Devising an explicit algorithm based on simple rules is difficult! L1 reg: $\ell = err(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$ favours few non-0 coefs, L2 favours small coefs
under-fitting → high bias (high training, high test error) → add features, decrease regularization term λ, increase degree of polynomial)
over-fitting → high variance (low training, high test error) → get more data, remove features, increase regularization term λ, decrease degree of polynom)

Linear Transformations

$$\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \underbrace{\overbrace{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}}^{\text{translation vectors}} \overbrace{\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{\text{direction vectors}} \overbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{\text{scaling matrix}}}_{T_{ItW}} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

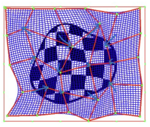
If there is more than one world, then:

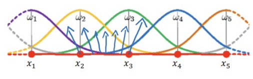
$$\begin{pmatrix} x_A \\ y_A \end{pmatrix} = \mathrm{T}^{A}_{WtI} T_{BtA} T^{B}_{ItW} \begin{pmatrix} x_B \\ y_B \end{pmatrix}$$

In 3D, there are identity (0 dof) rigid (translation and rotation, 3+3 dof), similar (scaling, 3+3+1 dof), affine (shear 12). In affine, if two lines are || , after affine T, its still true.

$$\text{shearing} = \begin{bmatrix} \cos\omega & \sin\omega & 0 \\ -\sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Non-linear Transformations





Embed image onto grid, prescribe motion at each grid point (red) and underlying blue grid is as a result of linear interpolation.
Medical Applications: Cardiac Motion and Respiration Motion tracking, Multi-modal Image Fusion, Pre- and Post-op comparison
Intra-Subject Registration: create an atlas and transform.

Challenges in Semantic Segmentation (every pixel in an image belongs to a class)

- *noise* – high-frequency pixel variability (not relevant/may obscure target)
- *partial volume* – quantized version of object (pixels may contain mix of two objects and both contribute to pixel value) and object may be elevated (unclear where to begin/end object)
- *intensity inhomogeneities* – varying contrast and intensity differences across the image plain
- *anisotropic resolution* – (not isotropic, where voxels are cubes) causes ↓ clarity in coarse dims
- *imaging artifacts* – implants may interfere with imaging modality
- *limited contrasts* – different tissues may have similar physical properties and leak boundaries
- *morphological variability* – variability in physiological conditions or imaging modalities

Pitfalls in Segmentation Evaluation

- *Structure Size* – equal differences between small and big structures change spatial overlap lots
- *Structure Shape* – spatial overlap metrics are unaware of complex shapes
- *Spatial alignment* – HD & DSC & IoU don't capture object centre point alignment
- *Holes* – Boundary-based metrics ignore overlap between structures
- *Noise* – Affects HD as it is spiked by a far away FP
- *Empty Label-Maps* – scores of 0 or NaN for each method with combo of empty ref. or predict.
- *Resolution* – same prediction shapes at different resolutions give different results
- *Over vs Under-segmentation* – for equal HD, DSC may be better for over than undersegment.

Segmentation Methods

- *Intensity Segmentation:* (hist) $\times$ regions must be homogenous, leakages, threshold loc. Hard
- *Region Based*: (start from seed) $\times$ requires user points, leakages, assumed homogeneity
- *Atlas Based:* (averaged templates) *Registration:* mutate multiple atlases into target and fuse labels (majority voting) (this saves pdf indicating contention between sources). $\surd$ robust, accurate, automatic $\times$ comput, expensive, poor for abnormalities, not for tumour segmentation.
- *Random Forests:* different modalities of 1 image, construct a tree to classify a pixel based on rules. Ensemble it by averaging answer of many trees. $\times$ no hierarchal features $\surd$ || & accurate

Output size = , # of param $C \times K \times K$

Expert Gold Standard: $\times$ training, tedious, intra (same dude) + inter (diff dude) observability variability, $\surd$ multiple segmentations, agreement can be quantified

- specificity = $\frac{TN}{N} = \frac{TN}{TN+FP}$
- $F_\beta = (1+\beta^2)\frac{Precision \cdot Recall}{(\beta^2 \cdot Precision)+Recall}$
- Jaccard Index/ IoU = $\frac{|S_g \cap S_p|}{|S_g \cup S_p|} = \frac{DSC}{2-DSC}$
- Dice Sim. Coeff. = $2\frac{|S_g \cap S_p|}{|S_g|+|S_p|} = F_1$
- Volume Sim = $1 - \frac{||S_g|-|S_p||}{|S_g|+|S_p|} = 1 - \frac{|FN-FP|}{2TP+FP+FN}$

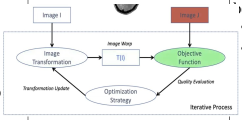
surface distance measure

- Hausdorff Distance = $\max(h(A,B), h(B,A)), h(A,B) = \max_{a\in A}\min_{b\in B}||a-b||$
- Average Surface Distance (create map and swap) $\frac{d(A,B)+d(B,A)}{2}, d(A,B) = \frac{1}{N}\sum_{a\in A}\min_{b\in B}||a-b||$

Multi-scale processing: 4 layers of 5^3 kernels followed by 1^3 kernel for classification. Multiple pathways for different sized snippets of the image. Then we concat. Feature maps from both pathways
Vision transformers: split image into patches, encode location, get hidden feature after convolutions, linear layer and pass through attention network similar to nlp. Upsample in U-net fashion with connections.

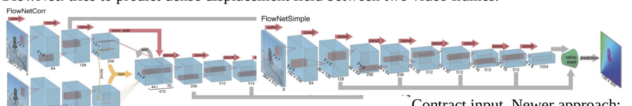
- (up)pooling and max (un)pooling with stored spatial location
- Transposed Convolution:
  $$(M-1)\times S - 2P + D\times(K-1) + P_{output} + 1$$
- Convolution: $\left\lfloor \frac{M+2P-D*(K-1)-1}{S} \right\rfloor + 1$
- Atrous spatial pyramid pooling: repeated max-pooling and striding reduces spatial resolution of the resulting feature map
- Padding during upsampling may introduce artifacts

Registration iterative process



Gradient descent or SGD.

FlowNet: tries to predict dense displacement field between two video frames.



Contract input. Newer approach: forces a pixel-wise correlation.
Then upsample for p.w. displacement. Train w/ flying rendered chairs (you know Ground Truth). Next evolution of **FlowNet2.0** passes image through once, applies the translation, and passes it into the next layer for further fine-tuning. In parallel, there is detailed matching, then concat all.
Optical Flow with Semantic Segmentation and Localized Layers: segment 'Things', 'Planes' & 'Stuff'. Then perform flow estimation on segmented objects for a sharper answer.
Non-rigid Image Registration Using Multi-scale 3D CNNs: randomly deform an image, then train a model to predict your known deformation. To use this network, you need to slide this network across the image and generate for each pixel a displacement vector.
Spatial Transformer Networks: takes feature map (original or pre-processed) and predicts transformation and transform the image according to this transform map. There is a localisation net which trains θ to then deform the grid.
Unsupervised Deformable Image Registration: Two images are fed into an NN to predict deformation. Then feed into spatial transformer, this transforms input and calculates sim. metric.
Voxel Morph: u-net architecture which produces a dense displacement field. Then it uses the spacial transformer to warp the image to the fixed image then minimise the loss to the network.

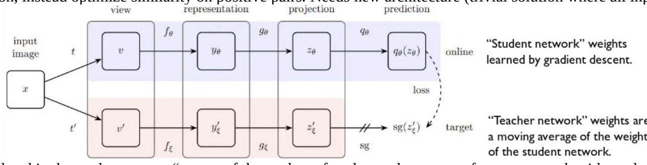
Objective: $C(T) = D(I \circ T, J)$ (**T**ransformation, **D**issimilarity measure, (**J**) Fixed image, $(I \circ T)$ Moving Image
Optimization: $\hat{T} = \arg\min_T C(T)$
Mono-modal Registration: Image intensities are related by a (simple) function. Assumption: the identity relationship between intensity distributions. Not good when brightness changes and subtraction is no longer the best metric.

- Sum of squared differences: $D_{SSD}(I \circ T, J) = \frac{1}{N}\sum_{i=1}^{N}(I(T(x_i)) - J(x_i))^2$
- Sum of absolute differences: $D_{SAD}(I \circ T, J) = \frac{1}{N}\sum_{i=1}^{N}|I(T(x_i)) - J(x_i)|$
- Correlation Coefficient: $$D_{CC}(I \circ T, J) = -\frac{\overbrace{\frac{1}{N}\sum_{i=1}^{N}(I(T(x_i)) - \mu_I)(J(x_i) - \mu_I)}^{cov(I,J)}}{\underbrace{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(I(T(x_i)) - \mu_I)^2}}_{\sigma_I}\underbrace{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(J(x_i) - \mu_J)^2}}_{\sigma_I}}$$

Multi-modal Registration: Image intensities are related by a complex function or statistical relationship. Measures "When are these two images the most statistically aligned". To avoid local minimas: increase dof, or gaussian smoothing. May require linear interpolation

- Intensity Histograms: plot intensities of both images on x and y, discretize histogram with bins. A Registered image will have more clustered regions (identity will be line $x = y$)
  - $p(i,j) = \frac{h(i,j)}{N}$ at a point, in one image i and other image j counts in the histogram. $p(i) = \sum_j p(i,j)$ sim for $p(j)$
  - Shannon Entropy: $H(I) = -\sum_i p(i)\log p(i)$ low value if every pixel has the same value, or high if randomness
  - Joint Entropy: $H(I,J) = -\sum_i\sum_j p(i,j)\log p(i,j)$ measures how clustered a space is, and minimising that entropy is a good criterium
  - Mutual Information: $MI(I,J) = H(I) + H(J) - H(I,J)$ describes how well one image can be explained by another image. $MI(I,J) = \sum_i\sum_j p(i,j)\log\frac{p(i,j)}{p(i)p(j)}$ with dissimilarity: $D_{MI}(I \circ T, J) = -MI(I \circ T, J)$
  - Normalized Mutual Information: $NMI(I,J) = \frac{H(I)+H(J)}{H(I,J)}$ with dissimilarity similar to above

Contrastive-based learning (SIMCLR): teaching the network to recognise meaningful pairs of images. Idea: create multiple versions of the same image and teach the network to recognise the correct pair. An encoder outputs hidden state h and the small MLP gets a smaller representation (in high dimensional space, comparing similarities is harder, and not necessarily robust). Augmentation Pipeline: Needs to reflect what information the model should disregard and what it should focus on. Needs to be difficult, otherwise, trivial features may be learnt. Normalised temperature contrastive loss: $sim(\vec{u},\vec{v}) = \frac{\vec{u}^\top\vec{v}}{||\vec{u}||\cdot||\vec{v}||}$ with Loss $\ell_{i,j} = -\log\frac{\exp(sim(\vec{z}_i,\vec{z}_j)/\tau)}{\sum_{k=1}^{SN}\vec{1}_{[k\neq i]}\exp(sim(\vec{z}_i,\vec{z}_k)/\tau)}$, τ temperature controls how much to penalise hard negatives (lower temp penalises more) $\mathbf{1}_{[k\neq i]}$ evaluates to 1 iff $k \neq i$. Triplet Loss: where x is the image, x^+ is the positive image, and x^- is a random negative $\mathcal{L}_{triplet}(\vec{x},\vec{x}^+,\vec{x}^-) = \sum_{x\in\mathcal{X}}\max(0, ||f(x) - f(x^+)||_2^2 - ||f(x) - f(x^-)||_2^2 + \epsilon)$.
You evaluate the model with *Fine-tuning* (all params) or *linear probing* (freeze encoder). Batch size should be high (4096). Since large batch size is expensive: *BYOL*: (more robust to small batch-size) removes the need to use negative pairs in the loss function, instead optimize similarity on positive pairs. Needs new architecture (trivial solution where all inputs go to 0)



Prediction head in the student asserts "output of the student after the pred = output of target network without the pred.". It matches the output of online and target. DINO uses visual transformers as encoders and use the attention to visualise the attention map of the network when they create the self-supervised encoding; creates self-supervised segmentation maps.

Generative-based learning: teaching the network to reconstruct an image from a corrupted version.

- Vision Transformers: as above, non-overlapping patches learn representation for patches.
- Masked Auto-encoders are scalable vision learners: divide the image in multiple sub-patches and randomly mask some of the patches. Feed it through the vision transformer. The encoder gets a representation f.e. patch and then fills in the blanks with a mask token you learn during training. The decoder predicts each patch (this is smaller since encoder does most of the work) Loss: $MSE = \sum(\hat{x}_i - x_i)^2$, where i is the pixel index. Masking ratio of >75% is best.

Joint-Embedding Prediction (i-Jepa): a mix of both.
Here, we pass available patches onto the encoder. In ||, sample several other regions and pass it through the same encoder. The task here, is to predict the encoded representation (instead of the output image) instead of wasting compute reconstructing ever single pixel in the image, we are going to reconstruct the important information which there should be if the network is trained summarised. Loss: $\frac{1}{M}\sum_{i=1}^{N} D(\hat{s}_y(i), s_y(i)) = \frac{1}{M}\sum_{i=1}^{M}\sum_{j\in B_i}||\hat{s}_{y_j} - s_{y_j}||_2^2$.

Inverse Problem: recover original image e.g. Inpainting, De-blurring/noising, Super-resolution. $y = Ax + n$, Classical Approach: least squares: (D *data-consistency term*) $\arg\min_x \mathcal{D}(Ax, y)$, $\mathcal{D}(Ax,y) = \frac{1}{2}||Ax - y||_2^2$ for which the solution is $\hat{x} = (A^\top A)^{-1}A^\top y$. Ill-posed Problem: slight differences to the image may yield drastically different results. $\arg\min_x \mathcal{D}(Ax,y) + \lambda\mathcal{R}(x)$ with $\mathcal{R}(x) = \frac{1}{2}||x||_2^2$ (*R prior knowledge on x – regularization term, with reg. Param λ*). Therefore $\hat{x} = (A^+\lambda I)^{-1}A^\top y$ which makes it better conditioned and noise suppression.

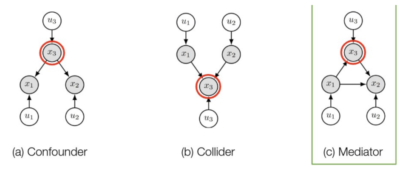
- Tikhonov Regulizer: $\mathcal{R}(x) = ||\nabla x||_2^2$ (L-2 norm of gradient)
- Total Variation Regularization: $\mathcal{R}(x) = ||\nabla x||_{2,1} = \sum_{i=0}^{n}\sqrt{\sum_d(|x|_i^{(d)})^2}$ (L-1 norm of grad)
- Wavelet Regularization: $\mathcal{R}(x) = ||\Psi x||_1$ (apply a sparsifying transformation and min that)
- Dictionary Learning: patches then encode each patch by a sparse combination of (e.g.) wavelet denoising and combine these to reconstruct the original image

Instead of choosing $\mathcal{R}$ a-priori based on a simple model of the image, learn $\mathcal{R}$ from training data
Model Agnoistic: (don't use anything about the forward transformation) Partly-model agnostic: upsample with a classical algorithm like linear interpolation which estimates the known transformation then train a model to remove difference between upsample on original.
Decoupled Approach: Deep-Proximal Gradient: estimate with Gradient Descent: set $\hat{x}^{(1)}$ and stepsize η, compute $z^k = \hat{x}^{(k)} + \eta A^\top(y - A\hat{x}^{(k)})$, $\hat{x}^{(k+1)} = \arg\min_x ||z^k - x||_2^2 + \eta\lambda\mathcal{R}(x)$ *we can replace the second term with a NN with a GAN:* which returns 0 for realistic images (on the manifold) with generator and discriminator. *Assume you know A*
$\min_\mathcal{G}\max_\mathcal{D} V(\mathcal{G},\mathcal{D}) = \underbrace{\mathbb{E}_{x\sim p_{data}(x)}[\log\mathcal{D}(x)]}_{\text{real}} + \underbrace{\mathbb{E}_{x\sim p_z(x)}[\log(1 - \mathcal{D}(\mathcal{G}(z))]}_{\text{synthetic}}$. We then redefine the problem w.r.t $\mathcal{G}$ and $\mathcal{D}$: $\mathrm{R(x)} = \begin{cases} 0 & x \in range(\mathcal{G}) \\ \infty & otherwise \end{cases}$. Here, we restrict the space to the image manifold so $\hat{x} = \arg\min_{x\in range(\mathcal{G})}||y - Ax||_2^2 = \mathcal{G}(z)$ and $\hat{z} = \arg\min_z ||y - A\mathcal{G}(z)||_2^2$
Unrolled Optimization: assuming $\mathcal{R}(x)$ is differentiable: the first term is GD for data consistency, & second is GD for regularization. $\hat{x}^{(k+1)} = \hat{x}^{(k)} + \eta A^\top(y - A\hat{x}^{(k)}) - \eta\nabla\mathcal{R}(\hat{x}^{(k)})$. Then replace GD for reg. with an NN.

Post-upsampling super-resolution: directly upsampled LR image into SR; requires learnable upsampling layers. √ fast and low mem, $^\times$ network has to learn entire upsampling pipeline, network typically limited to a specific up-sampling factor. **Pre-upsampling super-resolution**: use traditional upsampling (linear iterp.), refine upsampled using a CNN. √ upsampling operation is performed using interpolation, the correct smaller details, can be applied to a range of upscaling factors and image sizes $^\times$ operates on SR image, thus requires more comp & mem. **Progressive upsampling super-resolution**: use a cascade of CNNs to progressively reconstruct higher-resolution images, at each stage, upsample to higher resolution. √ decomposes task into simpler ones, reasonable efficiency $^\times$ difficult to train deep models. **Iterative up-and-down sampling super-resolution**: alternate between upsampling and downsampling operations, mutually connected up-and down-sampling stages √ has shown superior performance as it allows error feedback, easier training of deep networks.
Pixel-wise loss function (either L1 (non diff) or L2): $\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}|x_i - \hat{x}_i|^p$ or Huber loss function: $\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\delta(x_i - \hat{x}_i)$, $\delta(a) = \begin{cases} 0.5a^2 & \text{for } |a| \leq 1 \\ |a| - 0.5 & otherwise \end{cases}$. However, compute the Perceptual loss: loss on output θ of an intermediate layer l of a pre-trained network since the network has been pre-trianed on a task to classify objects e.g. then activations describe the image in more perceptual terms. $L = \frac{1}{M}\sum_{i=1}^{M}(\theta_i^{(l)}(x) - \theta_i^{(l)}(\hat{x}))^2$
Total Variation: $L = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\sum_d(|x|_i^{(d)})^2}$ assumes absolute value of gradient of image is low (piecewise constant). For 2D images: $L = \frac{1}{n_1 n_2}\sum_{i=1}^{n_1-1}\sum_{j=1}^{n_2-1}\sqrt{|x_{i-1,j} - x_{i,j}|^2 + |x_{i,j-1} - x_{i,j}|^2}$. GAN-based loss functions: $\min_\mathcal{G}\max_\mathcal{D}\mathbb{E}_{I_{HR}\sim p_{train}(x)}[\log\mathcal{D}(I_{HR})] + \mathbb{E}_{I_{LR}\sim p_\mathcal{G}(I_{LR})}[\log(1 - \mathcal{D}(\mathcal{G}(I_{LR}))]$ with aim to discern real high resolution and artificially upsampled. The discriminator output is loss function: $\mathcal{L} = -\log\mathcal{D}(G(I_{LR}))$.
Deep Image Prior: represent an image not by its pixel values, but by the weights of a vector (like latent encoding): start with random-initialized network, given image x, its parmaeters w are recovered by solving $\min_w ||x - \Psi(z_0; w)||^2$. It is shown that generating noise is hard, and generating real images is easy. For in-painting, we only reconstruct visible pixels and implicitly infer the pixels masked out: $\min_w ||m \bigodot (x - \Psi(z_0; w))||^2$ We can then reconstruct the iamge by fitting a noise vector into the image, and reconstruct the image.

Image Reconstruction: x-ray scan captures information by measuring the x-ray energy on the opposite side of emitter. The rotated scanner produces a Sinogram (ez to go from CT → Sinogram (forward) but inverse is not). (Same goes for K-Space ← MRI). The problem becomes "Recover an image $x \in \mathbb{K}^{N_x}$ from a set of observations $y \in \mathbb{K}^{N_y}$ which are corrupted by a noise $n \in \mathbb{K}^{N_y}$. Reconstruction approach: *Image domain*: use as much phyiscs knowledge as possible (forier transformation) and use nn to remove artifacts, or k-space domain for nn to create new frequency space then inverse forier transform to upsample.

Loss function: e.g. MSE, and learn: $\theta^{(k+1)} = \theta^{(k)} - \tau\frac{\partial\mathcal{L}(\theta,x)}{\partial\theta}|_{\theta=\theta^{(k)}}$. for **low sample size**, denoize through convolutions, then match with forier transform into signal domain to check the data is consistent (repeat blocks 5 times). For **Coarse Voxels** acquire thick slices through the organ (e.g. z-dimension) construct a simulation from a high quality image for forward process and generate pairs of H and L resolution data and train nn to absorb all images to high resolution (original). A residual network works well for this.



Causality: is the relationship between cause and effect. **Causal** $X \to Y$ (predict the effect (Y) from the cause (X)) and **Anti-Causal** $Y \to X$ (predict the cause (Y) from the effect (X)). Factors are **endogenous** when it is caused by a variable in the model, **exogenous** when caused by external factors to the model or independent to the model.

- **Confounder** is a variable that influences both the dependent and independent variable.
- **Collider** a variable that is causally influenced by two or more variables. By varying this you would experience correlation between x_1 and x_2
- **Mediator** is when an independent variable impacts a dependent variable through the causal pathway of an effect

Structural Causality Models:
Set of observations $X = \{x_1, \ldots, x_N\}$, set of unobserved conditions $U = \{u_1, \ldots, u_N\}$, set of causal mechanisms $F = \{f_1, \ldots, f_N\}$ where the value of each variable s a function of its parents (direct causes) $x_k := f_k(pa_k, u_k), k = 1, \ldots, N$.
Ladder of Causation:

- **Association**: *(seeing observing)* How would seeing X change my belief in Y? NNs good at this

SCMs with jointly independent exogenous noises are Markovian: $P(u_1, \ldots, u_N) = \prod_{k=1}^{N} P(u_k)$ where Markovian SCMs induce unique join <u>observational distribution</u> over the endogenous variables $P_M(x_1, \ldots, x_N) = \prod_{k=1}^{N} P_M(x_k|pa_k)$ so Each variable is independent of its non-descendants given its direct causes.

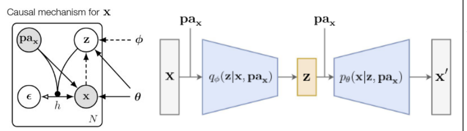
- **Intervention**: *(Doing, intervening)* What will Y be if I do X?

SCMs predict the causal effects of actions via interventions. Intervene with structural assignment $do(x_k := c)$ which induces a submodel M_c and its entailed distribution is known as the intervention distribution $P_{M\,c}(X|do(c))$

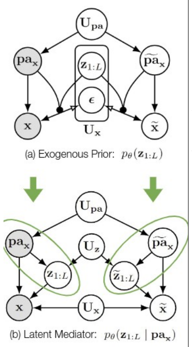
- **Counterfactual**: *(Imagining, Retrospection, Understanding)* What if X had not occurred?

Inference involves 3 steps:
1) <u>Abduction</u>: Update $P(U)$ given observed evidence, i.e. infer posterior $P(U|X)$
2) <u>Action</u>: Perform an intervention e.g. $do(\tilde{x}_k := c)$ and obtain M_c
3) <u>Prediction</u>: Use modified model $\langle M_c, P(U|X)\rangle$ to compute counterfactuals

This is difficult to do in a Deep Learning Setting.
Normalizing Flows: build complex probability distributions via successive (invertible) transformations of simple distributions (we can then invert to find exogenous noise term given the x)
<u>Change of variable:</u> $p(x) = p(u)|\frac{du}{dx}|$ (where we had a function f in terms of u, now we have a function in terms of u which also preserves the density) For training normalizing flows maximum log likelihood training objective $\log p(x) = \log p(f_\theta^{-1}(x)) + \log|\frac{df_\theta^{-1}(x)}{dx}|$. We can also condition on parents via learned $\log p(x|pa_x) = \log p(f_\theta^{-1}(pa_x, x)) + \log|det(\frac{\partial f_\theta^{-1}(pa_x,x)}{\partial x})|$ where in multivariate setting you need to compute the determinant of Jacobian matrix $.



VAE: Abduction is non-deterministic: $x := f_\theta(pa_x, u_x) = h(\epsilon; g_\theta(z, pa_x)) = \mu(z, pa_x) + \sigma(z, pa_x) \odot \epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I})$. Here, h is invertible but g_θ is not. X:image, pa: age, etc. g: encoder that predicts mean and std, and h: how we sample from the distribution.
<u>Factorized exogenous noise:</u>
$p(u_x|x, pa_x) \approx q_\phi(z|x, pa_x)\delta(\epsilon - h^{-1}(x; g_\theta(z, pa_x))$.
Example: Abduct $z \sim q_\phi(z|x, pa_x)$ *and* $\epsilon = h^{-1}(x; g_\theta(z, pa_x)) = \frac{x-\mu(z,pa_x)}{\sigma(z,pa_x)}$. *Action:* $do(pa_x := \tilde{pa}_x)$ *Predict:* $\tilde{x} := h(\epsilon; g_\theta(z, \tilde{pa}_x)) = \mu(z, \tilde{pa}_x) + \sigma(z, \tilde{pa}_x) \odot \epsilon$



<u>This doesn't scale to high resolutions</u>: have L latent dimensions. You can include an exogenous prior or conditional prior $p_\theta(z_{1:L}|pa_x)$ with residual blocks.

The conditional prior induces a latent mediator, as $z_{1:L}$ is no longer exogenous. Nonetheless, the underlying SCM has Markovian interpretation: $p(U) = p(U_x)(\prod_{k=1}^{K} p(U_{pa_k}))(\prod_{i=1}^{L} p(U_{z_i}))$
This allows you to perform <u>Casual Mediation Analysis</u>: The study of how a treatment effect is mediated by another variable, to help explain why or how an individual may respond to certain stimulus.

➔ Enables estimation of **Direct (DE)**, **Indirect (IE)** and **Total (TE)** causal effects:

$$\mathrm{DE}_\mathbf{x}(\widetilde{\mathbf{pa}}_\mathbf{x}) = \mathbb{E}\left[g_\theta(\widetilde{\mathbf{pa}}_\mathbf{x}, \mathbf{z}_{1:L}) - g_\theta(\mathbf{pa_x}, \mathbf{z}_{1:L})\right]$$
$$\mathrm{IE}_\mathbf{x}(\widetilde{\mathbf{z}}_{1:L}) = \mathbb{E}\left[g_\theta(\mathbf{pa_x}, \widetilde{\mathbf{z}}_{1:L}) - g_\theta(\mathbf{pa_x}, \mathbf{z}_{1:L})\right]$$
$$\mathrm{TE}_\mathbf{x}(\widetilde{\mathbf{pa}}_\mathbf{x}, \widetilde{\mathbf{z}}_{1:L}) = \mathbb{E}\left[g_\theta(\widetilde{\mathbf{pa}}_\mathbf{x}, \widetilde{\mathbf{z}}_{1:L}) - g_\theta(\mathbf{pa_x}, \mathbf{z}_{1:L})\right]$$

Soundness theorem: states that the properties of these are necessary:

- Composition: intervening on a variable to have the value it would otherwise have without the intervention will not affect other variables
- Intervening on a variable to have a specific value will cause the variable to take on that value
- Reversibility Axiom: precludes multiple solutions due to feedback loops. If setting a variable X to a value x results in a value y for a variable Y, and setting Y to a value y results in value x for X, then X and Y will naturally take on the values X and Y

there is a trade-off between composition and effectiveness.

Data quality: weighted loss depending on trust. <u>Optimal privacy preservation requires implementations that are private by design</u>. **Federated Learning**: train an ML model across decentralized clients with local data by sending the model to the cite, without exchanging data; only the result (encrypted local model output) is ever sent then aggregated across cites. $\mathcal{L}(\Theta) = \sum_{k=1}^{K} \frac{n_k}{N}\mathcal{L}_k(\Theta)$ with $\mathcal{L}_k(\Theta) = \frac{1}{n_k}\sum_{i \in P_k} \mathcal{L}(x_i, y_i, \Theta)$ gradient descent $\Theta^{j+1} := \Theta^j - \tau\nabla\mathcal{L}(\Theta)$ with SGD steps $\nabla\mathcal{L}(\Theta) \approx \sum_{i=1}^{m} \nabla\mathcal{L}(x_i, y_i, \Theta)$.
Careful: $\mathbb{E}_{P_k}[\mathcal{L}_k] = \mathcal{L}$ if iid, otherwise if data is homogeneous this is not the case.
Federated SGD: instead of sending back the individual gradients, allow clients to do their own SGD and send back model parameters where each client computes $\Theta_k^{j+1} := \Theta_k^j - \tau\nabla\mathcal{L}_k(\Theta)$ and the server computes $\Theta_k^{j+1} = \sum_{k=1}^{K} \frac{n_k}{N}\Theta_k^{j+1}$. **Differences**: it doesn't make sense to do GD on local set because of noisy gradient. If you send the gradient, you get more reliable GD but with constant communication. However, in federated, after 10 iterations client shares updated params, so there is better comms.
Challenges: <u>non-iid data</u>: local dataset is not representative of whole population, <u>unbalanced data</u>: varying, <u>massively distributed data</u>: a cite may only have one type for each class, <u>communication costs</u>: overhead depends on the # of clients and Hz of updates from/to server.

Algorithm 1 FederatedAveraging. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

```
Server executes:
  initialize $w_0$
  for each round $t = 1, 2, \ldots$ do
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of $m$ clients)
    for each client $k \in S_t$ in parallel do
      $w_{t+1}^k \leftarrow$ ClientUpdate$(k, w_t)$
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate$(k, w)$:   // Run on client $k$
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)
  for each local epoch $i$ from 1 to $E$ do
    for batch $b \in \mathcal{B}$ do
      $w \leftarrow w - \eta\nabla\ell(w; b)$
  return $w$ to server
```

Interpretability: trees are only interpretable until a point, fairness, accountability, trust and causality doesn't necessarily imply interpretability.
Ablation Test: train without certain features and study the impact, difficult and expensive **Fit function**: do sensitivity analysis on saliency maps. Take two samples and the gradient difference between the two indicates how drastically if you change a feature value how the classification changes. However, a value nearby may be totally different – diff interpretations.

Interpretation strategies: <u>visualization and attribution</u> (identify input features responsible for model decisions) <u>visualize filters</u> (easy to implement, limited practical value) <u>instead, visualize activations generated by kernels</u> (easy to interpret esp. for early values) <u>occlusion</u> If masked out region causes a significant drop in confidence, the masked-out region is important.

Saliency Maps (DeconvNet): inverts the process of the forward pass (take an image up until a layer, then try to reverse it, no training, just backprob, record information, e.g. for maxpool the locations, then insert values where maxs were on the forward pass). Saliency maps are computed w.r.t derivate of input pixels (not weights). In classification, take the prediction and do backprob. **Guided Backpropagation**: positive grad = feature neuron is interested in, negative otherwise. Therefore, set all negative gradients to 0.
<u>ReLU activation:</u> $f_i^{l+1} = ReLU(f_i^l) = \max(f_i^l, 0)$
<u>backprop:</u> $R_i^l = (f_i^l > 0) \cdot R_i^{l+1}$, where $R_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}}$
<u>deconvnet:</u> $R_i^l = (R_i^{l+1} > 0) \cdot R_i^{l+1}$
<u>guided:</u> $R_i^l = (f_i^l > 0) \cdot (R_i^{l+1} > 0) \cdot R_i^{l+1}$
Cam and Grad-Cam: allows you to query what specific channels do in terms of activation **Deep Dream**: find an image x such that the activation $\phi_n(x)$ at layer n is high: $\max_x \phi_n(x) - \lambda\mathcal{R}(x)$ where R is a regulizer. 1) forward propagate to layer n, non minimisation of loss, instead maximise L2 norm of activations and backpropagate to input layer. *Resulting image will show learned features*.

Adversarial Attacks: assuming a linear layer $\theta^\top x$ we can think of an adversarial example that contains a non perceivable pertubation $\eta : \tilde{x} = x + \eta$. Then the logits become $\theta^\top x + \theta^\top\eta$ we thus maximise the value of $\theta^\top\eta$ such that $||\eta||_\infty \leq \epsilon$ (the max element of eta is less than epsilon) and assume a pertubation
$\eta = \epsilon \cdot \mathrm{sign}(\theta) \iff \theta^\top\eta = \epsilon \cdot \theta^\top\mathrm{sign}(\theta) = \epsilon||\theta||_1 = \epsilon mn$
Here the average magnitude of an element of θ is given by m. The change in activation given by the perturbation increases linearly with respect to n where for images, n is # of pixels.
Fast Gradient Sign Model: Perform gradient descent in order to maximize the loss. Consider the input image x to be a trainable parameter and compute the gradient with respect to the input image to create a perturbation. $\eta = \epsilon \cdot \mathrm{sign}(\nabla_x\mathcal{L}(\theta, x, y))$ where the example can be created as $\tilde{x} = x + \epsilon \cdot \mathrm{sign}(\nabla_x\mathcal{L}(\theta, x, y))$

Homomorphic Encryption: for some operation $\odot$, $[x \odot y] \equiv [x] \odot [y]$; finds an encryption scheme, such that the encrypted data can be manipulated in the encrypted domain as though it had been decrypted. <u>This is computationally expensive 100x slower in DL and hard to find encryption for certain $\odot$; restricted to certain operations. BUT can perform inference on encrypted data and doesn't require interaction between data and model owners.</u> **Secure Multi-Party Computation**: create a splitting scheme and let clients computation and return answer. <u>Shared governance: can only retrieve data if all agree, confidentiality; noone knows the true value.</u>

K-anonymity: prevents record linkage attacks which reverse engineer by ensuring that for any persons data, there are at least $k-1$ other records which are indistinguishable. Only really feasible for smaller big datasets with small number of fields for each record.

Model Inversion: Attacker uses internal representations of the joint model to reconstruct individual training samples or their sensitive attributes. **Membership inference**: Attacker obtains a data record and determines if it was used to train a particular model. **Attribute Inference**: Attacker uses model access and auxiliary information about the victim to obtain the sensitive values of their data.

Algorithm 1 Differentially private SGD (Outline)

```
Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate $\eta_t$, noise scale $\sigma$, group size $L$, gradient norm bound $C$.
  Initialize $\theta_0$ randomly
  for $t \in [T]$ do
    Take a random sample $L_t$ with sampling probability $L/N$
    Compute gradient
    For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$
    Clip gradient
    $\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$
    Add noise
    $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\sum_i(\bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2C^2\mathbf{I}))$
    Descent
    $\theta_{t+1} \leftarrow \theta_t - \eta_t\tilde{\mathbf{g}}_t$
  Output $\theta_T$ and compute the overall privacy cost $(\varepsilon, \delta)$ using a privacy accounting method.
```

Per sample – clip gradients so their norm is controlled

Differential Privacy: perturb the data so that information about the single individual is reduced while retaining the capability of learning. "preventable attacks". <u>Randomized responses</u>: draw statistical conclusions from the dataset without revealing information about individual data points by adding a controlled amount of noise. If a person is surveyed multiple times less protection. (epsilon differential privacy) **Addition algorithm:** <u>An algorithm is made approximately invariant to inclusion of simple data example</u> iff excluding one person doesn't change result of model.