1 Intro

Perceptron: while $\exists x. w^T x > 0 \neq y$ do $w = w + \eta(y - \widehat{y})x$

MLP: $\forall l \quad x^{(l)} = \sigma((w^{(l)})^T x^{(l-1)} + b^{(l)}), f(x; w, b) = x^{(L)}$

Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$, $\nabla : \sigma(x) \cdot (1 - \sigma(x))$

Softmax: $softmax(x_i) = \exp(x_i)/\sum_j \exp(x_j)$

tanh: $\nabla : 1 - tanh(x)^2$

MLE: maximize $\log L(\theta) = \log \prod p(x_i|\theta) = \sum \log p(x_i|\theta)$

L_{CE}: $-\frac{1}{N}\sum y_i \log(\sigma(w^T x_i)) + (1-y_i)\log(1-\sigma(w^T x_i))$ (MLE)

Universal Approximation: $\exists g(x) = \sum v_i \sigma(w_i^T x + b_i) \approx f(x)$ and $|g(x) - f(x)| < \epsilon$, σ non-const, bounded, continus

SGD: $\theta = \theta - \eta \nabla C(\theta)$, **Batch:** gradient avg. of whole dataset

2 Convolutional Neural Network

Generalization: trade-off specifity vs. invariance

Receptive Field: area triggering neuron, inhibit or excit

Hierarchy: **S**imple cells, specific stimuly, **C**omplex cells, complex stimuly

HMAX: S: $y = \exp(-\frac{1}{2\sigma^2}\sum^{n_{S_k}}(w_j - x_j)^2)$, C: $y = \max_{n_{C_k}} y_j$

Linear: $T(\alpha u + \beta v) = \alpha T(u) + \beta T(v)$
Invariant: $T(f(u)) = T(u)$
Equivariant: $T(f(u)) = f(T(u))$

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

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

Matrix $I * K$: band matrix $K \in \mathbb{R}^{n+m-1 \times n}$, k_i on diagonal

Diff.: Conv. with kernel [-1, 1], $\frac{\delta f}{\delta x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$

Dimension: $\frac{I_{height} + 2 \cdot \text{padding} - \text{dilation} \cdot (K_{height} - 1) - 1}{\text{stride}} + 1$

Weight sharing: Same feature detector K for whole I

Stride: used to reduce size, replaces pooling layers
Dillation: fast increase of rec. field, get global context

CNN-fwd: $z_{i,j}^{(l)} = w^{(l)} * z^{(l-1)} + b = (\sum_{m,n} w_{m,n}^{(l)} z_{i-m,j-n}^{(l-1)}) + b$

CNN-bwd (z): $\delta_{i,j}^{(l-1)} = \frac{\delta C}{\delta z_{i,j}^{(l-1)}} = \sum_{i',j'} \frac{\delta C}{\delta z_{i',j'}^{(l)}} \frac{\delta z_{i',j'}^{(l)}}{\delta z_{i,j}^{(l-1)}}$

$= \sum_{i',j'} \delta_{i',j'}^{(l)} w_{i'-i,j'-j}^{(l)} = \delta_z^l * ROT_{180}(w^{(l)})$

TODO: Do calculation yourself to see why it is flipped, or better, understand the relation to the "standart" convolution

Pooling: $z^l = \max\{z_i^{l-1}\}$, $\frac{\partial z^l}{\partial z_i^{l-1}} = 1$ if was max else 0

2.1 Evolution of architectures

2.2 VGG vs. AlexNet

less kernels/filters ($\downarrow$ params), more layers ($\uparrow$ perceptive field)

2.3 GoogleNet

More layers, removed fully connected layer on the top.
Inception: use 1x1 conv. layers to reduce layer depth

2.4 ResNet

Residual-Con: Skips weight layers with residual connections.

3 Fully Convolutional Neural Network

Semantic segment.: extract patch, run through cnn, classify + size independen, - only local context

Downsample: pooling or strides, otherwise very expensive

Nearest Neighbor: copy value to the whole output

Bed-of-nails: zero all outputs but one copy of input

Max-unpooling: remember max element from downsampling, use that in upsampling

Learnable Upsampling: input as weight, add up filters in output

UNet: copy early stage tensors to upsampling, combines local and global feature maps

4 Recurrent Neural Network

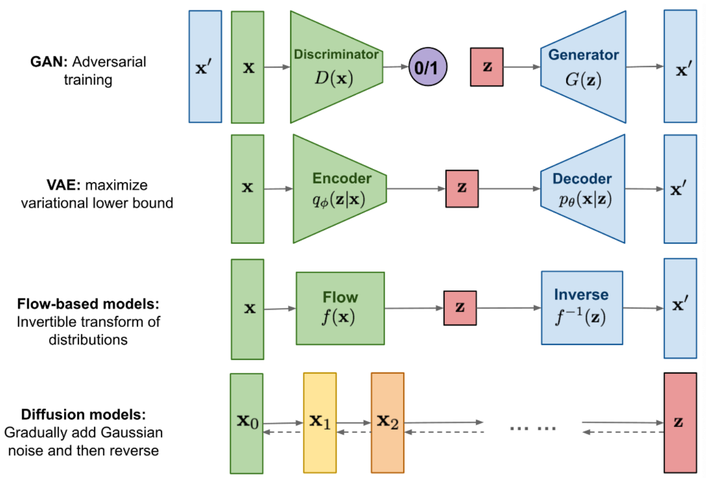
RNN: $\widehat{y} = W_h h^t$, $h^t = tanh(W[h^{t-1}, x^t])$,
$h \in \mathbb{R}^n, W \in \mathbb{R}^{[n \times 2n]}$ + variable sequence length

Backprop: $\frac{\delta L}{\delta W} = \sum_{t=1}^{S}\sum_{k=1}^{t} \frac{\delta L^t}{\delta y^t}\frac{\delta y^t}{\delta h^t}(\prod_{i=k+1}^{t}\frac{\delta h^i}{\delta h^{i-1}})\frac{\delta^+ h^k}{\delta W}$

Gradient Problem: $h^t = W^T h^{t-1} = (W^T)^t h^1 = (Q^T \Lambda^t Q) h^1$
- explode: clip (stability), - vanish: memory cell

Naive Memory: $c^{(t)} = W_c c^{(t-1)} + W_g g^{(t)}$, $h^{(t)} = tanh(c^t)$

LSTM: $c_t^l = f \odot c_{t-1}^l + i \odot g$, $h_t = o \odot tanh(c_t^l)$,
$f, i, o, g = [\sigma, \sigma, \sigma, tanh] \odot W^l [h_{t-1}^l h_t^{l-1}]^T$, $W^l \in \mathbb{R}^{4n \times 2n}$
1) forget, 2) new info, 3) output gen



5 Variational Auto Encoders

AE: optimize $\theta_f, \theta_g = \arg\min \sum_n^N \|x_n - g_\theta(f_\theta(x_n))\|^2$

PCA: $z = f(x) = Wx + b$, $\widehat{x} = g(z) = W^* z + c$

NN: $z = f(x) = \sigma(Wx + b)$, $\widehat{x} = g(\widehat{a}(x)) = \sigma(W^* z + c)$

Latent Space: meaningful DOF, continous and interpolatable undercomp: compress, features; overcomp: copy components

Denoise: input+gaussian noise, reconstruct w. overcomp. z

Limit: z not interpolatable; +reconstruction, -new samples

VAE: The latent space Z is approximated by $f(x) \sim \mathcal{N}(\hat{\mu}, \hat{\sigma} I)$

Encode: $q_\phi(z|x)$ for $\mu_{(z|x)}, \Sigma_{(z|x)}$, $z|x \sim \mathcal{N}(\mu_{(z|x)}, \Sigma_{(z|x)})$

Decode: $p_\theta(x|z)$ for $\mu_{(\widehat{x}|z)}, \Sigma_{(\widehat{x}|z)}$, $\widehat{x}|z \sim \mathcal{N}(\mu_{(\widehat{x}|z)}, \Sigma_{(\widehat{x}|z)})$

LH: $p_\theta(x) = \int_z p_\theta(x|z) p_\theta(z) dz$, Post: $p_\theta(z|x) = \frac{p_\theta(X|Z) p_\theta(z)}{p_\theta(x)}$

KL: $-D_{KL}(q_\phi(z|x)||p_\theta(z)) = \int_x q_\phi(z|x) \log(\frac{p_\theta(z)}{q_\phi(z|x)})$

$= \frac{1}{2}\sum_i^J (1 + \log(\sigma_j^2) - \mu_j - \sigma_j^2)$, if $q \sim \mathcal{N}(\mu, \sigma I), p \sim \mathcal{N}(0, I)$

Properties: asymmetric and non-negative

from exercise: $\int p(z) \log q(z) dz =$

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

ELBO: $p_\theta(x) \geq E_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p_\theta(z))$

Re-parameterization: $z = \mu + \sigma\epsilon$, $\epsilon \sim \mathcal{N}(0, 1)$

Goal: Learn features that correspond to distinct factors of variaton e.g., digits and style (or thickness, orientation)

β-VAE: $\mathcal{L} = -E_{q_\phi(z|x)}[\log p_\theta(x|z)] + \beta D_{KL}(q_\phi(z|x)||p_\theta(z))$

Faces: generated faces are typically blurry

6 Auto-regressive Models

Generativ vs. Discrim.: $P(X|Y) = P(X, Y)$(or $P(X)$)

Regressive Property: $x_t = b_0 + b_1 x_{t-1} + b_2 x_{t-2}$

Sequence Model: $p(x) = \prod^N p(x_i|x_1, .., x_{i-1}) = \prod p(x_i|x_{<i})$
+ NLL gives good comparison metric

Believe Net: $\widehat{x}_i = p(x_i = 1|x_{<i}) = Ber(\sigma(\sum_1^{i-1} w_j^i x_j + w_0^i))$

NADE: $h_i = \sigma(b + \sum^{i-1} w_j x_j)$ (in $O(H)$, update in $O(1)$)
$\widehat{x}_i = p(x_i = 1|x_{<i}) = \sigma(c_i + V_i h_i)$ (N times $\Rightarrow$ total $O(NH)$)
Train: $\frac{1}{T}\Sigma \log(p(x^t)) = \frac{1}{T}\sum^T \sum^D \log p(x_i^{(i)}|x_{<i}^{(t)})$

PixelRNN: $p(x) = \prod^{N^2} p(x_i|x_{<i}) = p(x_{i,R}|x_{<i}) p(x_{i,G}|x_{<i}, x_{i,R}$
autoreg. from nature: h_t summarises $x_{<t}$; - train/gen is slow

PixelCNN: Use conv + mask, parallelize training, blind spot

WaveNet: dilated convolution for large scale temp. dependencies

Dilated Conv: exponential receptive field size increase

Self-Attention: $K = XW_K, V = XW_V, Q = x_t W_Q (\in \mathbb{R}^D)$,
$X \in \mathbb{R}^{T \times D}, W \in \mathbb{R}^{D \times D}, \alpha = softmx(QK^T/\sqrt{D}), x_{t+1} = \alpha V$
$X = softmax(\frac{(W_Q X)(W_K X)^T}{\sqrt{D}} + M)(W_V X)$
Complexity/path: $O(T^2 D)/O(1)$; RNN: $O(nD^2)/O(n)$, Conv: $O(knD^2)/O(\log_k(n))$

7 Normalizing Flows

+ invertible, +exact LL, +latent space

Variable change 1D: $p_x(x) = p_z(h(x))\,|h(x)|$
2D: $\int\int f(x, y) dx dy = \int\int f(g(u, v), h(u, v)) J(u, v) du dv$

Matrix det. lemma: $det(A+uv^T) = (1+v^T A^{-1} u) det(A)$

Normalizing Flow: $f: \mathbb{R} \to \mathbb{R}$, cont. and invertible

$p_X(x;\theta) = p_Z(f_\theta^{-1}(x)) \left| det(\frac{\partial f_\theta^{-1}(x)}{\partial x}) \right| = p_z(z) \left| det(\frac{\partial f(z)}{\partial z}) \right|^{-1}$

Triangular Jacobian det in $O(d)$, else $O(d^3)$

Coupling: $(y_A; y_B)^T = (h(x^A, \beta(x^B)); x^B)^T$ | h:elemnt, β:NN $(x^A; x^B)^T = (h^{-1}(y^A; \beta(y^B)); y^B)^T$, $J = ((h'; h' f'), (0; 1))$

Composition: $p_X(x;\theta) = p_Z(f_\theta^{-1}(x)) \prod_k \left| det(\frac{\partial f_\theta^{-1}(x)}{\partial x}) \right|$

Train:$logp_x(D) = \sum_x^D (logp_z(f^{-1}(x)) + \sum_k \log \left| det(\frac{\partial f^{-1}(x)}{\partial x}) \right|)$

Inference: $z \sim p_z(.)$, $\hat{x} = f(z)$

Model: $x_i \to squeeze \to n.flow \to split \to z_i$(repeat $L-1$) $\to squeeze \to n.flow \to z_L$, z_i are outputs

Squeeze/Split: reduce spatial dim, pass on half

Flow: actnorm, 1x1 conv, coupling

StyleGAN vs. StyleFlow: Replace mapping Net with normal flow

8 Generative Adversarial Networks

Imperfect Model: High LL can result in bad samples and vv memorise training data (-LL), high noise samples (+LL)

Generator: map $z \in \mathbb{R}^Q$ to observ. $x \in \mathbb{R}^D$, $G: \mathbb{R}^Q \to \mathbb{R}^D$

Discriminator: trained on $\hat{x}$ and x, $D: \mathbb{R}^D \to [0,1]$

No markov chain necessary

Loss:$-\frac{1}{2N}(\sum_i^N (y^{(i)}) \log(D(x^{(i)})) + \sum_n^{2N} (1 - y^{(i)}) \log(1 - D(x^{(i)})))$

Train: $G^*, D^* = \arg\min_G \arg\max_D \log(D(x)) + \log(1 - D(\hat{x}))$

Opt: $V(G, D^*) = \mathbb{E}_{x \sim p_d}(\log(D^*(x))) + \mathbb{E}_{x \sim p_m}(\log(1 - D^*(x)))$ $= -\log(4) + 2D_{JS}(p_d(x) || p_m(x)) = -\log(4)$ if $p_d(x) = p_m(x)$

$f(x) = a \log x + b \log(1-x) \in [0,1]$ has max at $\frac{a}{a+b}$

Update D: for k : $\nabla_{\Theta_D} \frac{1}{N} \sum \log(D(x^{(i)})) + \log(1 - D(G(z^{(i)})))$

Optimum $D^* = \frac{p_{data}(x)}{p_{data}(x) + p_{model}(x)}$

Update G: $\nabla_{\Theta_G} \frac{1}{N} \sum \log(D(G(z^{(i)})))$ (ascent)

Assumption: capacity, $D \to D^*$, opti. p_{model}

Mode Collapse: G finds one mode, D fails to reject

Oscillation: limited capacity and not D^*

Wasserstein: work to similarity; +no collapse, +stable

Cons: no explicit $p(x)$, nor sample LL, carful balancing, less theory

9 Parametric Body Models

LBS: $t_i' = \sum_k w_{ki} G_k(\theta, J) t_i$

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

Pipeline: Template mesh, joint locations, shape to body shape, add pose correction, LBS

Learned GD: $\Theta^{t+1} = \Theta^t + F(\frac{\partial L}{\partial \Theta}, \Theta^t, x)$

Points to Surfaces

10 Neural Implicit Representations

Traditional: cam, pointcloud, mesh, tracked mesh

Voxel: 3D grid, limited resolution in $O(n^3)$

Points: Sensor meassurements, no connectivity/ topology

Mesh: Vertices and surfaces, self-intersections, discont.

Implicit Repr.: set-level of cont. function, no approx error store SDF values on a regular grid

Neural Impl. Repr: function as NN represents shape; +low memory

from Mesh: $\mathcal{L}(\theta, \phi) = \sum BCE(f_\theta(p_{ij}, z_i), o_{ij})$, rand. query

from Pointcloud: $\mathcal{L}(\theta) = \sum |f_\theta(x_i)|^2 + \lambda \mathbb{E}_X(\|\nabla_X f_\theta(x)\| - 1)^2$

Eikonal PDE: $\|\nabla f(x)\| = 1, f(x) = 0, x \in \Omega$, gives SDF Ω $\mathcal{L}(\theta) = \Sigma_i |f_\theta(x_i)|^2 + \lambda \mathbb{E}_X(\|\nabla_x f_\theta(x)\| - 1)^2$; converges!

Derivating Volume rendering point location + encoded picture, 5 ResNet, Occupancy and Texture head

Forward: $\forall u$, ray from r_0 through u to root $\hat{p}$ ($\downarrow$), $u = t_\theta(\hat{p})$ $y_2 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_1) + f(x_1), x_2 = x_1 - f(x_1) \frac{x_1 - x_0}{f(x_1) - f(x_0)}$

Render: shoot rays, rough occupancy est., secant, texture query

Backprob: $\mathcal{L}(\hat{I}, I) = \sum \left\| \hat{I}_u - I_u \right\|$, $\frac{\partial \mathcal{L}}{\partial \theta} = \sum \frac{\partial \mathcal{L}}{\partial \hat{I}_u} (\frac{\partial t_\theta(\hat{p})}{\partial \theta} + \frac{\partial t_\theta(\hat{p})}{\partial \hat{p}} \frac{\partial \hat{p}}{\partial \theta})$, $\frac{\partial \hat{p}}{\partial \theta} = -w(\frac{\partial f_\theta(\hat{p})}{\partial \hat{p}} w)^{-1} \frac{\partial f_\theta(\hat{p})}{\partial \theta}$

NeRF: $F(x, y, z, \theta, \phi) \to (r, g, b, \sigma)$, whereas F is FCNN w. ReLU

-static only, -slow render, -need many views

Render: Shoot ray, evaluate all, alpha compose for color

α-composition: $\alpha_i = 1 - e^{-\sigma_i (t_{i+1} - t_i)}$

Transmittance: $T_i = \prod_1^{i-1} (1 - \alpha_j)$, **Color:** $c = \sum T_i \alpha_i c_i$

Positional encoding: location in fourier space, easier to approximate high frequencies

11 Reinforcement Learning

map states to actions, maximize reward, in uncertain and unknown environment

Return: $G_t = \sum^\infty y^k R_{t+k+1}$

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

Q-Func: $q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$, +no trans function

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

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

DP: compute optimal policy in perfect model, limited utility

Greedy V policy: $\pi'(s) = \arg\max_a (r(s, a) + \gamma V_\pi(p(s, a)))$ fix point $V^*(s) = \max_a r(s, a) + \gamma V^*(p(s, a))$, π^* of V^* is π^* calculate new value function forall state, more efficient

Policy iteration: Evaluate V with current policy, improve policy

X: +exact, +converges, -trans. prob, -iterate states, -memory

TD learning: $\Delta V(s) = r(s, a) + \gamma V(s') - V(s)$, +less variance $V(s) \leftarrow V(s) + \alpha \Delta V(s)$, +use only visited states, +effiecient, +no trans. prob, -local min, -biased

eps-greedy:. take best action, but random with low prob

SARSA: $\Delta Q(S, A) = R + \gamma Q(S', A') - Q(S, A)$, on-policy $Q(S, A) \leftarrow Q(S, A) + \alpha \Delta Q(S, A)$

Q-Learning: $\Delta Q(S, A) = R_{t+1} + \gamma \max_a [Q(S', a)] - Q(S, A)$ $Q(S, A) \leftarrow Q(S, A) + \alpha \Delta Q(S, A)$, off-policy

Deep Q: $\mathcal{L}(\theta) = (R + \gamma \max_{a'}[Q_\theta(S', a')] - Q_\theta(S, A))^2$ i.i.d assumption, use replay buffer to replay

learn $\pi: S_t \to A_t$ and $v_\pi: S_t \to V(S_t)$ as NN

Policy Gradient: $\pi(a_t | s_t) = \mathcal{N}(\mu_t, \sigma_t^2 | s_t)$. $p(\tau) = p(s_1) \prod \pi(a_t | s_t) p(s_{t+1} | a_t, s_t)$

Update: $\theta^* = \arg\max J(\theta)$, $\theta = \theta + \nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau)}[(\sum^T \nabla_\theta \log \pi_\theta(a_t^i | s_t^i))(\sum^T y^t r(s_t^i, a_t^i))]$

Reinforce: $\nabla_\theta J(\theta) = \frac{1}{N} \sum_i [(\sum^T \nabla_\theta \log \pi_\theta(a_t^i | s_t^i))(\sum^T y^t r(s_t^i, a$ $b(s_t^i))]$

Actor-Critic: $\nabla_\theta J(\theta) = \frac{1}{N} \sum_i \sum^T \nabla_\theta \log \pi_\theta(a_t^i | s_t^i)(r(s_t^i, a_t^i) + \gamma V(s_{t+1}^i) - V(s_t^i))$