1 Backprop & Training

Cross-entropy: $H(p,q) = -\mathbb{E}_p[\log q]$. **MLE**: $\hat{\theta} = \arg\max_\theta \sum_i \log p(x_i|\theta)$. **KL div**: $\mathrm{KL}(p\|q) = \mathbb{E}_p \log(p/q)$. If $p = \sum_i \delta_{x_i}$ empirical, $H(p,q) = \mathrm{NLL}(q) = -\mathrm{KL}(p\|q) + H(p)$.

2 CNN

Neural Findings (1) Rapid serial visual presentation(RSVP), (2) Hubel & Wiesel, receptive fields (3) HMAX Model, simple/complex cell.
Pre-DL Model: Neurocognitron, LeNet-5, **Dataset**: ImageNet.
3 Properties:
(1) **Linear** $T(\alpha u + \beta v) = \alpha T(u) + \beta T(v)$.
(2) **invariance** to f, $T(f(u)) = T(u)$.
(3) **equivariant** to f, $T(f(u)) = f(T(u))$.

Convolution

F:$z^{(l)}_{ij} = w^{(l)} * z^{(l-1)} + b = \sum_{mn} w^{(l)}_{mn} z^{(l-1)}_{i-m,j-n} + b$.

B on z:$\delta^{(l-1)}_{i,j} := \frac{\partial C}{\partial z^{(l-1)}_{i,j}} = \sum_{i'j'} \frac{\partial C}{\partial z^{(l)}_{i,j}} \frac{\partial z^{(l)}_{ij}}{\partial z^{(l-1)}_{ij}} = \sum_{i'j'} \delta^{(l)}_{ij} w^{(l)}_{i'-l,j'-j} = \delta^{(l)} * \mathrm{ROT}_{180^\circ}(w^{(l)})$.

B on w:$\frac{\partial C}{\partial w^{(l)}_{m,n}} = \sum_{i',j'} \frac{\partial C}{\partial z^{(l)}_{i,j}} \frac{\partial z^{(l)}_{i,j}}{\partial w^{(l)}_{m,n}} = \delta^{(l)} * \mathrm{ROT}_{180^\circ}(z^{(l-1)})$

$L_{\text{out}} = \left\lfloor \frac{L_{\text{in}} + 2\mathrm{Pad}_{=0} - \mathrm{Dilat}_{=1}(\mathrm{Ker}-1) - 1}{\mathrm{Stride}_{=1}} + 1 \right\rfloor$ for Conv and Pool.
n_{connect} of each neuron $3k^2$.
$n_{\text{para}} = (k^2 \dim_{\text{in}} + 1)\dim_{\text{out}}$
Correlation ⇔ Conv only $C_{m,n} = K_{-m,-n}$.
Conv can be seen as Matmul, $I * K = \mathrm{Kl}$,
Differentiation by Conv, $\mathrm{Ker} = (1,-1)$.
Weight sharing btw pixels.

Pooling

Goal (1) increase receptive field. (2) noise robustness.
Max-Pool F: $z^{(l)} = \max\{z^{(l-1)}_i\}$, B: $\delta^{(l-1)} = \{\delta^{(l)}\}_{i^* = \mathrm{argmax}_i\{z^{(l-1)}_i\}}$.

Application

Alexnet Smaller filter, more layers.
GoogLeNet Deeper, Inception module, 1x1 conv for dim/channel reduction64 → 32, Auxiliary c1s head ⇒ More efficient.
ResNet Adv for residual connection: (1) better convergence. (2) propagate high frequency information (U-net).

3 Full CNN, upsampling

Goal Pixel predict, semantic seg.
Unpooling Methods: Nearest Neighbor, Bed of Nail, Max Unpooling (need maxpool from prev net).

ConvTranspose2D

shape: $L_{\text{out}} = (L_{\text{in}} - 1)\mathrm{Stride}_{=1} - 2\mathrm{Pad}_{=0} + \mathrm{Dilat}_{=1}(\mathrm{Ker} - 1) + \mathrm{OutPad}_{=0} + 1$
App U-Net: combine global+local features using tensors from previous layer.

4 RNN

Def $h^t = f(h^{t-1}, x^t; W), \hat{y}^t = W_{hy} h^t$.
Loss $L = \sum_t L^t = \sum_t L(y^t, \hat{y}^t)$.
BPTT $\frac{\partial L^t}{\partial W} = \sum_{k=1}^t \frac{\partial L^t}{\partial \hat{y}^t} \frac{\partial \hat{y}^t}{\partial h^t} \frac{\partial h^t}{\partial h^k} \frac{\partial^+ h^k}{\partial W}$,
$\frac{\partial h^t}{\partial h^k} = \prod_{i=k+1}^t W_h^T \mathrm{diag}[f'(h^{i-1})]$.

Gradient vanishing/exploding

$\lambda_1 := \max \lambda(W_{hh})$, $\gamma > \|\mathrm{diag}(f'(h^{i-1}))\|$, then $\lambda_1 \gtrless \gamma^{-1}$ vanish/explode.
DisAdv for GV/E (1) Instabilities during training lead to Inf/NaN. (2) hard to capture long-term dependencies, cuz no gradient at lower levels, no learning. (3) large gradients⇔jumper over local minima or oscillate.

LSTM

Input, forget, output $\in [0,1]$, gate $\in \mathbb{R}$.
Vanila: $h^l_t = \tanh W^l \begin{pmatrix} h^{l-1}_t \\ h^l_{t-1} \end{pmatrix}$. **LSTM**:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h^{l-1}_t \\ h^l_{t-1} \end{pmatrix}, \quad c^l_t = f \odot c^l_{t-1} + i \odot g, h^l_t = o \odot \tanh\left(c^l_t\right), W^l[4n \times 2n].$$

Gradient clipping→prevent ∇explode.

5 Variational Auto-Encoder

Supervise f maps data $x \to$ label y. **Unsuper** Data's underlying hidden structure: Clustering, feature learning, dim reduc, density estim. **Generative** learn p_{M} sim to $p_{\mathcal{D}}$, then sample fromp_{M}.

Auto-Encoder (AE)

Encoder proj to latent space, **Decoder** proj back. **Loss** $\mathrm{Diff}(x_{\text{in}}, x_{\text{recon}})$. PCA is Linear AE.
APP (1) Denoise AE: randomly set pixels to 0 then recover. (2) Inpainting AE: cover part of image then recover. (3) 3D Human Motion: add noise to skeleton, recover with AE then pass to LSTM per frame.
Comment Reconstruction ✓, Generation ✗. Latent space not well-structured: no continuity, no interpolation.

Variational Auto-Encoder (VAE)

$\ln p_\theta(x) = \mathrm{ELBO}_{\theta,\phi}(x) + \mathrm{KL}(q_\phi(z|x)\|p_\theta(z|x))$.
Objective: $\max_{\theta,\phi} \sum_i \mathrm{ELBO}_{\theta,\phi}(x_i)$.
$\mathrm{ELBO} = \mathbb{E}_{q_\phi} \ln p_\theta(x|z) - \mathrm{KL}(q_\phi(z|x)\|p_\theta(z))$
$\mathbb{E}_{q_\phi}[\log p_\theta(x|z)]$ recon-likelihood, encourage clustering for similar samples in latent space. $-\mathrm{KL}(q_\phi(z|x)\|p_\theta(z))$ latent code loss, make posterior close to prior, encourage latent representations evenly around center.
(1) If only recon: VAE→AE, sharp recon but sparce latent space. (2) No recon loss: compact embedding like Gaussian, but bad for reconstruction.
Encoder: $q_\phi(z|x) = \mathcal{N}(\mu_{\mathrm{nn}}(x), \Sigma_{\mathrm{nn}}(x))$,
Decoder: $p_\theta(x|z) = \mathcal{N}(\mu_{\mathrm{nn}}(z), \Sigma_{\mathrm{nn}}(z))$.
Approximate expectation in loss with sampling.
Reparam trick: $z \sim \mathcal{N}(\mu, \sigma^2) \Leftrightarrow \epsilon \sim \mathcal{N}(0,1), z = \mu + \sigma\epsilon, z = f(x, \epsilon, \theta)$

Disentangle respresentation

semi-superv-learn can lead to disentangle. $p(x|y,z)$, y digits, z style.
UnSupervised disentangle, β-**VAE**:
Assume $p(x|z) \approx p(x|v,w)$, v cond indep, w cond dep, $\log p(v|x) = \sum_k \log p(v_k|x)$.
Train: $\max_{\phi,\theta} \mathbb{E}_{x,q_\phi} \ln p_\theta(x|z)$, s.t. $\mathrm{KL}(q_\phi\|p_\theta(z)) < \delta \Leftrightarrow L_{\beta,\phi,\theta} = \mathrm{NLL} + \beta\,\mathrm{KL}$, $\beta > 1$ stronger constraint on latent space than VAE.

Integrals of Gaussian

$p = \mathcal{N}(\mu, \Sigma), q = \mathcal{N}(m, L)$, (1) $H_p = [D(\ln 2\pi + 1) + \ln|L|]/2$. (2) $H_{p,q} = -\int p \ln q dx = [D \ln 2\pi + \ln|L| + \mathrm{Tr}(L^{-1}\Sigma) + (\mu - m)^\top L^{-1}(\mu - m)]/2$. (3) If $\Sigma = \mathrm{diag}\{\sigma_i^2\}, L = \mathrm{diag}\{l_i^2\}$, $H_{p,q} = (D \ln 2\pi + \sum_{i=1}^D \ln l_j^2 + \sum_{i=1}^D (\sigma_i^2 + (\mu_i - m_i)^2)/l_i^2)/2$. (4) $\mathrm{KL}(p\|q) = [\ln|L|/|\Sigma| + \mathrm{Tr}(L^{-1}\Sigma) - D + (\mu - m)^\top L^{-1}(\mu - m)]/2$.

Hierarchical Latent Variable Models

Encoder $q_\phi(z_{1:L}|x) = \prod_{i=1}^L q_\phi(z_i|x)$, Decoder $p_\theta(x|z) = p_\theta(x|z_1)$, Prior $p_\theta(z_{1:L}) = p_\theta(z_L)\prod_{i=1}^{L-1} p_\theta(z_i|z_{i+1})$, $\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi}[\log p_\theta(x|z_1)] - D_{KL}(q_\phi(z_L|x)\|p_\theta(z_L)) - \sum_{i=1}^{L-1} \mathbb{E}_{z_{i+1}\sim q_\phi(z_{i+1}|x)} D_{KL}(q_\phi(z_i|x)\|p_\theta(z_i|z_{i+1}))$.

6 Autoregressive models

Goal Density estim $p(x) = \prod_i p(x_{i,j,k}|x_{<i})$.

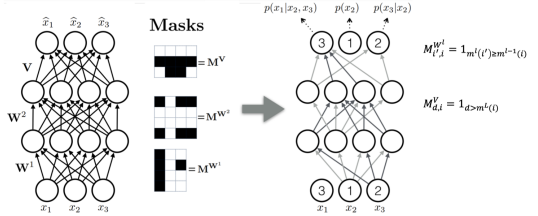
Fully Visible Sigmoid Belief Networks

Tabular approach need 2^L states not acceptable, replace with regression $f_i(x_{1:i-1}) = \sigma(\alpha^i_0 + \sum_{j=1}^{i-1} \alpha^i_j x_j)$,
$p_{\theta_i}(x_i|x_{<i}) = \mathrm{Ber}(f(x_{1:i-1}))$.

Neural Autoreg Density Estim (NADE)

Extend FVSBN with NN, $h_i = \sigma(b + W_{<i}x_{<i}), \hat{x}_i = \sigma(c_i + V_i \mathbf{h}^{(i)})$. Leverage by $(b + W_{<i+1}x_{<i+1}) - (b + W_{<i}x_{<i}) = W_{i+1}x_{i+1}$.
ML $\sum_{t=1}^T \ln p(x^t) = \sum_{t=1,i=1}^{T,D} \ln p(x^{(t)}_i | x^{(t)}_{<i})$. Ground truth fed in conditional term for higher accuracy.
Pros: (1) Comput cost $O(TD)$. (2) second-order optim OK (3) Reals and multinomials OK. (4) Random Order works fine.
Variant: (1) Reals (RNADE) conds are mixt of gaussian. (2) Order-less and deep (DeepNADE): NN to assign conditional distri to variable given subset of the others. (3) ConvNADE

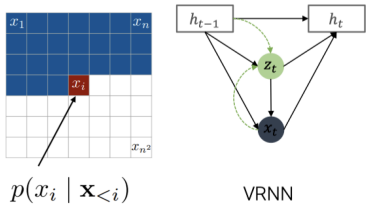
Masked Autoencoder (MADE)



Constrain AE s.t. output fulfill autoreg property: No path between output unit x_d and $x_{d+1:D}$ (relative to some ordering).
Achieve by masking AE⇒each output units network don't take latter as input. During Training, randomly re-mask, similar to dropout.

PixelCNN

PixelRNN: Generate pixels staring from corner. Depend on previous pixels by LSTM. Cons: slow due to sequential generation, explicit pixel dependencies.
PixelCNN: Starting from corner. Dependency by CNN over context region.
Pred $p(x_i|x_{<i}) = p(x_{i,R}|x_{<i})p(x_{i,G}|x_{i,R},x_{<i})\times p(x_{i,B}|x_{i,R},x_{i,G},x_{<i})$
Finite kernel mask Conv →blind spot: actual prediction of pixel does not depend on some region of unmasked part.
Improvement: (1) more expressive nonlinear: $\mathbf{h}_{k+1} = \tanh\left(W_{k,f} * \mathbf{h}_k\right) \odot \sigma(W_{k,g} * \mathbf{h}_k)$. (2) Vertical stack and Horizontal stack for blind spot.
Results: img-like patches no semantics.
Pros: (1) explicit likelihood (2) $p_\theta(\mathcal{D}_{\text{train}})$ good evaluation metric (3) Good samples
Cons: (3) Seq genera is slow



Temporal causal NN (TCN): WaveNet

Idea: Adapt PixelCNN to Audio.
Prob: larger dim than images ($> 16000/s$).
Trick: Dilated Conv, exponential increase in receptive field. (Stride Conv not allow to preserve resolution)

Variational RNN

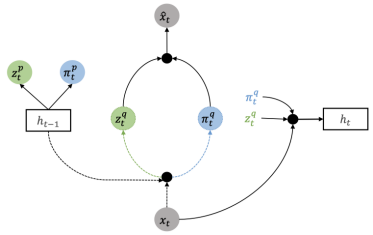
Goal increase expressiveness, noise robustness by stochastic latent variables into hidden state of an RNN.
Model (1) Deterministic transition $h_t = f_\theta(h_{t-1}, x_t, z_t)$. (2) Dynamic prior $p_\theta(z_t|h_{t-1})$, overall prior $p_\theta(z) = \prod_{t=1}^T p_\theta(z_t|z_{<t}, x_{<t}) = \prod_{t=1}^T p_\theta(z_t|h_{t-1})\times p_{f_\theta}(h_{t-1}|h_{t-2}, z_{t-1}, x_{t-1})$. Explicit temporal depend btw timesteps in latent space.
Adv for z in f_θ: (1) robust by randomness of z (2) latent z is high level, more informative on future.
D: $p(x_t|z_t) = g^{\text{out}}(h_{t-1}, z_t)$,
E: $q_\phi(z|x) = \prod_{t=1}^T q_\phi(z_t|x_t, x_{<t}, z_{<t})$,
ELBO $\sum_t \log p_\theta(x_t|x_{<t}) \geq \mathcal{L}_{\theta,\phi} = \sum_{t=1}^T \mathbb{E}_{q_\phi(z_t|x_t,x_{<t},z_{<t})}[\log p_\theta(x_t|z_t, z_{<t}, x_{<t})] - \text{KL}(q_\phi(z_t|x_t, x_{<t}, z_{<t})\|p_\theta(z_t|z_{<t}, x_{<t}))$.
KL in VRNN: (1) balance the informative z_t from x_t dependent q_ϕ (if not x_t depend, z_t mostly from h_{t-1}) and overfit to memorize x_t by KL to prior. (2) Force the prior also to be informative from h_{t-1}.

conditional VRNN

Hidden state $\{z_t, \pi_t\}$. Priors: $p(z_t|h_{t-1}) = g^{p,z}(h_{t-1})$, $p(\pi_t|h_{t-1}) = g^{p,\pi}(h_{t-1})$. Transition: $h_t = \tau(x_t, z_t, \pi_t, h_{t-1})$ D: $p(x_t|z_t, \pi_t) = g^{\text{out}}(z_t, \pi_t)$ (no h_{t-1} here). E: $q(z_t|x_t) = g^{q,z}(h_{t-1}, x_t)$, $q(\pi_t|x_t) = g^{q,\pi}(h_{t-1}, x_t)$.
ELBO: $\sum_{t=1}^T \mathbb{E}_{q(z_t,\pi_t x_t)} \log p(x_t z_t, \pi_t) - \text{KL}(q(z_t x_t)\|p(z_t)) - \text{KL}(q(\pi_t x_t)\|p(\pi_t))$
APP Digital Handwriting Modeling.

Summary

Pros $p(x)$ tractable, easy to train, sample, but slow. **Cons** No natural latent variable representation (except C-VRNN, STCN).



Self-attentioan and Transformers

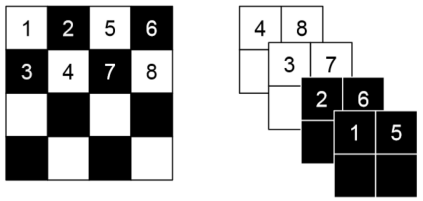
$X = \text{softmax}((W_Q X)(W_K X)^\top/\sqrt{D} + M) \odot W_V X$, mask M prevent accessing future.
Positional Encoding unique sinusoidal encoding to preserve ordering. $\text{PE}_{i,t} = \sin/\cos(\omega_k t)$ for $i = 2k/2k+1$, $\omega_k = 10000^{-2k/d}$, t position in seq, i conponent in embedding vec. Cost $O(T^2 D)$.
Trick multi-head attention.
APP 3D human pose and mesh recon, 3D objects Mesh generation, 3D human motion modelling.

7 Normalizing Flows

Idea $p_{X;\theta}(x) = p_Z(f_\theta^{-1}(x))|\det \partial_x f_\theta^{-1}(x)|$.
Require (1) NN differentiable, invertible, preserve the dim, (2) Jacobian computed efficiently. **Trick** triangular matrix to reduce the complexity of Det to $O(d)$.
Flow Trans: $x = f_k \circ f_{k-1} \circ \cdots f_2 \circ f_1(z)$, $p_x(x) = p_z(f^{-1}(x))\prod_k |\det \partial_x f_k^{-1}(x)|$
ML $\log p_x(\mathcal{D}) = \sum_{x\in\mathcal{D}} \ln p_z(f^{-1}(x)) + \sum_k \ln|\det \partial_x f_k^{-1}(x)|$. **Planner**: $f(z) = z + uh(w^T z + b)$, $|\det \partial_z f| = |1 + h' u^T w|$.
Radial: $f(z) = z + \beta h(\alpha, r)(z - z_0)$, $r = |z - z_0|$, $h(\alpha, r) = (\alpha + r)^{-1}$, $|\det \partial_z f| = [1 + \beta h(\alpha, r)]^{d-1}[1 + \beta h(\alpha, r) + \beta h'(\alpha, r) r]$

Coupling Layer

F $\begin{pmatrix} y^A \\ y^B \end{pmatrix} = \begin{pmatrix} h(x^A, \beta(x^B)) \\ x^B \end{pmatrix}$, R $\begin{pmatrix} x^A \\ x^B \end{pmatrix} = \begin{pmatrix} h^{-1}(y^A, \beta(y^B))) \\ y^B \end{pmatrix}$, Jacob $\begin{pmatrix} h' & h'\beta' \\ 0 & 1 \end{pmatrix}$.
$h(x) = (h(x_1), \ldots, h(x_n))^\top$ element-wise. $\partial_{x^\top} h = \text{diag}(h'(x_1), \ldots, h'(x_n))$.
Continuous NF: model infinite number of trans with proper temporal discretization. Neural ODE.



NF in CV, Flow Block (FB)

In NF, part of output processed, else passed to latter layer. Shuffle needed and different among papers.
Squeeze & split: reduce spacial dimension and distribute them into the channel, splitting in channel. (i, j): spacial idx.

FB1 - activation norm

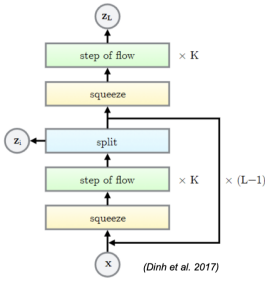
Scale and bias per channel, sim to BN, data depend, trainable. F $y_{i,j} = s \odot x_{i,j} + b$, R $x_{i,j} = (y_{i,j} - b)/s$, LogDet $\text{H} \cdot W \cdot \text{sum}(\log|s|)$.

FB2 - invertible 1x1 conv

F $y_{i,j} = W x_{i,j}$ R $x_{i,j} = W^{-1} y_{i,j}$ LogDet $h \cdot w \cdot \log|\det W|$, $W : [c \times c]$, $O(c^3)$ for $|\det W|$ reduce to $O(c)$ by $W = PL(U + \text{diag}(s))$, $\log|\det W| = \text{sum}(\log|s|)$.

Fb3 - (conditional) coupling layer

F $x_a, x_b = \text{split}(x)$, $(\log s, t) = \text{NN}(x_b)$, $s = \exp(\log s)$, $y_a = s \odot x_a + t$, $y_b = x_b$, $y = \text{concat}(y_a, y_b)$, R $y_a, y_b = \text{split}(y)$, $(\log s, t) = \text{NN}(y_b)$, $s = \exp(\log s)$, $x_a = (y_a - t)/s$, $x_b = y_b$, $x = \text{concat}(x_a, x_b)$, LogDet $\text{sum}(\log(|s|))$.
Conditional operation $\beta(x^B; C)$.



Application

Super-Res Flow learns a distribution of HR variants. Cond on a low-res image. Pretrained CNN encodes a low-resolution image: $u = g_\theta(x)$ inject to actnorm, affects all channels and spatial locations.
F $h^{n+1} = \exp\left(f^n_{\theta,s}(u)\right) \cdot h^n + f_{\theta,b}(u)$, R $h^n = \exp(-f^n\theta, s(u)) \cdot (h^{n+1} - f^n\theta, b(u))$, LogDet $\sum_{ijk} f^n\theta, s(u)_{ijk}$.
StyleFlow(cmp to StyleGAN) Replace mapping network with a continuous normalizing flow, Condition on image attributes.
C-Flow
Conditioning $Flow_A$: Standard affine coupling layer, to images.
Conditioned $Flow_B$: trans param conditioned on, to point cloud, sketch, seg, etc.
Enable (1) (multimodal) image-to-image mapping (2) style transfer (3) image manipulation (4) 3D point cloud reconstruction from images.
APP Probabilistic Modeling for Human Mesh Recovery

Difficalty for geneartive model

Evaluating $\log p(x)$ is hard and usually computationally not tractable.

8 GANs, implicit model, neural sampler, likelihood-free model

Intuition Likelihood not good indicator for generation, can be independent: $\ln[0.01p_{\text{true}} + 0.99q_{\text{noise}}] \geq \ln[0.01p_{\text{true}}] = \ln p(x) - \ln 100$.

Pros (1) Highly expressive, (2) Density function not defined or intractable. **Cons** Lack of theory and learning algo cmp to explicit models.

Idea Two-player game. **Generator** fool the discriminator with real-looking images $G : \mathbb{R}^Q \to \mathbb{R}^D$ from Gaussian noise to images. **Discriminator** distinguish btw real and fake img $D : \mathbb{R}^D \to [0,1]$.

Obj $\min_G \max_D V_{G,D} = \ln D(x) + \ln(1 - D(\hat{x}))$

Optim $D^* = \arg\max_D V_{G,D} = \frac{p_{\text{D}}(x)}{p_{\text{D}}(x)+p_{\text{M}}(x)}$.

Jensen-Shannon Div $\text{JS}(p\|q := [D_{KL}(p\|\frac{p+q}{2}) + D_{KL}(q\|\frac{p+q}{2})]/2$, $\text{JS}(p\|q) = 0$ iff $p_{\text{D}} \equiv p_{\text{M}}$.

$V_{G,D^*} = -\ln 4 + 2\,\text{JS}(p_d(x)\|p_m(x))$, global optim if $p_{\text{D}}(x) \equiv p_{\text{M}}(x)$.

If G, D enough capacity, each step can reach D^*, update p_{M} directly instead of its param, then $V(p_{\text{M}}, D^*)$ is convex, global optim can reach and $\propto \sup_D \mathbb{E}_{p_{\text{M}}(x)} \ln(1 - D(x))dx$.

(1) In practice pptim D in inner-loop is computationally prohibitive and lead to overfitting on finite datasets.

(2) **Aim** keep D near optimum and G changes only slowly: k steps of optim D (typically $k \in \{1,\dots,5\}$), 1 step of optim G with small lr.

(3) $\log(1 - D(G(z)))$ near zero $\to$ smaller $\nabla \Rightarrow$ gradient ascent $\max_G \mathbb{E}_{z\sim p_z(z)}[\ln(D(G(z)))]$.

Issues

1. Difficult to train, Mode collapse (no sample diversity), saddle point in dual energy landscape. Sol: unrolled GAN.

2. Low dim manifolds in high dim prob space have little overlap. D of vanilla GAN saturates if no overlapping support. JS only measures similarity, not 'work' required from p_{M} to p_{D}. Sol: Wasserstein GAN. GAN can be generalized to entire family of div.

Pros and Cons

Pros (1) A wide variety of functions and distributions can be modeled (flexibility). (2) Only backprop when training, no sampling, stochastic optim. (3) No approximation to likelihood required as in VAEs. (4) Samples more realistic than other DGMs.

Cons (1) No explicit p_M. (2) no direct means to evaluate likelihood (3) Careful balancing G and D during training .

Applications

Pix2Pix cond GAN, Obj: $\mathcal{L}(G,D) = \mathcal{L}_{cGAN}(G,D) + \lambda\mathcal{L}_{L1}(G)$, $\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\ln D(x,y)] + \mathbb{E}_{x,z}[1 - \ln D(x, G(x,z))]$, x conditions, $\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1]$ reconstruction loss. requires paired images as training data. Translate samples from two side.

CycleGAN unpaired image translation: Two conditional GANs $\mathcal{L}(G,F,D_X,D_Y) = \mathcal{L}_{GAN}(G,D_Y,X,Y) + \mathcal{L}_{GAN}(F,D_X,Y,X) + \lambda\mathcal{L}_{\text{cyc}}(G,F)$, $L_{cyc} = . = \mathbb{E}_{x\sim p_{\text{D}}(x)}\|\|F(G(x)) - x\|_1] + \mathbb{E}_{y\sim p_{\text{D}}(y)}[\|G(F(y)) - y\|_1]$. Transfer and Transfer back.

GauGAN Given semantic seg and reference style, synthesize image. Problem with pix2pix: Unconditional norm layer in G "washes away" information of semantic labels. Sol: Spatial cond norm $\gamma^i_{c,y,x}(m)\frac{h^i_{n,c,y,x} - \mu^i_c}{\sigma^i_c} + \beta^i_{c,y,x}(m)$

Progressive Growing of GANs: Grow the generator and discriminator resolution by adding layers during training.

StyleGAN

9 Parametric Body Models

Challenge in/out-of-plane rotation, foreshortening, scaling, intra-category variation, aspect Ratio.

2D Body Modeling

Spring Models, Pictorial Structure Model minimize deg of mismatch $S(I,L) = \sum_{i\in V}\alpha_i \cdot \phi(I,l_i) + \sum_{ij\in E}\beta_{ij}\cdot\psi(l_i,l_j)$, ϕ unary term, likelihood this patch of image for this part i, ψ pairwise term btw part (i,j) with spring model β.

PSM with Flexible Mixtures $S(I,L,M) = \sum_{i\in V}\alpha_i^{m_i}\phi(I,l_i) + \sum_{ij\in E}\beta_{ij}^{m_i m_j}\psi(l_i,l_j) + S(M)$, m_i stands for mixture type m_i, e.g. orientation. $S(M) = \sum_{ij\in E} b_{ij}^{m_i m_j}$ co-occurrence bias, prior knowledge about mixture part co-occurrence likelihood.

Feature Representation Learning

Direct Regress ConvNet output (x,y) for certain body parts.

Heatmaps ConvNet, each keypoint with separate binary heatmap. Keypoint positive, else negative. usually Gaussian blured around keypoint.

Conv Pose Machine iteratively generate heatmaps, then with original img generate more accurate heatmap.

Think-slicing Networks combine conv feature + body struct. Energy term for a seq of slice/frame + temporal displace for each part $S_{\text{slice}} = \sum_{t=1}^T S(I^t, p^t) + \sum_{(i,i^*)\in E_f}\psi_{i,i^*}(p_i, p'_{i^*})$, $\psi_{i,j}(p_i,p_j) = w_{i,j}\cdot d(p_i - p_j)$. $d(p_i - p_j) = [\Delta x, \Delta x^2, \Delta y, \Delta y^2]^\top$.

Inference $\text{score}_i(p_i) = \phi_i(p_i|I) + \sum_{k\in\text{child}(i)} m_{ki}(p_i)$, $m_{ki}(p_i) = \max_{p_k}(\text{score}_k(p_k) + \psi_{k,i}(p_k,p_i))$ in a bottom up manner. Use subgradient for max operator when training.

3D human Pose and Shape: SMLP

Representation 3D mesh ($\tilde{7}000$ vertices and faces). Design mesh (by artist) to align unordered scans and incomplete 3D points from raw scan.

Challenge Chicken-and-egg problem, solve model and registration jointly.

Linear Blend Skinning (Naive) $\mathbf{t}'_i = \sum_k w_{ki}\mathbf{G}_k(\boldsymbol{\theta},\mathbf{J})\mathbf{t}_i$ $\mathbf{t},\mathbf{t}'$ rest/trans vertices, w_{ki} blend skinning weights, G_k rigid bone transformation, θ pose, J Joint locations.

Result creates artifact, cuz unified blending weights try to cover all kinds of trans.

SPML-LBS idea pose/shape dependent rest pose.

$T_P(\vec{\beta},\vec{\theta}) = \overline{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta})$, β body shape param, *shape blend shape* $B_S(\vec{\beta};\mathcal{S}) = \sum_{n=1}^{|\vec{\beta}|}\beta_n\mathbf{S}_n$, *pose blend shape* $B_P(\vec{\theta};\mathcal{P}) = \sum_{n=1}^{9K}(R_n(\vec{\theta}) - R_n(\vec{\theta}^*))\mathbf{P}_n$, θ^* rest pose.

Joint location dependency $J(\vec{\beta};\mathcal{J},\overline{\mathbf{T}},\mathcal{S}) = \mathcal{J}\times(\overline{\mathbf{T}} + B_S(\vec{\beta};\mathcal{S}))$, $\mathcal{J}$ regression matrix from rest verticese to rest joints,

Overall Mesh $M(\vec{\beta},\vec{\theta};\Phi)$: $\bar{\mathbf{t}}'_i = \sum_{k=1}^K w_{k,i}G'_k(\vec{\theta},J(\vec{\beta}))(\bar{\mathbf{t}}_i + \mathbf{b}_{S,i}(\vec{\beta}) + \mathbf{b}_{P,i}(\vec{\theta}))$.

Model param $\Phi = \{\overline{\mathbf{T}},\mathcal{W},\mathcal{S},\mathcal{J},\mathcal{P}\}$.

Train First $\{\mathcal{J},\mathcal{N},\mathcal{P}\}$ with multi-pose dataset, then $\{\overline{\mathbf{T}},\mathcal{S}\}$ with multi-shape dataset, separate models for men and women.

(1) Pose Parameter Training, linear comb of forllowing loss:

1. E_D: L_2 loss btw registerd and model vertices. 2. E_Y: symmetry regularization on vertice and joints. 3. E_J: $J(\vec{\beta};\mathcal{J},\overline{\mathbf{T}},\mathcal{S})$ close to default J. 4. $E_P(\mathcal{P}) = \|\mathcal{P}\|_F^2$ prevent overfitting of pose-dependent blend shape. 5. $E_W(\mathcal{W}) = \|\mathcal{W} - \mathcal{W}_I\|_F^2$ blend weights towards initial weights

Non-negative least squares with additional term that encourages the weights to add to one works best for $\mathcal{J}$.

(2) Shape Parameter Training:

1. estimate pose θ from generic shape $\arg\min_{\vec{\theta}}\sum_e\|W_e(\hat{\mathbf{T}}^P_\mu + B_P(\vec{\theta};\mathcal{P}),\hat{\mathbf{J}}^P_\mu,\vec{\theta},\mathcal{W}) - \mathbf{V}^S_{j,e}\|^2$, 2. get shape-depend vertices $\hat{\mathbf{T}}^S_j = \arg\min_{\hat{\mathbf{T}}}\|W(\hat{\mathbf{T}} + B_P(\vec{\theta}_j;\mathcal{P}),\mathcal{J}\hat{\mathbf{T}},\vec{\theta}_j,\mathcal{W}) - \mathbf{V}^S_j\|^2$. 3. PCA on $\{\hat{\mathbf{T}}^S_j\}_{j=1}^{S_{\text{subj}}}$ to obtain $\{\overline{\mathbf{T}},\mathcal{S}\}$

DMPL: Dynamic SMPL Additional term for velocity and acceleration.

SMPL Estimation Methods

Optimization Based Fitting SMPLify: (1) Given image, pass to DNN to get β,θ, (2) from SMPL get joints, (3) $\theta^* = \text{argmin}_\theta\|\text{Joint diff}\| + \text{Prior Reg}$

Cons: (1) Hand-crafted optimization routine. (2) Sensitive to initialization (3) Slow convergence.

Learned Gradient Descent (LGD) $\theta^{t+1} = \theta^t + F(\partial_\theta L_{\text{reproj}}, \theta^t, x)$, F is NN.

Train: render with different views current pose $\to$ learn how to optimize.

APP:Football match pose estimation

From key points to detailed 3D surface

Challenges: Self-occlusion, lack of depth info, articulated motion, mon-rigid deformation (clothing)

Templated-based Capture

(1) capture body with template cloth model (2) rack cloth model deformation with time. Cons: Laborious preprocessing, No public access to models → Automatic and general pipeline required.
Regression-based Reconstruction recover every pixel.
Templated-based + Regression-based

10 Implicit Surfaces and Neural Radiance Fields

3D representations: Voxels (Discretization of 3D space into grid), Points cloud, meshes (vertices and faces).

Implicit shape representation

Represent surface as the zero level-set of a continuous function $S = \{x : f(x) = 0\}$.

Neural IR (NIR)

Occupancy Networks $f_\theta : \mathbb{R}^3 \times \mathcal{X} \to [0,1]$, 3D location and condition as input, output probability
signed distance field: DeepSDF $f_\theta : \mathbb{R}^3 \times \mathcal{X} \to \mathbb{R}$, output shape represented by $f\theta(p) = \tau$.
Supervise NIR from Other representation:
(1) **Watertight meshes** Query GT occupancy or SDF from GT meshes, Train with cross-entropy loss.
(2) **Points cloud: Implicit Geometric Regularization (IGR)** unordered and hard, $\chi = \{x_i\}_{i\in I} \subset \mathbb{R}^3$), learn $f_\theta(x)$ is approximately the signed distance function to a plausible surface M defined by χ. Continuous Shortest Path (Eikonal PDE), $\|\nabla f(x)\| = v^{-1}(x)$ for $x \in \Omega_0 \subset \mathbb{R}^n$, $f(x) = q(x)$ for $x \in \Omega_T$. When $v(x) = 1$, learned function f is approximately the distance to the suface. Loss: $\mathcal{L}(\theta) = \sum_{i\in I} |f_\theta(x_i)|^2 + \lambda \mathbb{E}_x (\|\nabla_x f_\theta(x)\| - 1)^2$. Many solutions, local minima. SGD for NN is used as implicit regularization.
(3) **Images: Differentiable Volumetric Rendering (DVR)** NN output texture/-color $t_\theta(p)$ and occupancy $f_\theta(p) \in [0,1]$. **Forward** 1. given observation position $\mathbf{r}_0$ and relative pixel → direction of the ray $\mathbf{w}$, 2. find the solution d for surface, $f_\theta(\mathbf{p} = \mathbf{r}_0 + d\mathbf{w}) = \tau$, 3. use $t_\theta(\mathbf{p} = \mathbf{r}_0 + d\mathbf{w})$ for texture and color. Secant method for linesearch of zero point $x_2 = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$
Backward 1. Loss: differnce w.r.t images, $\mathcal{L}(\hat{\mathbf{I}}, \mathbf{I}) = \sum_{\mathbf{u}} \left\| \hat{\mathbf{I}}_{\mathbf{u}} - \mathbf{I}_{\mathbf{u}} \right\|$, $\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{\mathbf{u}} \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{I}}_{\mathbf{u}}} \cdot \frac{\partial \hat{\mathbf{I}}_{\mathbf{u}}}{\partial \theta}$, $\frac{\partial \hat{\mathbf{I}}_{\mathbf{u}}}{\partial \theta} = \frac{\partial \mathrm{t}_\theta(\widehat{\mathbf{p}})}{\partial \theta} + \frac{\partial \mathrm{t}_\theta(\widehat{\mathbf{p}})}{\partial \widehat{\mathbf{p}}} \cdot \frac{\partial \widehat{\mathbf{p}}}{\partial \theta}$, 2. From $f_\theta(\widehat{\mathbf{p}}) = \tau$ and $\widehat{\mathbf{p}} = \mathbf{r}_0 + \hat{d}(\theta)\mathbf{w}$ → implicit gradient $\frac{\partial \widehat{\mathbf{p}}}{\partial \theta} = -\mathbf{w}\left(\frac{\partial f_\theta(\widehat{\mathbf{p}})}{\partial \widehat{\mathbf{p}}} \cdot \mathbf{w}\right)^{-1} \frac{\partial f_\theta(\widehat{\mathbf{p}})}{\partial \theta}$.

Neural Radiance Field (NeRF)

Motivation Surfaces are good, but scenes are more complex.
Network $F_\Theta(x, y, z, \theta, \phi) = (r, g, b, \sigma)$, σ output density.
(θ, ϕ) view direction, added at late stage of network, cuz most of the texture is not view dependent, otherwise, will fall into trivial solution of sphere with complex texture. Density (geometry) is independent of viewing direction. Viewing direction only applied at a later layer, which limits the viewdependent effects and thus encourages detailed geometry.
Volume Rendering $\alpha = 1 - \mathrm{e}^{(-\sigma_i \delta_i)}, \delta_i = t_{i+1} - t_i$, Transmittance $T_i = \prod_{j=1}^{i-1}\left(1 - \alpha_j\right)$, color $c = \sum_{i=1}^{N} T_i \alpha_i c_i$.
Train $\min_\theta \sum_i \|\mathrm{render}_i(F_\theta) - I_i\|^2$, sampling efficiency is a big issue.
trick: Positional encoding pass lowdim x, y, z coordinates via fixed positional encoding controlled by L or random Fourier features of varying frequencies, instead of directly use (x, y, z). $\mathrm{PE} = \mathrm{cat}[\cos k\mathbf{v}, \sin k\mathbf{v}]_{k=1}^{2^L}$, or $\gamma(\mathbf{v}) = [\cos(\mathbf{Bv}), \sin(\mathbf{Bv})] \quad \mathbf{B} \sim \mathcal{N}\left(0, \sigma^2\right)$, too big mapping bandwith σ leads to noisy images (overfit).

11 Reinforcement Learning

MDP: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$, reward $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, transition $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, discount factor γ, initial state s_0. $G_t := \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$, value func $V_\pi(s) := \mathbb{E}_\pi[G_t | S_t = s]$, Q func $Q(s,a) := \mathbb{E}_\pi[G_t | S_t = s, a_t = a]$. Bellman eq $V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(s') | S_t = s]$, Bellman optim eq $V_*(s) = \max_a Q_*(s,a) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V_\pi(s')]$
Dynamic Programming Requires environment model (transition prob)
Monte-Carlo No need env.
Temporal-Difference DP + MC, no env.

Dynamic Programming (DP)

Value Iter (1) compute V_* (2) get π_* by V_*.
Policy Iter (1) compute V_π for π (2) update $\nu_\pi \to \pi'$
Pros (1) Exact, (2) convergence guaranteed in finite iter, (3) VI is more efficient than PI. **Cons** (1) need transition prob, (2) need to iterate over whole state space, (3) require memory $\propto$ to state space.

Monte-Carlo

$V_\pi(s) \approx \frac{1}{N} \sum_i G_i$, need experience, no transition prob(dynamics), high variance.

Temporal-Difference

$\Delta V(s) = r(s,a) + \gamma V(s') - V(s)$,
$V(s) \to V(s) + \alpha \Delta V(s)$

Exploration v.s. Exploitation

Random Visit states close to starting point more often, Far away states get neglected. **Greedy** Can find reward quickly, Getting stuck in local minimum.
Trade-off ε-greedy Policy
SARSA on policy, computes the Q-Value according to a policy and then the agent follows that policy. $\Delta Q(s,a) = r(s,a) + \gamma Q(s',a') - Q(s,a), Q(s,a) \to Q(s,a) + \alpha \Delta Q(s,a)$, can use ε-greedy Policy here.
Q-Learning Off-Policy, computes the Q-Value according to a greedy policy, but the agent follows a different exploration policy. $\Delta Q(s,a) = r(s,a) + \gamma \max_{a'}\{Q(s',a')\} - Q(s,a), Q(s,a) \to Q(s,a) + \alpha \Delta Q(s,a)$.
Comment For continuous action space the problem is intractable.
Pros 1. Less variance than Monte Carlo Sampling due to bootstrapping, 2. More sample efficient than Dynamic Programming, 3. Do not need to know the transition prob matrix.
Cons 1. Biased due to bootstrapping, 2. Exploration/Exploitation dilemma.
Deep RL $V_\pi(s) \approx V_\pi(s, \theta)$
Deep Q-learning SGD on $\mathrm{L}(\theta) = (r + \gamma \max_{a'}\{Q_\theta(s',a')\} - Q_\theta(s,a))^2$, no grad in max. **Prob** States visited in a trajectory are strongly correlated, **Sol** randomly generated sample transitions from replay buffer. OK to use old samples, since Q-learning is off-policy.

Policy Gradient

Policy: $\pi(a_t|s_t, \theta) = N(\mu_t, \sigma_t^2|s_t, \theta)$. θ params of NN.
Trajectories: $p(\tau) = p(s_1, a_1, \ldots, s_T, a_T) = p(s_1)\prod_{t=1}^{T} \pi(a_t|s_t) p(s_{t+1}|a_t, s_t)$
Idea Good trajectories made more likely. Bad trajectories made less likely.
Exploration: collecting trajectory data. The agent samples action at every timestep from the policy probability distribution (on-policy methods).
Evaluation of policy: compute the expectation of the trajectory reward. $J(\theta) = E_{\tau\sim p_\theta(\tau)}[\sum_t \gamma^t r(s_t, a_t)]$, $\theta = \theta + \nabla_\theta J(\theta)$, $\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p(\tau)}[\nabla_\theta \log p(\tau) r(\tau)] = \mathbb{E}_{\tau\sim p(\tau)}[(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i)) \times (\sum_{t=0}^{T} \gamma^t r(s_t^i, a_t^i))]$.
Hint: $\nabla_\theta p(s_{t+1}|a_t, s_t) = 0$.
Problem MC on gradient high variance. **Solution** add a baseline $\nabla J(\theta) = \frac{1}{N}\sum_i (\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i)) \times (\sum_{t=0}^{T} \gamma^t r(s_t^i, a_t^i) - b(s_t^i))$ Baseline: average reward, estimate of the $V(s)$.

Actor-critic

Bootstrapping in G_τ: $r(s_t^i, a_t^i) + \gamma V(s_{t+1}^i) - V(s_t^i)$, introduce bias and reduce the variance.
$\nabla_\theta J(\theta) = \frac{1}{N}\sum_i \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) \times (r(s_t^i, a_t^i) + \gamma V(s_{t+1}^i) - V(s_t^i))$.
The policy (actor) and the value function (critic) are both represented by neural networks (SOTA).