqrt{\frac1\pi}$, $Y_1^{-1}(\theta,\phi)=\sqrt{\frac{3}{4\pi}}\sin\phi\sin\theta\dots$

$Y_1^0(\theta,\phi)=\sqrt{\frac{3}{4\pi}}\cos\theta$ $\quad Y_1^1(\theta,\phi)=\sqrt{\frac{3}{4\pi}}\cos\phi\sin\theta$

$x=r\sin\phi\sin\theta$ $\quad y=r\cos\phi\sin\theta$ $\quad z=r\cos\theta$

When $\theta=\pi/2$, $z=0$ [3D→2D] denote r as a

Y_0^0 Constant r → Circle w/ 0 Center

$Y_1^{-1}=\sqrt{\frac{3}{4\pi}}\frac{x}{r}$ $\quad Y_1^0=\sqrt{\frac{3}{4\pi}}\frac{z}{r}$ $\quad Y_1^1=\sqrt{\frac{3}{4\pi}}\frac{y}{r}$

Using absolute val of Y_m^ℓ as new radius

$x^2+y^2+z^2=a^2x^2/r^2$ $\quad x^2+y^2+z^2=a^2y^2/r^2$

$x^2+y^2=|ax|$ and $x^2+y^2=|ay|$

Circle center at $|\frac a2|$ $\forall x>0$ $-|\frac a2|<0$

The Y_m^ℓ define spheres in 3D, where Y_1^0 aligned with the z-axis.

PARAMETRIC BODY MODELS

Pictorial Structure Model

$S(I,L)=\underbrace{\sum_{i\in V}\alpha_i\cdot\phi(I,\ell_i)}_{①}+\underbrace{\sum_{i,j\in E}\beta_{ij}\cdot\psi(\ell_i,\ell_j)}_{②}$

① Unary term: how likely on image I will body joint i be located at ℓ_i

② Pairwise term: likelihood body joint i is linked with joint j

Direct Regression Based on deep CNN to directly regress x, y coordinates involving a refiner

Heatmaps Improve human pose estimation with occlusions. Each keypoint has a seperate binary heatmap (option of $\mathcal{N}(\mu,\sigma)$ around it)

2D→3D Lift directly, no restrictions 2m Arm

SMPL Represent via 3D mesh, V~7000

Body defined by Shape and Pose ①

Shape PCA of meshes in canonical pose to estimate directions of maximum shape variation and obtain a low-dimensional subspace (10D-300D) in the canonical pose.

Pose Linear Blend Skinning (4)

Each vertex t_i in rest position is transformed $t_i' = \sum_{k=1}^K w_{ki} G_k(\theta, J) t_i$

w_{ki}: Blend skinning weights by artist

G_k: Rigid bone transform

J: joint locations θ: Pose

Posed vertices are linear combination of transformed template vertices.

⊖ Produces well known artifacts

↳ Augment base shape: Pose blend shape is a vector of vertex displacements in rest pose. SMPL: Learn from data:

$t_i' = \sum_k w_{ki} G_k(\theta, J(\beta))(t_i + s_i(\beta) + p_i(\theta))$

Joints $J(\beta)$ depends on shape β (PCA)

Shape correctives $s(\beta) = S\beta$ Can be NN

Pose correctives $p(\theta) = P\theta$

$B_S(\beta, S) = \sum_{n=1}^{|S|} \beta_n S_n$ (2)

$\beta = [\beta_1, \dots, \beta_{|\beta|}]^T$ linear shape coefficients

$S_n \in \mathbb{R}^{3N}$ orthonormal PC of shape displace

$B_P(\theta, P) = \sum_{n=1}^{9K} (R_n(\theta) - R_n(\theta^*)) P_n$ (3)

23 Joints, K=3, $R(\theta)$ length 23×9

$P_n \in \mathbb{R}^{3N}$ vector of vertex displacements

$P = [P_1, \dots, P_{9K}] \in \mathbb{R}^{3N \times 9K}$ all 207 B_P

$M(\vec\beta, \vec\theta; \Phi)$ Mesh $J(\vec\beta; J, \bar T, S)$

Update Rule $\theta^{t+1} \leftarrow \theta^t - \lambda\left(\frac{\partial L_{reproj}}{\partial\theta} + \frac{\partial L_{prior}}{\partial \theta}\right)$

⊖ Hand-crafted optimization routine

⊖ Sensitive to initialization

⊖ Slow convergence

target 2D pose

Learned Gradient Descent

$\theta^{t+1} = \theta^t + \underset{\text{NN}}{F}(\underset{\text{actual grads}}{\partial L_{reproj}/\partial\theta}, \underset{\text{curr state}}{\theta^t, x})$

	Reconstruction	Human Detail	hard-poses
Regression-based		✓	✗
Template-based		✓ Needs Temp.	✓

Animatable Neural implicit Surfaces

1. Input Posed 3D Meshes
2. Learned Continuous Canonical Shape and skinning weights
3. Continuous Implicit Surfaces → unseen poses.

REINFORCEMENT LEARNING

Markov Property $P(S_{t+1}|S_t) = P(S_{t+1}|S_1,\dots,S_t)$

Markov Chain $\langle S,P\rangle$ $P_{ss'} = P[S_{t+1}=s'|S_t=s]$

Markov Reward Process $\langle S,P,R,\gamma\rangle$

$R = \mathbb{E}[R_{t+1}|S_t=s]$ $\gamma \in (0,1)$

Return G_t total discounted reward from t. $G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^\infty \gamma^k R_{t+k+1}$

Markov Decision Process $\langle S,A,P,R,\gamma\rangle$

Bellman $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t=s]$

$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t=s] = \sum_a \pi(a|s) \sum_{s'}\sum_r p(s',r|s,a)[r+\gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1}=s']]$

$= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)(r + \gamma v_\pi(s'))$

Non-linear, No closed form

Action-Value $q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t=s, A_t=a]$

$= \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1},A_{t+1})|S_t=s,A_t=a]$

Bellman Optimality $v_*(s) = \max_\pi v_\pi(s)$

$= \max_a q_*(s,a) = \max_a \sum_{s',r} p(s',r|s,a)(r+\gamma v_*(s'))$

$\pi'(s) = \arg\max_{a\in A}(r(s,a) + \gamma V_\pi(P(s,a)))$

$V_{\pi'}(s) \geq V_\pi(s)$

Value Iter $V^*_{new}(s) = \max_{a\in A}\{r(s,a) + \gamma V^*_{old}(s)\}$

Policy Iter. Initialize π_0

Until Convergence: iterates while S ⊖

- Compute $V^\pi(x)$ $\forall x$
- Compute Greedy Policy π_G w.r.t V^π
- Set $\pi \leftarrow \pi_G$

Finds exact solution in polynomial # iterations! $\pi^* \in O^*(n^2 m/(1-\gamma))$

Guaranteed to monotonically improve

⊖ needs to know transition matrix

MonteCarlo Sampling Run episodes w/ given π and compute samples of $\sum_{i=0}\gamma^i r(s_{t+i}, a_{t+i})$ of $V_\pi(s_t)$

⊕ Unbiased estimate, no need system dynam

⊖ high variance, explore/exploit dilemma

need termination state (slow for long episo)

MODEL-FREE RL

OnPolicy Compute Q according to π and follows π. OffPolicy Compute Q according to GREEDY π, follows different exploration

TD Learning $\Delta V(s) = r(s,a) + \gamma V(s') - V(s)$

$V(s) \leftarrow V(s) + \alpha\Delta V(s)$ on-policy

follows π to obtain (x,a,r,x')

SARSA $\Delta Q(S,A) = R + \gamma Q(S',A') - Q(S,A)$

$Q(S,A) \leftarrow Q(S,A) + \alpha \Delta Q(S,A)$

on-policy Use current policy to get the rollout of states, actions, rewards

Q-learning off-Policy policy to Update Q different π to collect data ε

$\Delta Q(S,A) = R_{t+1} + \gamma \max_a \{Q(S',a)\} - Q(S,A)$ $Q(S,A) \leftarrow Q(S,A) + \alpha\Delta Q(S,A)$

⊕ Less Variance than MC due bootstrap

⊕ More Sample efficient than D.P.

⊕ Don't need to know transition M

⊖ Biased due to bootstrapping, using old estimates as lables

⊖ Explore, exploit dilemma

DEEP REINFORCEMENT LEARNING

Deep Q Learning NN to learn (s,a)

$\mathcal{L}(\theta) = (R + \gamma \max_{a'}\{Q_\theta(s',a')\} - Q_\theta(s,A))^2$

Replay Buffer to store generated samples, during updates randomly sample transitions from it. Q-Learning is off-policy so we can use old samples.

⊖ Limited to discrete action spaces

Policy Gradients $\pi(a_t|s_t) = \mathcal{N}(\mu_t, \sigma_t^2|s_t)$

Collect rollout trajectories (explore)

$p(\tau) = p(s_1, a_1, \dots, s_T, a_T) = p(s_1)\prod_{t=1}^T \pi(a_t|s_t) p(s_{t+1}|a_t, s_t)$

Samples action $\forall t$ from $\pi(a_t|s_t)$ [on-policy]

Evaluation $J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}[\sum_t \gamma^t r(s_t, a_t)]$

PolicyUpdate Goal: $\theta^* = \arg\max J(\theta)$

$\theta \leftarrow \theta + \nabla_\theta J(\theta)$ where $J(\theta) = \mathbb{E}_\tau[r(\tau)]$

$= \int p(\tau) r(\tau) d\tau$ so $\nabla_\theta J(\theta) = \int \nabla_\theta p(\tau) r(\tau) d\tau$

$= \int p(\tau)\nabla_\theta \log p(\tau) r(\tau) d\tau$ w/ $\nabla \log f = \frac{\nabla f}{f}$

$= \mathbb{E}_\tau[\nabla_\theta \log p(\tau) r(\tau)]$ $-b(s_t^i)$

$= \mathbb{E}_\tau[\nabla_\theta \sum_{t=0}^T \log \pi_\theta(a_t|s_t) \sum_{t=0}^T \gamma^t r(s_t, a_t)]$

REINFORCE to compute $\mathbb{E}$ in practice

$\theta \leftarrow \theta + \eta\gamma^t G_t \nabla \log\pi(A_t|x_t; \theta)$

Unbiased high variance → Baseline

ACTOR-CRITIC use bootstrapping to esti. r.

$\nabla_\theta J(\theta) = \frac{1}{N}\sum_i \sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t^i|s_t^i)(r(s_t^i, a_t^i) + \gamma V(s_{t+1}^i) - V(s_t^i))$ TD-Error

NEURAL NETWORK BASICS

MLE $\theta^*_{MLE} = \arg\max_\theta P_m(y|X,\theta)$ (1)

$= \arg\max_\theta \prod_{i=1}^N P_m(y_i|x_i, \theta)$ (2)

$= \arg\max_\theta \sum_{i=1}^N \log P_m(y_i|x_i,\theta)$ (3)

Universal Approx Theorem Let $\sigma: \mathbb{R}\to\mathbb{R}$ non-constant, bounded, continuous, $I_m = [0,1]^m$, space of real-valued function on $I_m = C(I_m)$ $\forall f\in C(I_m)$ cont, $\forall\varepsilon>0$, $n\in\mathbb{Z}$, $v_i, b_i \in \mathbb{R}$, $\vec w_i \in \mathbb{R}^m$, $\forall i\in\{1,\dots,N\}$

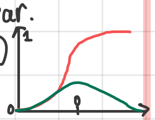
$f(x) \approx g(x) = \sum_{i=1}^N v_i \sigma(w_i^T x + b_i) \wedge |g(x) - f(x)| < \varepsilon, \forall x \in I_m$

Softmax $(\vec z)_i = \exp(z_i)/\sum_{j=1}^K \exp(z_j)$

Activation functions make layer to layer mappings non-linear.

Sigmoid $\sigma(x) = 1/(1+e^{-x})$

$\sigma'(x) = \sigma(x)(1-\sigma(x))$

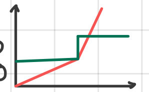


$\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x}) = 2\sigma(2x) - 1$

$\tanh'(x) = 1 - \tanh(x)^2$

ReLU $(x) = \max(0, x)$

$\mathrm{ReLU}'(x) = \mathbb{1}(x>0)$



Leaky ReLU $(x) = \begin{cases}\alpha x & x<0 \\ x & x\geq 0\end{cases}$

Randomized Leaky ReLU

$\alpha \sim \mathrm{Unif}(a,b)$ test $\alpha = \mathbb{E}[\alpha] = (a+b)/2$

Ensemble train different m on same d OR same m on different d and aggregate to decrease variance.

Dropout M share θ, train small % of m

Bagging M independent, train all till conv.

BatchNorm $z_i^{norm} = (z_i - \mu)/\sqrt{\varepsilon + \sigma^2}$

$\tilde z_i = \gamma z_i^{norm} + \beta$ At testing μ, σ

$x \in \mathbb{R}^d$, $y, b \in \mathbb{R}^C$ $W \in \mathbb{R}^{C\times d}$ from layer $l \in \mathbb{R}^n$ to $l \in \mathbb{R}^K$ $(n+1)\cdot K$

PROBABILITY AND MATH

$\frac{\partial}{\partial \vec x}[\vec a^T \vec x] = \frac{\partial}{\partial \vec x}[\vec x^T \vec a] = \vec a$ $\frac{\partial}{\partial \vec x}[\vec b^T A \vec x] = A^T \vec b$

$\frac{\partial}{\partial X}[\vec x^T A] = A^T$ $\frac{\partial}{\partial \vec x}[\vec x \odot \vec a] = \mathrm{diag}(\vec a)$

KL-Div $KL(q\|p) = \int q(\theta)\log\frac{q(\theta)}{p(\theta)}d\theta$

↳ $KL(q\|p) \geq 0$ $KL(q\|p) \neq KL(p\|q)$

JS-Div $JS(q\|p) = \frac12 KL(q\|M) + \frac12 KL(p\|M)$ where $M = \frac{q+p}{2}$

Convolutions $A * ROT_{180} B = A \star B$

$A * B = ROT_{180}(B \star A)$; $A\star B :=$ stride B on A

2D ChangeofVars $\iint_R f(x,y)dxdy = \iint_S f(g(u,v), h(u,v)) J(u,v) du dv$ Let $x=f(z)$ f inv, cont, diff, $dx = |\det(\frac{\partial f(z)}{\partial z})| dz \Rightarrow p_x(x) = p_z(f^{-1}(x))|\det(\frac{\partial f(z)}{\partial z})|$

MatrixDetLemma $\det(A + uv^T) = (1 + v^T A^{-1} u)\det(A)$

Jensen $g(\mathbb{E}[x]) \leq \mathbb{E}[g(x)]$ $g(\cdot)$ convex

Chain Rule $p(x_{1:T}) = p(x_1)p(x_2|x_1)\cdots p(x_T|x_{T-1}\dots x_1)$

Product $p(x,y) = p(y|x)p(x)$

Bayes $p(x|y)p(y) = p(y|x)p(x)$

$p(x_t|x_{t-1}, x_0) = p(x_{t-1}|x_t, x_0)p(x_t|x_0)/p(x_{t-1}|x_0)$