MIL
- embedding-based → lack of interpretability
  - ↳ aggregates embeddings then predicts label
- instance-based → propagate error
  - ↳ predicts each labels then aggregates them

Attention-based MIL

$$Z = \sum_{k=1}^{K} a_k h_k \quad , \quad a_k = \frac{e^{w^T \tanh(V h_k^T)}}{\sum_{j=1}^{K} e^{w^T \tanh(V h_j^T)}}$$

h_k: embedding feature map

Attention model

$$\alpha_i = \frac{e^{\langle q_i, k_i \rangle}}{\sum_j e^{\langle q_i, k_j \rangle}}$$

α_i

$\alpha_1 V_1 + \alpha_2 V_2 + \cdots + \alpha_n V_n$

patch 1 | patch 2 | ----- | patch n

W_K W_Q W_V

d × [Key] [Query] [Value]

K_1 Q_1 V_1

$\text{Softmax}(Q_i^T K) V$

order $\sim O(n^2)$

not efficient for whole slide images!

Dual-Stream MIL

like attention model but just calculate attention for main patch. q_m.

2 Streams
- uses MIL max pooling for classification in first stream.
- uses from q of main patch in classification to generate attention of other patches to q_m.

Graph Convolutional Network

$$h_i^{(l+1)} = \sigma\left(h_i^{(l)} W_0^{(l)} + \sum_{j \in N_i} \frac{1}{c_{ij}} h_j^{(l)} W_1^{(l)} \right)$$

GNN / GCN
- Node classification
- Graph classification
- Link classification

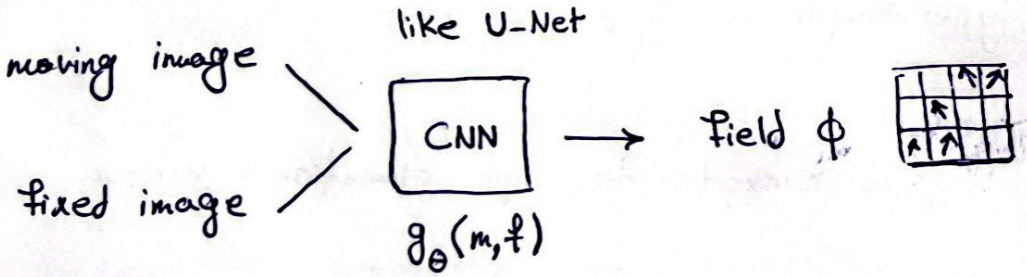
Registration

- optimization problem for each 2 images / old approach
  - ⊗ m x n pixels ~ 9000 sec / each pair time consuming
  - ⊗ affine transformation (translation, scaling, rotation)
    it is not good when deformation exists.

$$\phi_{m,f} = \arg\min_{\phi} \underbrace{\| m \circ \phi - f \|}_{\text{image match}} + \underbrace{\lambda \, Reg(\phi)}_{\text{Smooth field (regualatization)}}$$

- it may give local min.

avoid from discontinuous and large field.



- unsupervised (voxelmorph) ~ $L = \| m \circ \phi - f \| + \lambda \, Reg(\phi)$
  ↓ loss
- fast for new image pair

$p' = p + u(p)$ ~ for being differentiable.

Contour landmarks in image pair

- maximum dice score of these landmarks as loss or validation metric.

$$\text{Dice score} = 2 \times \frac{\text{area of overlap}}{\text{area of Union}}$$

Synth Morph ~ uses from synthetic data

- for generalization of registration model.
  - Create imaginary data that are able to register.
  - OOD generalization.

Hyper Morph ~ λ as an input

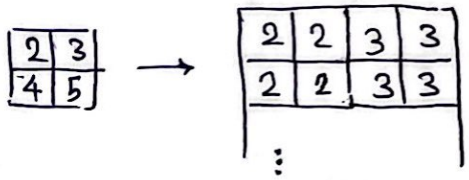
- λ converts to high dimentional vector.
- Set an arbitrary λ in inference for custom dataset
- if variation of dataset is high you should Set a small λ and vice versa.

Segmentation , Object detection , Instance Segmentation.

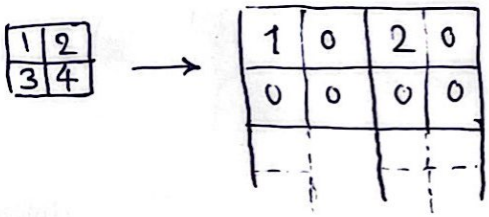
Semantic segmentation ~ Vnet downsample & upsample

- upsampling approaches ~

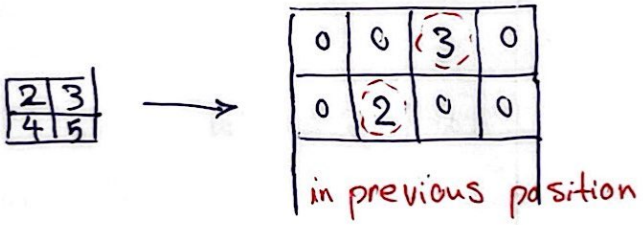
unpooling
- nearest neighbor



- bed of nails



Max unpooling ~ remember which element was max.



learnable - Convolution transpose ~ deconvolution

$$\begin{bmatrix} A_1 & A_2 & \cdots & A_K \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{bmatrix} = \sum_{i=1}^{K} x_i A_i$$

* information loss is in latent layer (embeddings)
so we consider skip connection. → in each step

Why U Net needs small dataset to train?

- Fully Convolutional network has small number of weights
- 512×512 loss terms
- we can use a lot of transformation to augment data.
- High resolution convert to many patches.

object_detection

- Sliding window: time consuming
  too many forward pass (CNN)
- R-CNN: region proposal ~ 2k
  2k forward pass in Conv Net
  ~ time consuming
- Fast R-CNN: one Conv Net to extract feature map – ROI from a proposal method – Crop and resize features – CNN
  Class - box offset
  - ROI pool ~ Snapping
  - ROI align ~ No Snapping
    bilinear interpolation

$$f_{x,y} = \sum_{i,j=1}^{2} f_{ij} \max(0, 1-|x-x_i|) \times \max(0, 1-|y-y_i|)$$

    the max pooling

Faster R-CNN

for each pixel (in feature map) ~> k boxes ≡

~ insert Region Proposal Network ~ 300 most confident proposals

ROI pooling → Class, box offset

4 loss in training :
- RPN object/not object
- RPN regress box coordinates
- Final Classification
- Final box coordinates

Two stage object detector
1. RPN
2. ROI pooling, class, box offset

Single-stage object detector → YOLO, SSD, RetinaNet

~ for each pixel $(5 \times B + C)$
- $(dx, dy, dh, dw,$ confidence) — obj/not obj
- B → aspect ratios
- C → classes

Mask R-CNN ~ like Faster R-CNN

~ with one more head for mask prediction.

- we should find also mask object in feature map the ROI align it and enforce mask in mask prediction head.

Interpretability

- Attribution methods → assigns "attribution score" to each input pixel ~ Saliency map

debug
reliability
verify
discovery
right to explanation

$$\text{Saliency}(x) := \nabla_x f(x) = \frac{\partial f(x)}{\partial x}$$

- Saliency maps are noisy!

Hypothesis 1: Saliency maps are truthful
noise is important to make decision

Hypothesis 2: gradients are discontinuous

DNN uses piece-wise linear functions
(ReLU activation, maxpooling, ...)

Sudden jumps in $f(x)$ over small changes in input.

- Solution: Smooth the gradient

$$\text{SmoothGrad}(x) := \frac{1}{n} \int_1^n \frac{\partial f(x^*)}{\partial x^*}$$

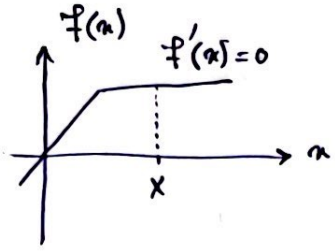
$$x^* = x + N(0, \sigma^2)$$

gaussian noise

this solution smooth the jumps in $f(x)$

Hypothesis 3: $f(x)$ saturates

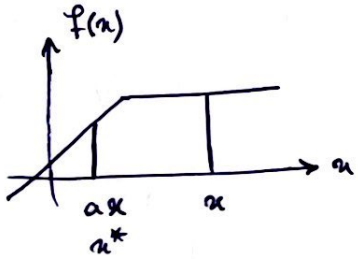
Small derivative locally. maximum possible activation.



but a feature may have strong effect globally.

- Solution: Interior gradient

Scale back the x. $x^* = \alpha x \quad 0 < \alpha \leq 1$



$$\text{IntGrad}(x) := \frac{\partial f(x^*)}{\partial x^*}$$

Backprop-based

1. Deconvnet map feature pattern to input space

pass reconstructed signal through ReLUs.

removing noise by removing negative gradient

Backprop-based

2. Guided backpropagation

Combine Deconvnet with back propagation

Zero out the cells which became zero in forward pass.

removing negative gradient + Consider forward activation

Evaluating Attribution methods

- Qualitative
  - Coherence ~ discriminative features
  - Class sensitivity ~ sensitive to class labels
- Quantitative
  - selectivity ~ removing feature with high attribution
  - ROAR/KAR ~ remove and retrain - keep and retrain

- selectivity ~ remove feature with high attribution (salient feature) cause large decrease in class probability.

if saliency features are very strong we get a higher AOC (area over curve)

- ROAR/KAR ~ measure how the performance of classifier changes as specific features are removed based on attribution method.

Sensitivity may be not accurate

ROAR: replace N% of pixels which are most important

KAR: replace N% of pixels whis are least important

Federated learning

data cannot be centralized ~ motivation

- too costly
- too sensitive ~ privacy

if we have centralized dataset, the use standard machine learning settings.

Each party learning on its own? No.

1. local dataset of each party may be too small.
   - over fitting
   - non-statistically significant result on medical data
2. local dataset may be biased.
   - not representative of target distribution.

Federate learning → collaboratively train a ML model while keeping the data decentralized

distributed-learning: data is centrally stored
- train faster
- data is distributed uniformly at random across workers.

FL → data is naturally distributed
- not independent and identically distributed
- imbalanced ~ total number of each dataset are not equal

challenges in FL

- privacy constraints
- possibly limited reliability / availability of participants
- malicious parties ~ achieving robustness

FL
- Cross-device ~ massive number of parties $\simeq 10^{10}$
  - ↳ has above challenges + small dataset per party
- Cross-silo ~ 2-100 parties
  - ↳ this is good

FL
- Server-orchestrated ~ server-client communications
  - global coordination, global aggregation
  - server is a single point of failure, bottleneck
- Fully decentralized ~ device-device communication
  - local aggregation
  - Scales to a large number of devices
  - like graph

notation

set of K parties

party k ~ dataset D_k, n_k: number of data

$D_1, D_2, \dots, D_K \quad n = \sum_k n_k$

problem: $\min_\theta F(\theta; D)$ → parameter of model

$$F(\theta; D) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(\theta : D_k)$$

$$F_k(\theta, D_k) = \sum_{d \in D_k} f(\theta; d)$$ → Cost function

FedAvg or local SGD

FedAvg

ρ : Client sampling rate

L : number of step in ClientUpdate

$L=1, \rho=1 \rightarrow$ Classic parallel SGD

updates are aggregated and model synchronized at each step.

* this algorithm is for server-orchestrated

FedAvg for fully decentralized setting

- undirected graph

$G = (\{1,2,\dots,K\}, E)$

$\{k,l\} \in E \rightarrow k$ and l can exchange messages.

W^{KxK} matrix : weight 0 or 1 if $\{k,l\} \notin E$ then 0

$\theta = \{\theta_1, \theta_2, \dots, \theta_K\}$ for each party

$$[W\theta]_k = \sum_{l \in N_k} W_{k,l}\, \theta_l$$

k : for party k

N_k : neighbors of k

$\rightarrow$ local aggregation among neighbors of k

Convergence rate depends on topology ~ more connected faster

two steps in each round

1. local update with $\theta_k^t - \frac{n_k}{B} \sum_{d \in B} \nabla f(\theta_k^t : d)$
2. local aggregation

Dealing with Non iid Data problem: Client Drift

Strong non-iid

x^* is not average of x_1^* and x_2^*

- Scaffold
  - correction terms $C_1, C_2, \dots, C_K$ for each party
    update C as well as θ in each round
  - Convergence rate is more than parallel SGD

FedAvg slower than parallel SGD for strongly non iid data
~ large G

* learning from non-iid data is difficult because
  each party wants the model to go in particular direction

personalized model θ_K ~ Meta Model

- global model which easily adapts to each party

$$F(\theta_1, \dots, \theta_K; D) = \frac{1}{K} \sum_{K=1}^{K} F_K(\theta_K; D_K) + \underbrace{\frac{\lambda}{2K} \sum_{K=1}^{K} \|\theta_K - \frac{1}{K}\sum_{l=1}^{K}\theta_l\|^2}_{\text{regularization}}$$

$$F(\theta; D) = \frac{1}{K} \sum_{K=1}^{K} F_K(\theta - \alpha \nabla F_K(\theta); D_K)$$

ViT: Vision Transformers

- naive solution → flatten image to height x width vector
  16x16 image → 256 vector
- transformer time complexity ~ $O(n^2)$ 256^2 too much!
- nice image representation
- Semi-supervised learning ~ linear classification on iGPT representation

ViT is more general, they have less inductive bias.

CNN {
- locality
- two dimensional neighborhood structure
- translation equivariance

ViT {
- image patches 16x16 words
- position embedding interpolation during fine-tuning + upscaling

→ {
- Performs worse than CNN when there's not as much data
- Performs better than CNN when there's lots of data

inductive bias {
- ViT learns best representation for a task
  - lack of inductive bias
- CNNs are biased from beginning. they have inductive bias.
  - locality ~ they generate feature using local pixels.

Global receptive field

- CNNs have a receptive field limited by kernel size.
- ViTs have global receptive field
- ViT shows distance between Q and K (two tokens)

mean attention distance is very high when a token really want to attend to another token.

ViT Cons

1. patches not very fine grained.
2. Some issues in task like segmentation, fine grained classification.

need exact baundary

you should scale up data and model size in vision transformer, to get better performance.

* lack of inductive bias is bad when we don't have enough data.

Diffusion Model

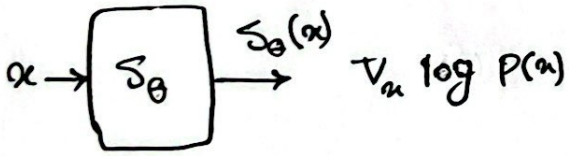
gradient of <u>data distribution</u>

$\nabla_x \log P(x)$: gradient ascent

Score function (matching)

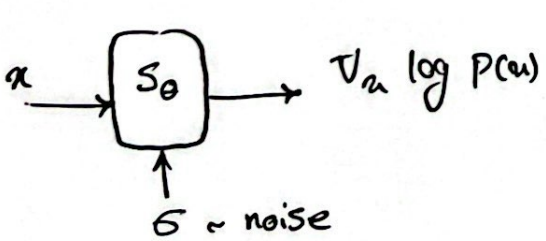
$$\mathbb{E}_{P(x)}\left[\text{trace}\left(\nabla_x S_\theta(x) + \frac{1}{2} \| S_\theta(x) \|_2^2 \right) \right]$$

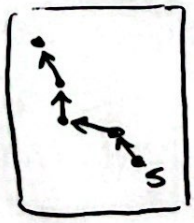
loss function for training of S_θ network



$\mathbb{R}^{28 \times 28}$

non-sense gradient ~ starting point is out of <u>data manifold</u>





Langevin dynamics (noise)

$$\tilde{x}_{t+1} \leftarrow \tilde{x}_t + \frac{\varepsilon}{2} S_\theta(\tilde{x}_t) + \sqrt{\varepsilon}\, z_t \qquad z_t \sim N(0, I)$$

$$\sigma_1 > \sigma_2 > \cdots > \sigma_K$$