2 Introduction

MLE: $\hat{\theta} = \arg\max_\theta \sum_i \log p(x_i|\theta)$.
MLPs are universal function approximators / Boolean machines / classification functions: σ non-constant bounded continuous, $\epsilon > 0$, $\forall f \in C(I_m)$ (cube of dim m), $f(x) \approx g(x) = \sum_{i=1}^{n} v_i \sigma(w_i^T x + b_i)$ (single hidden layer).
Double Descent both epoch-wise and model complexity-wise: bias-variance tradeoff until interpolation threshold, then gets better again. $\text{EMC}_{D,\epsilon}(\mathcal{T}) = \max\{n | \mathbb{E}_{S\sim D}[\text{Error}_S(\mathcal{T}(S))] \leq \epsilon\}$. If $\ll n$ (underparam) or $\gg n$ (overparam) can decrease test error by making n bigger, if $\approx n$ error can increase or decrease.
Supervised learning f maps data $x \to$ label y. **Unsupervised learning** data underlying structure: clustering, feature learning, dim reduction, density estimation.
Generative modelling learn p_M sim to $p_{\mathcal{D}}$, then sample fromp_M:

- Explicit density:
  - Approximate:
    - Variational: VAE, Diffusion
    - Markov Chain: Boltzmann mach.
  - Tractable:
    - Autoregressive: FVSBN/NADE/-MADE, Pixel(c/r)nn, WaveNet/tcn, Autoregressive Transformer, LLM
    - Normalizing Flows
- Implicit density:
  - Direct: Generative Adversarial Nets
  - Markov Chain: Gen Stoch Nets

3 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 challenge.
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) **Equivariance** to f, $T(f(u)) = f(T(u))$.

Convolution

Correlation $A \otimes B$ **Convolution** $A * B$
(Cross-)Correlat. slides matrices in original orientation, and $A*B = A \otimes \text{ROT}_{180°}(B)$

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

Pooling

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

Application

Alexnet Smaller filter, more layers.
VGG +layers, -params, +receptive field.
GoogLeNet Deeper, Inception module, 1x1 conv for dim/channel reduction64 $\to$ 32, Auxiliary cls head $\Rightarrow$ More efficient.
ResNet +layers, faster, uses residual connections: (1) better convergence. (2) propagate high freq info (U-net).
U-Net fully CNN, combine global+local features using tensors from prev layer.

Fully CNN

CNN + final MLP only fixed size, for segmentat. can't see whole context; fully CNN convolutions at original image size expensive so uses down+up-sampling.
Unpooling Methods: Nearest Neighbor, Bed of Nail, Max Unpooling (need maxpool from prev net).
ConvTranspose2D / strided upconvolution shape: $L_{\text{out}} = (L_{\text{in}} - 1)\text{Stride}_{=1} - 2\text{Pad}_{=0} + \text{Dilat}_{=1}(\text{Ker} - 1) + \text{OutPad}_{=0} + 1$
Motion infilling uses conv. autoenc.

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 $\frac{\partial L^t}{\partial W} = \sum_{k=1}^{t} \frac{\partial L^t}{\partial \hat{y}^t} \frac{\partial \hat{y}^t}{\partial h^t} \frac{\partial h^t}{\partial h^k} \frac{\partial^+ h^k}{\partial W}$,
$\frac{\partial h^t}{\partial h^k} = \prod_{i=k+1}^{t} W_h^T \text{diag}[f'(h^{i-1})]$.

Gradient vanishing/exploding

$\lambda_1 := \max \lambda(W_{hh})$, $\gamma > \|\text{diag}(f'(h^{i-1}))\|$, 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$\Leftrightarrow$jumper over local minima or oscillate.
Gradient clipping$\to$prevent ∇ explode.
LSTM Inpt, forgt, outpt $\in [0,1]$, gate $\in \mathbb{R}$. (predecessor to LSTM is GRU)
Vanila: $h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$. **LSTM**:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}, \; W : 4n \times 2n,$$

$c_t = f \odot c_{t-1} + i \odot g \qquad h_t = o \odot \tanh(c_t)$

5 Variational Auto-Encoder

Auto-Encoder (AE)

Encoder proj to latent space z, **Decoder** proj back. **Loss** $\sum \|x_i - g(f(x_i))\|_2^2$. Optimal linear autoencoder is PCA. Undercomplete: $|Z| < |X|$, else overcomplete. Overcomp. is for denoising, inpainting.
Dis/Adv Reconstruction ✓, Generation ✗. Latent space not well-structured: no continuity, no interpolation.
Applications (1) Denoising 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.

Variational Auto-Encoder (VAE)

$\ln p_\theta(x) = \text{ELBO}_{\theta,\phi}(x) + \text{KL}(q_\phi(z|x) \| p_\theta(z|x))$.
Objective: $\max_{\theta,\phi} \sum_i \text{ELBO}_{\theta,\phi}(x_i)$.
$\text{ELBO} = \mathbb{E}_{q_\phi} \ln p_\theta(x|z) - \text{KL}(q_\phi(z|x) \| p_\theta(z))$
$\mathbb{E}_{q_\phi}[\log p_\theta(x|z)]$ reconstruction likelihood, encourage clustering for similar samples in latent space. $-\text{KL}(q_\phi(z|x) \| p_\theta(z))$ makes posterior close to prior, encourages latent representations evenly around center, compactness, smooth interp.
(1) If only recon: VAE$\to$AE, sharp recon but sparce latent space. (2) No recon loss: compact embedding like Gaussian, but bad for reconstruction.
Encoder: $q_\phi(z|x) = \mathcal{N}(\mu_{\text{nn}}(x), \Sigma_{\text{nn}}(x))$,
Decoder: $p_\theta(x|z) = \mathcal{N}(\mu_{\text{nn}}(z), \Sigma_{\text{nn}}(z))$.
Approximate $\mathbb{E}$ in loss with sampling.

Derivation $\log p(x) = \log \int p(z) p(x|z)\, dz =$
$\log \int p(z) p(x|z) \frac{q(z|x)}{q(z|x)}\, dz =$
$\log \mathbb{E}_{z\sim q(z|x)} \left[p(x|z) \frac{p(z)}{q(z|x)} \right] \geq$
$\mathbb{E}_{z\sim q(z|x)} \left[\log \left(p(x|z) \frac{p(z)}{q(z|x)} \right) \right] =$
$\mathbb{E}_{z\sim q(z|x)} \left[\log p(x|z) - \log \frac{q(z|x)}{p(z)} \right] =$
$\mathbb{E}_{z\sim q(z|x)} [\log p(x|z)] - D_{\text{KL}}(q(z|x) \| p(z))$

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 conditionally indep factors, w cond dep facts (entangled), $\log p(v|x) = \sum_k \log p(v_k|x)$. Train: $\max_{\phi,\theta} \mathbb{E}_{x,q_\phi} \ln p_\theta(x|z)$, s.t. $\text{KL}(q_\phi \| p_\theta(z)) < \delta \Leftrightarrow L_{\beta,\phi,\theta} = \text{NLL} + \beta \text{KL}$, $\beta > 1$ stronger constraint on latent space than VAE.

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 $p(x) = \prod_i p(x_i|x_{<i})$.

Fully Visible Sigmoid Belief Networks

Tabular approach needs 2^L states not acceptable, replace with regression $f_i(x_{1:i-1}) = \sigma(\alpha_0^i + \sum_{j=1}^{i-1} \alpha_j^i x_j)$, $p_{\theta_i}(x_i|x_{<i}) = \text{Ber}(f(x_{1:i-1}))$.

Neural Autoreg Density Estim (NADE)

Extend FVSBN with NN, $h_i = \sigma(b + W_{<i} x_{<i})$, output $\hat{x}_i = \sigma(c_i + V_i \mathbf{h}^{(i)})$. Note: $(b + W_{\leq i} x_{\leq i}) - (b + W_{<i} x_{<i}) = W_i x_i$.
Train w/ NLL $\sum_{t=1,i=1}^{T,D} \ln p(x_i^{(t)} | x_{<i}^{(t)})$. 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 fulfills 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.

PixelRNN / 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, only see context region, so faster training.
Pred $p(x_i|x_{<i}) = p(x_{i,R}|x_{<i})p(x_{i,G}|x_{i,R}, x_{<i}) \times p(x_{i,B}|x_{i,R}, x_{i,G}, x_{<i})$ for RGB handling
Finite kernel mask Conv →blind spot: actual prediction of pixel does not depend on some region of unmasked part.

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 $p_\theta(z) = \prod_{t=1}^T p_\theta(z_t|z_{<t}, x_{<t}) = \prod_{t=1}^T p_\theta(z_t|h_{t-1}) \times p_{f_\theta}(h_{t-1}|h_{t-2}, z_{t-1}, x_{t-1})$. Each z_t depends on previous $x_{<t}$ and $z_{<t}$.
Adv for z in f_θ: (1) robust by randomness of z (2) latent z is high level, more informative on future.
Dec: $p(x_t|z_t) = g^{\text{out}}(h_{t-1}, z_t)$,
Enc: $q_\phi(z|x) = \prod_{t=1}^T q_\phi(z_t|x_t, x_{<t}, z_{<t})$,
ELBO $\sum_t \log p_\theta(x_t|x_{<t}) \geq \mathcal{L}_{\theta,\phi} = \sum_{t=1}^T \mathbb{E}_{q_\phi(z_t|x_t,x_{<t},z_{<t})}[\log p_\theta(x_t|z_t, z_{<t}, x_{<t})] - \mathsf{KL}(q_\phi(z_t|x_t, x_{<t}, z_{<t}) \| p_\theta(z_t|z_{<t}, x_{<t}))$.
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: $\sum_{t=1}^T \mathbb{E}_{q(z_t,\pi_t x_t)} \log p(x_t z_t, \pi_t) - \mathsf{KL}(q(z_t x_t)\|p(z_t)) - \mathsf{KL}(q(\pi_t x_t)\|p(\pi_t))$
APP Digital Handwriting Modeling.

Summary

Pros $p(x)$ tractable, explicit NLL metric to compare performance, easy to train, sample. **Cons** Slow, 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. $\mathsf{PE}_{i,t} = \sin/\cos(\omega_k t)$ for $i = 2k/2k+1$, $\omega_k = 10000^{-2k/d}$, t position in seq, i component 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.

LLM / GPT

Needs tokenization, can be finetuned by combining unsupervised pretraining $L_1(X) = \sum \log P(x_i|x_{i-1:i-k})$ plus supervised finetuning $L_2(X,Y) = \sum \log P(y|x\dots)$, then $L = L_2(X,Y) + \lambda L_1(X)$
For **autoreg. image/video generation**: problems with image quality, computational requirements. Solution: quantize images into tokens (e.g. Vector-Quantized **VQ-VAE** uses CNN enc/dec): lower dimensionality, more semantically meaningful. Also, can use LLMs.

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

Intuition Likelihood not good indicator for generation, can be independent: $\ln[0.01 p_{\text{true}} + 0.99 q_{\text{noise}}] \geq \ln[0.01 p_{\text{true}}] = \ln p(x) - \ln 100$.
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 $D: \mathbb{R}^D \to [0,1]$ (0=fake).
Objective $\text{argmin}_G \max_D V_{G,D} = \mathbb{E}_{x\sim p_d}[\ln D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[\ln(1 - D(\hat{x}))] = \mathbb{E}_{x\sim p_d}[\ln D(x)] + \mathbb{E}_{\hat{z}\sim p_z}[\ln(1 - D(G(z)))]$
Optimal $D^* = \text{argmax}_D V_{G,D} = \frac{p_{\text{D}}(x)}{p_{\text{D}}(x) + p_{\text{M}}(x)}$, proof by turning $\mathbb{E}$ into integrals and solving for $0 = \nabla_D$integral body (and ensure concave, $\nabla\nabla_D > 0$). Then $V_{G,D^*} = -\ln 4 + 2\,\mathsf{JS}(p_d(x)\|p_m(x))$, global optim if $p_d(x) \equiv p_m(x)$.
If G, D enough capacity (strong assumpt), 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 $\propto \sup_D \mathbb{E}_{p_m(x)} \ln(1 - D(x))dx$.
(1) In practice finding optimal 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 → smaller ∇ ⇒ 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 (simulate k' more steps of D to calculate grads for G, then revert to prev D).
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. (4) lack of theory and learning algo wrt explicit models.

Applications

Progressive Growing of GANs: Grow the generator and discriminator resolution by adding layers during training.
StyleGAN uses layer-wise style conditioning with AdaIN (see tutorial), allows style mixing at different layers, has separate mapping network to provide conditioning.
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. Translates samples from two sides e.g. facade rectangles to facade img.
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.
BicycleGAN both image and style cycle consistency.
Vid2vid uses $p(\tilde{y}_{1:T} \mid x_{1:T}) = \prod_{t=1}^T p(\tilde{y}_t \mid \tilde{y}_{t-L:t-1}, x_{t-L:t})$ to also use past frame.
GAN can be used anywhere: **RLHF** used for training LLMs.
3D Applications: 3D-GAN outputs objects, PlatonicGAN uses 3D-GAN but converts back to 2D for training to make use of images, HoloGAN like PlatonicGAN but outputs 3D features that can be rendered with neural rendereing, EG3D uses tri-plane projections 3D representation and neural rendering

8 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:$x = f_k \circ f_{k-1} \circ \cdots f_2 \circ f_1(z)$, $p_x(x) = p_z(f^{-1}(x)) \prod_k |\det \partial_x f_k^{-1}(x)|$

Coupling Layer

f: $\begin{pmatrix} y^A \\ y^B \end{pmatrix} = \begin{pmatrix} h(x^A, \beta(x^B)) \\ x^B \end{pmatrix}$, f^{-1}: $\begin{pmatrix} x^A \\ x^B \end{pmatrix} = \begin{pmatrix} h^{-1}(y^A, \beta(y^B))) \\ y^B \end{pmatrix}$, Jacob: $\begin{pmatrix} h' & h'\beta' \\ 0 & 1 \end{pmatrix}$.
$h(x) = (h(x_1), \ldots, h(x_n))^\top$ element-wise. $\partial_{x^\top} h = \mathrm{diag}(h'(x_1), \ldots, h'(x_n))$.
Continuous NF: model infinite number of trans with proper temporal discretization. I.e. model the evolution of a neural ODE.

Multiscale architecture, *FB = step of flow*

x -> [Squeeze, [FB1, FB2, FB3]×K, Split -> z_i]×($L-1$), Squeeze, [FB1, FB2, FB3]×K -> z_L
Squeeze reduces spatial dimensions (and increases num of channels). Split outputs half of the features z (so it iteratively gets smaller).

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 $\mathrm{H} \cdot W \cdot \mathrm{sum}(\log|s|)$.

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 $W = PL(U + \mathrm{diag}(s))$, $\log|\det W| = \mathrm{sum}(\log|s|)$.

Fb3 - (conditional) affine coupling layer

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 with two flows
Conditioning $Flow_A$: Standard affine coupling layer, to images.
Conditioned $Flow_B$: trans param conditioned on, to point cloud, sketch, seg, etc.
Applications (1) (multimodal) image-to-image mapping (2) style transfer (3) image manipulation (4) 3D point cloud reconstruction from images. (5) Probabilistic Modeling for Human Mesh Recovery (6) PointFlow: 3D Point Cloud Generation (7) but long training and low res

Difficulty for generative model

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

9 Diffusion & Foundation models

Denoising Diffusion Probability Models

Adv VAE/AR low quality, GANs unstable to train, NF must be invertible. Diffusion model none of these.
NN for denoising: usually UNet with conditioning on t.
Conditioning (1) train on just images of cats (2) classifier guidance: nudge noise in direction of cats with $\omega \cdot \nabla$classif, but problems if limited or errorful classif, and must train classif together with NN (3) classifier-free guidance: condition NN but still include 20% of unconditional for diversity and quality tradeoff.
Latent diffusion models: denoise in smaller latent space with more semantics, so first need to train an autoencoder, then use frozen decoder to reconstruct image. App: Stable Diff, DALL-E 3, SORA.
Text-to-image limited, so do feature modulation on top of text embeddings. Can use also for photo relighting (IC-Light).
Zero-Conv in **ControlNet** is alternative feat mod $y_c = NN_{\text{frozen}}(x) + Z_{2,\text{init}=0}(NN2_{\text{trainable}}(x + Z_{2,\text{init}=0}(c)))$, Z_1 and Z_2 are 1x1 conv.

DDPM algorithm

Adding noise markovian: $q(x_t|x_{t-1}) = N(\sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)I)$, for $q(x_t|x_0)$ see loss
Loss $\|\epsilon - \epsilon_\theta(\sqrt{\overline{\alpha}_t} x_0 + \sqrt{1-\overline{\alpha}_t}\epsilon, t)\|_2^2$
Sampl $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t, t)) + \sigma_t z$
where $\alpha_t = 1 - \beta_t$, $\overline{\alpha}_t = \prod_{i=1}^t \alpha_i$, $\epsilon, x_T, z \sim N(0, I)$, if $t = 1 \rightarrow z = 0$.
Deriv $\log p(x) \geq ELBO_\theta(x) = \mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0|x_1)] - \mathcal{D}_{\mathrm{KL}}(q(x_T|x_0) \| p(x_T)) + \sum_{t=2}^T \mathbb{E}_{q(x_t|x_0)}[\mathcal{D}_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t))]$, with $q(x_{t-1}|x_t, x_0)$ is Gaussian

Foundation models

(1 gen): general enc + task dec, NLP = ELMO BERT ERNIE, vision = ViT *(imgs are tokens)*, MaskedAutoEncoder/MAE *(reconstruct after removing some tokens)*, ViTpose and Sapiens *(humans)*, SegmentAnything/SAM, CLIP *(features as conditions, shared space with contrastive learning by pushing high cos sim. of similar text+img pairs)* DINOv2 *(self-superv, knowledge distilled from student/teacher with local/global view, backprop diff(t,s) through student and update teacher*, MassivelyMultimodalMaskedModeling/4M; **(2 gen)**: general model + task finetune, NLP = GPT-3, diffusion = DreamBooth *(text2img, easily finetune with new token to represent "my dog")* Zero-1-to-3 *(3D view synthesis by cond on img and camera params)* SiTH *(like 0123 but also works well on humans)*; **(3 gen)** NLP = LLMs.

10 Parametric Body Models

Challenge rotation, foreshortening, scaling, intra-category variation, aspect ratio.
2D Body Modeling *(all fully supervised)*
Pictorial Structures body made of cylinders and springs, minimize how well image matches + how much body deforms: $\mathbf{argmin}_L \sum_{i=1}^n m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j)$
Direct Regression ConvNet output (x, y) for certain body parts; **Iterative Error Feedback** refines on smaller sorrounding area; better with Gaussian heatmap.
Convolutional Pose Machines iteratively predicts whole heatmap w/ increasing receptive field; **OpenPose** adds Part Affinity Fields / associativity vectors *btmup*; **ViT-Pose** transformer encoder-decoder *tpdwn*.

3D face: Statistical Shape Models

Obtain vertex correspondences (i.e. registration) w/ bootstrapping; shapes S_i in full correspondence, $\overline{S}$ avg, $x_i = S_i - \overline{S}$ in mat X, covariance $C = \frac{1}{m} X X^\top = \frac{1}{m} U \mathrm{diag}(\sigma_i^2) U^\top$ (SVD), $c_i = U^\top x_i$, full model $S = \overline{S} + \sum_{i=1}^m c_i \sigma_i u_i$.

3D: SMPL Skinned Multi-Person Linear

$S = M(\overset{\text{pose}}{\theta}, \overset{\text{shape}}{\beta}) = W(T_P(\beta, \theta), J(\beta), \theta, \mathcal{W})$
$T_P(\beta,\theta) = \overline{T} + B_S(\beta) + B_P(\theta)$; $J(\beta) = \mathcal{J} \cdot (\overline{T} + B_S(\beta))$
$B_S(\beta, \mathcal{S}) = \sum_{i=1}^{|\beta|} \beta_n S_n$ $\quad \downarrow R$ flat rotation matrices
$B_P(\theta, \mathcal{P}) = \sum_{i=1}^{9K} (R(\theta) - R(\theta^*))_n P_n$
$V' = \{t_i\}_{i=0}^N = R_0(T_P(\beta, \theta) - j_0) + j_0 + t$
$t'_i = \sum_{k=1}^K w_{k,i} G'_k(\theta, J) t_i$ plain LBS
$t'_i = \sum_{k=1}^K w_{k,i} G'_k(\theta, J(\beta))(t_i + b_{S,i}(\beta) + b_{P,i}(\theta))$
W Linear Blend Skipping fun, factored model (shape and pose separate); $\mathcal{W} \in \mathbb{R}^{K'_{\leq 4} \times 3N_{=6890}}$ skinning weights; β 10 shape params; $\theta \in \mathbb{R}^{3K+3}$ joint angles for $K_{=23}$ joints in angle-axis form wrt to parent; $\overline{T} \in \mathbb{R}^{3N}$ rest pose; $B_S, B_P \in \mathbb{R}^{3N}$ shape/-pose blend shapes; $J : \mathbb{R}^{|\beta|} \rightarrow \mathbb{R}^{3K}$ joint mapper; $\mathcal{J}$ regression matrix learned w/ least squares; $\mathcal{S}$ shape displacements learned w/ PCA; $\mathcal{P}$ blend shapes / pose correctives learned; R_0, t global root orientation/translation (*not* wrt root!), j_0 root joint, V' template vertices; G'_k global transform; $\Phi = \{\overline{T}, \mathcal{W}, \mathcal{J}, \mathcal{S}, \mathcal{P}\}$.
Registration chicken-and-egg, so solve model and registration jointly. First **train** $\{\mathcal{W}, \mathcal{J}, \mathcal{P}\}$ by minimizing surface reconstr. error on multi-pose dataset; regularize $\mathcal{P}$ to 0, $\mathcal{W}$ to good initial, $\mathcal{J}$ to be local. Then train $\{\overline{T}, \mathcal{S}\}$ w/ PCA using multi-shape normalized pose dataset (so no pose contribs in shape space), separate models for men and women.
(1) Pose Parameter Training, linear comb of following 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-dep. 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 acceloration. **Clothing** can be represented with offsets from SMPL (hard to train), or with implicit surfaces (performance heavy).
Human Mesh Recovery (HMR): regresses θ, β and camera from 2D. Uses 2D reprojection loss and discriminator loss to tell if similar to priors.
Optimization Based Fitting / SMPLify: like HMR but uses 2D keypoint objective for reprojection loss. $\Theta^{t+1} = \Theta^t - \lambda\left(\frac{\mathrm{d}L_{reproj}}{\mathrm{d}\Theta} + \frac{\mathrm{d}L_{prior}}{\mathrm{d}\Theta}\right)$. But hand-crafted, slow to converge, sensitive to initialization.
Learned Gradient Descent (LGD)
$\Theta^{t+1} = \Theta^t + F(\frac{\mathrm{d}L_{\text{reproj}}}{\mathrm{d}\Theta}, \Theta^t, x)$, F is NN.
Learned iterative fitting (when IMU/EM sensors instead of img) iteratively fit model with RNN, reproject by simulating sensor measurements, minim. w/ LGD.

12 Implicit Surfaces and Neural Radiance Fields

3D representations: Voxels (Discretization of 3D space into grid $\mathcal{O}(n^3)$, low res),Points cloud (no topology/connectivity), meshes (vertices and faces, mixed res, self-inters, not general), Implicit (level set $S = \{x : f(x) = 0\}$ where f continuous).

Neural Implicit Representations (NIR)

Occupancy nets $f_\theta : \mathbb{R}^3 \times \mathcal{X} \to [0,1]$, 3D location and cond $\to$ occupancy; **Signed dist field: DeepSDF** $f_\theta : \mathbb{R}^3 \times \mathcal{X} \to \mathbb{R}$

Supervise NIR from other reprs:
(1) **Watertight meshes** sample random points then cross-entropy loss.
(2) **Points cloud: Implicit Geom Regulariz (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), when $\|\nabla f(x)\| = 1$, learned f is $\approx$ the distance to the suface. Loss: $\mathcal{L}(\theta) = \sum_{i\in I} |f_\theta(x_i)|^2 + \lambda \mathbb{E}_x (\|\nabla_x f_\theta(x)\| - 1)^2$.
(3) **Img/monocular vid: 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 $\to$ direction of the ray $\mathbf{w}$, 2. find the solution d for surface, $f_\theta(\mathbf{p} = \mathbf{r}_0 + d\mathbf{w}) = 0$, 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 $f_\theta(\widehat{\mathbf{p}}) = 0$ and $\widehat{\mathbf{p}} = \mathbf{r}_0 + \hat{d}(\theta)\mathbf{w} \to$ 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, need opacity.
NN $F_\theta(x, y, z, \theta, \phi) = (r, g, b, \sigma)$, σ density. (θ, ϕ) view direction, added at late stage of network, to avoid trivial sol.
Volume Rendering $\alpha = 1 - \mathrm{e}^{(-\sigma_i \delta_i)}$, $\delta_i = t_{i+1} - t_i$, Transmittance $T_i = \prod_{j=1}^{i-1}\left(1 - \alpha_j\right)$, Final ray color $c = \sum_{i=1}^N T_i \alpha_i c_i$.
Train $\min_\theta \sum_i \|\text{render}_i(F_\theta) - I_i\|^2$, sampling efficiency is a big issue.
Positional encoding pass low-dim 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), to model higher freqs. $\mathsf{PE} = \text{cat}[\cos k\mathbf{v}, \sin k\mathbf{v}]_{k=1}^{2^L}$, or $\gamma(\mathbf{v}) = [\cos(\mathbf{B}\mathbf{v}), \sin(\mathbf{B}\mathbf{v})]$ $\mathbf{B} \sim \mathcal{N}\left(0, \sigma^2\right)$, too big mapp bandwith $\sigma \to$ noisy img overfit.
App: SNARF (changes applied to human in canonical pose), Vid2Avatar (separate static bg from fg)

Gaussian splatting

Ellipsoids easy to render and to check for inters, and allow for thin structures.
Each has o occupancy, 3D μ pos, $\Sigma = RSS^TR^T$ rotat+diag scaling (semidef. pos). Need to splat on 2D: μ', Σ'.
$\alpha = o \cdot e^{-0.5(x-\mu')^T \Sigma'^{-1}(x-\mu')}$

13 Tutorials

Loss Mean Sq Err: $L_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^n (y_i - \hat{y}_i)^2$; Binary Cross Entropy $L_{\text{BCE}} = -\frac{1}{n}\sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$; Categorical Cross Entropy $L_{\text{CCE}} = -\sum_{i=1}^n \sum_{j=1}^C y_{ij} \log\left(\hat{y}_{ij}\right)$; Hinge loss $L_{\text{Hinge}} = \frac{1}{n}\sum_{i=1}^n \max(0, 1 - y_i \hat{y}_i)$.
Regularization (+ loss) L1 $\|\theta\|_1$, L2 $\frac{1}{2}\|\theta\|_2^2$
Dropout at training time disable node j in layer l w/ prob p: $r_j^{[l]} \sim \text{Bernoulli}(p)$, then $\tilde{y}^{[l]} = r^{[l]} \odot y^{[l]}$; at inference time do weight scaling $\tilde{\theta} = \theta p$.
Normalization $\mu = \frac{1}{n}\sum x_i$; $\sigma^2 = \frac{1}{n-1}\sum(x_i - \mu)^2$, $x_i^N = (x_i - \mu)/\sqrt{\sigma^2}$.
Batch norm. $x_i^N = (x_i - \mu)/\sqrt{\sigma^2 + \epsilon}$ where ϵ small for numerical stability, then $\tilde{x}_i = \gamma x_i^N + \beta$ where $\gamma, \beta \in \mathbb{R}$ learnable. In theory solves internal covariate shift, in practice makes smoother landscape, allows higher learning rate.
Feature modulation $c' = \gamma(s) \odot c + \beta(s)$ if we want to apply info from embedding s to embedding c.
AdaIN = normalization $\circ$ feat mod.
Activation ReLU $\max(0, x)$, leaky ReLU $\max(\alpha x, x)$, random lky ReLU $\alpha \sim \mathcal{U}(a, b)$
$\sigma(x) = \frac{1}{1+e^{-x}}$ $\quad \sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$
$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ $\quad \tanh'(x) = 1 - \tanh^2(x)$
Optimization GD $\theta \mathrel{-}= \eta \nabla_\theta L(\theta)$; SGD like GD but on just one element; Mini-batch GD on a batch $\nabla_\theta(\frac{1}{m}\sum_i L(\theta, x_i, y_i))$; Polyak Momentum $v \leftarrow \alpha v - \epsilon \nabla_\theta(\frac{1}{m}\sum_i L(\theta, x_i, y_i))$, $\theta \mathrel{-}= v$ but can diverge; Nesterov's Momentum $v \leftarrow \alpha v - \epsilon \nabla_\theta(\frac{1}{m}\sum_i L(\theta \underline{+ \alpha v}, x_i, y_i))$.
Learning rate AdaGrad divide by $\sqrt{\sum_i \|\text{previousGrad}_i\|_2^2}$; RMSProp exponential weighted average; Adam = AdaGrad + Momentum SGD.
Bagging/Ensemble models trained separately; they have minima often connected by constant loss. Fast Geometric Ens. trains model normally for 80% of time, then final 20% w/ cyclic lr, then ensemble (but many models!). Stochastic Weight Avging: just one model and keep running avg of parameters when lr is min in cycle.

14 Appendix

Jacobian$_x(y)$ (numerator layout) $= \frac{\mathrm{d}y}{\mathrm{d}x} = \begin{bmatrix} \frac{\mathrm{d}y_1}{\mathrm{d}x_1} & \cdots & \frac{\mathrm{d}y_1}{\mathrm{d}x_n} \\ \frac{\mathrm{d}\ddot{y}_m}{\mathrm{d}x_1} & \cdots & \frac{\mathrm{d}\ddot{y}_m}{\mathrm{d}x_n} \end{bmatrix} \in \mathbb{R}^{m\times n}$

Divergence

Kullback-Leibler $D_{\text{KL}}(p\|q) = \int p(x)\log\frac{p(x)}{q(x)}dx$; Jensen-Shannon $D_{\text{JS}}(p\|q) = \frac{1}{2}D_{\text{KL}}\left(p\middle\|\frac{1}{2}(p+q)\right) + \frac{1}{2}D_{\text{KL}}\left(q\middle\|\frac{1}{2}(p+q)\right)$; Cross-entropy $H(p, q) = -\mathbb{E}_p[\log q]$; if $p = \sum_i \delta_{x_i}$ empirical, $H(p, q) = \text{NLL}(q) = -\mathsf{KL}(p\|q) + H(p)$.

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

Reparam trick

$z \sim \mathcal{N}(\mu, \mathrm{diag}(\sigma_i^2)) \Leftrightarrow \epsilon \sim \mathcal{N}(0, I), z = \mu + \sigma \odot \epsilon = f(x, \epsilon, \theta) \implies \mathbb{E}_{z\sim q(z|x)}[f(z)] = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[f(\mu + \sigma \odot \epsilon)]$

Integral change of vars

Given $x = g(u)$, $\int_{x_0=g(a)}^{x_1=g(b)} f(x)dx = \int_{u_0=a}^{u_1=b} f(g(u))g'(u)du$.
For 2D given $x = g(u, v)$, $y = h(u, v)$, $\iint_R f(x, y)dxdy = \iint_G f(g(u, v), h(u, v))J(u, v)dudv$