Neural Networks

Part II

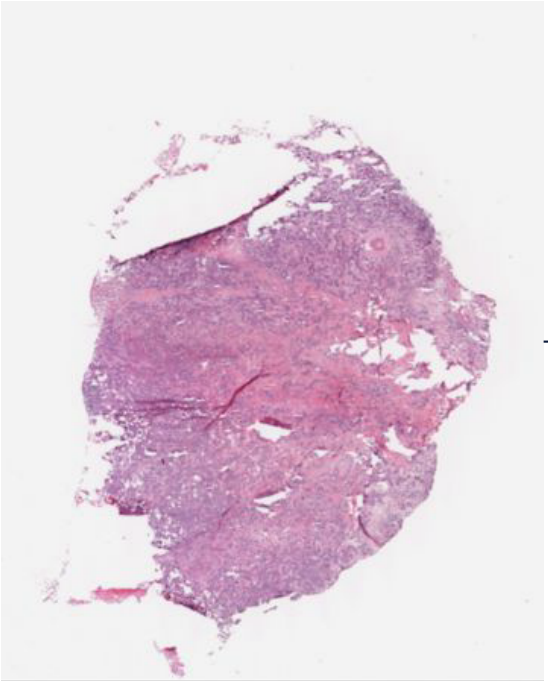
CPH 101A

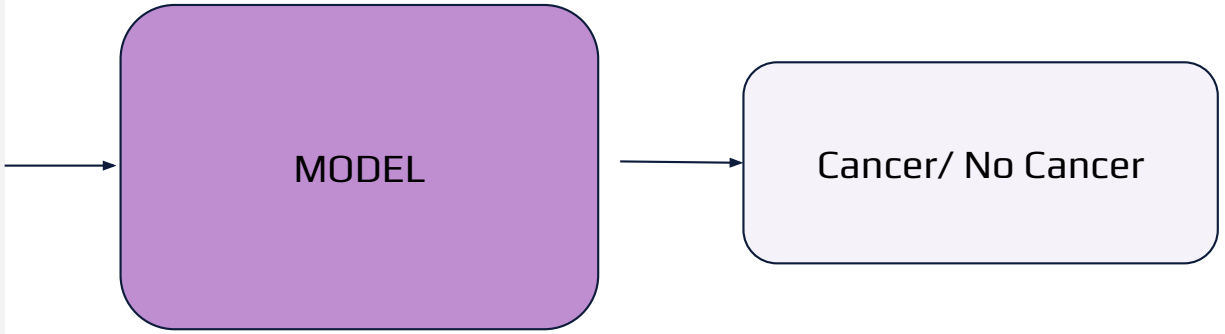
Agenda

1. Recap
2. How training works
3. Failure modes
4. Partial Remedies

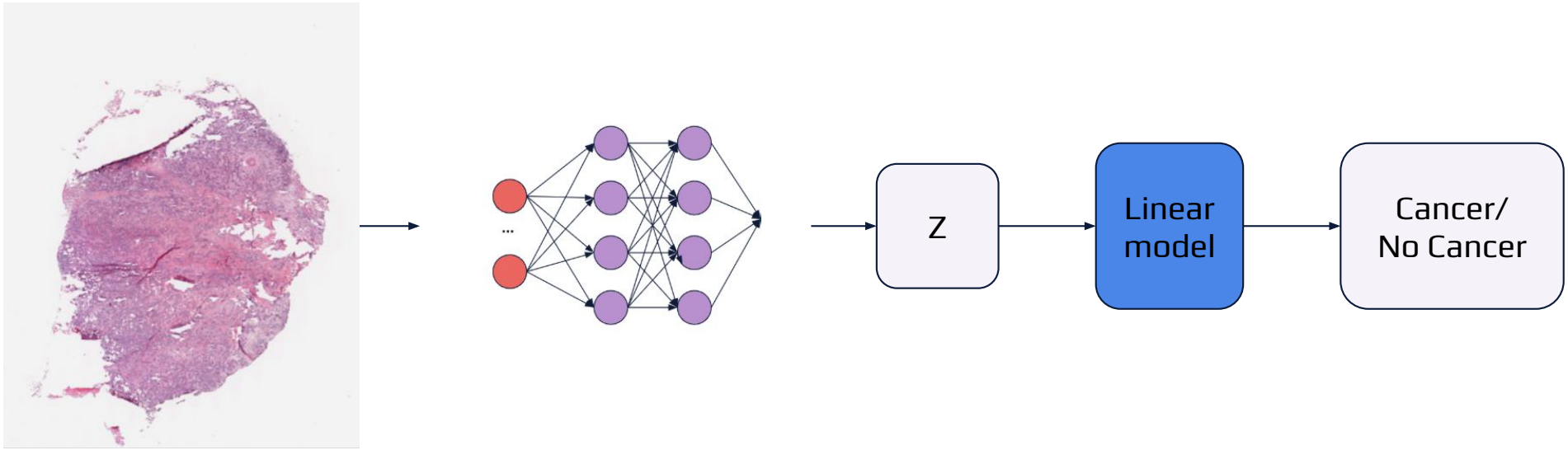
   Initialization, Optimizers, Hypothesis class, Regularizers

Goal: identify cancer slides

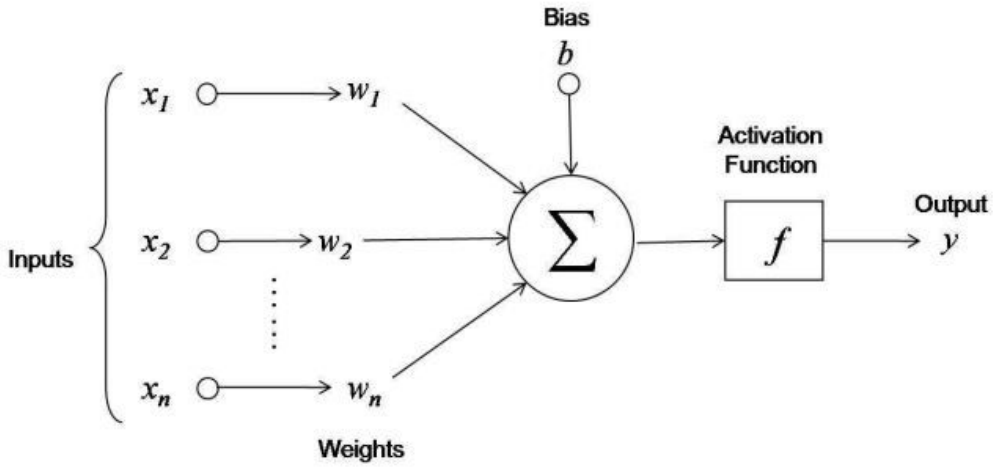




Intermediate features might work:
Can we learn them from scratch?



Neuron



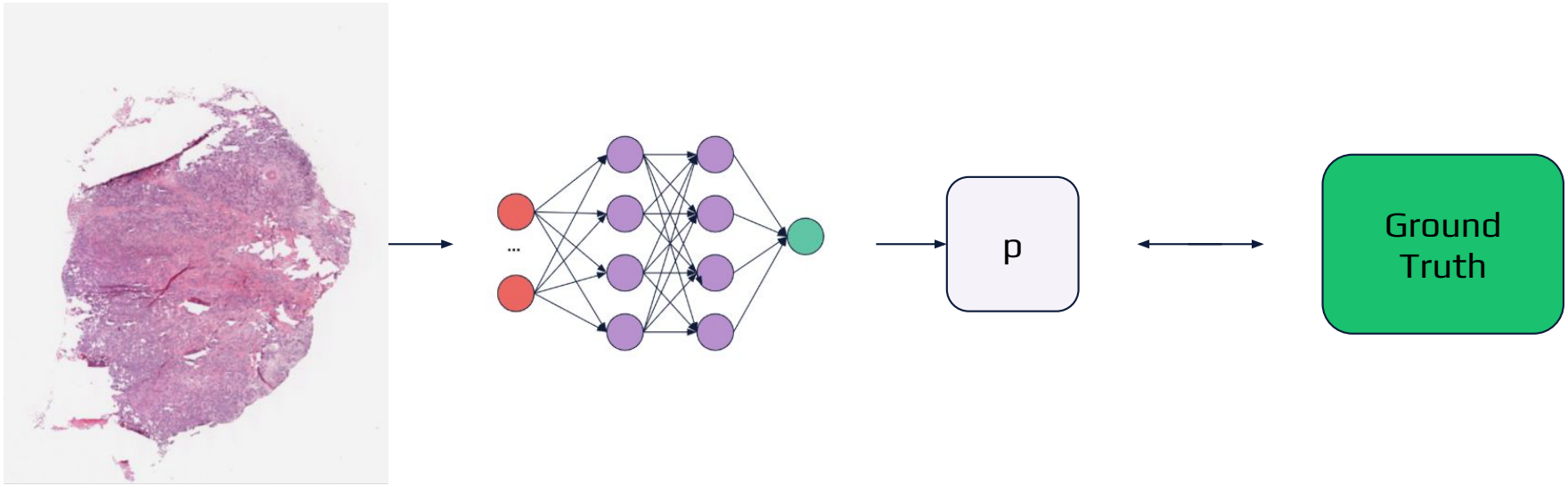
$\theta \in \mathbb{R}^n$ weights

$b \in \mathbb{R}$ bias

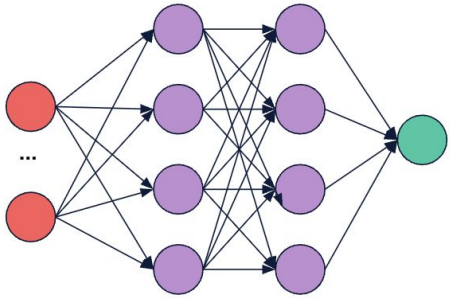
$a = \theta x + b$

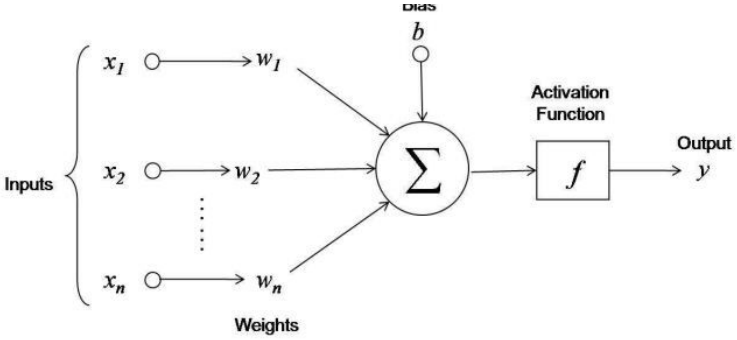
$z = f(a)$

How do we train a Neural Network?



How do we train a Neural Network?





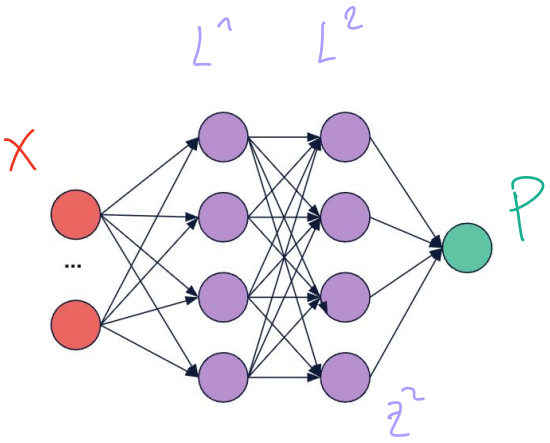
1) Loss

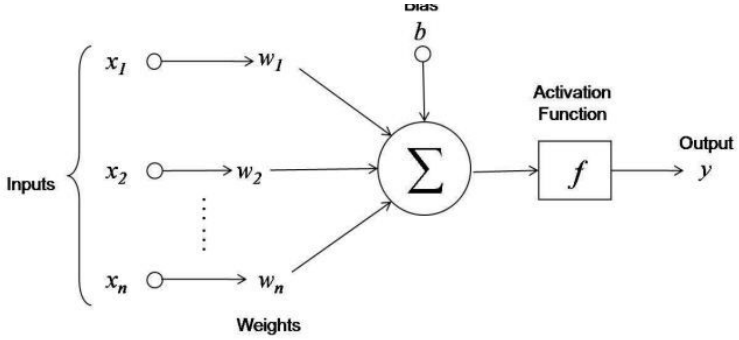
$$L = \frac{1}{2}(p - y)^2$$

p: prediction

y: target

How do we train a Neural Network?





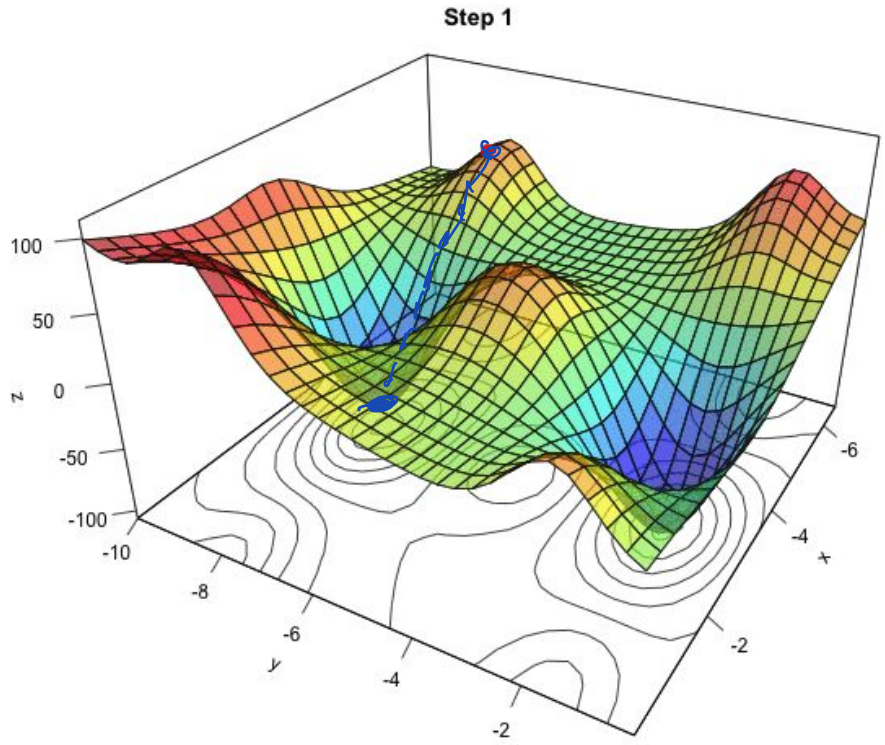
2) Gradient Descent

$$p = \theta^p z^2 + b^p$$

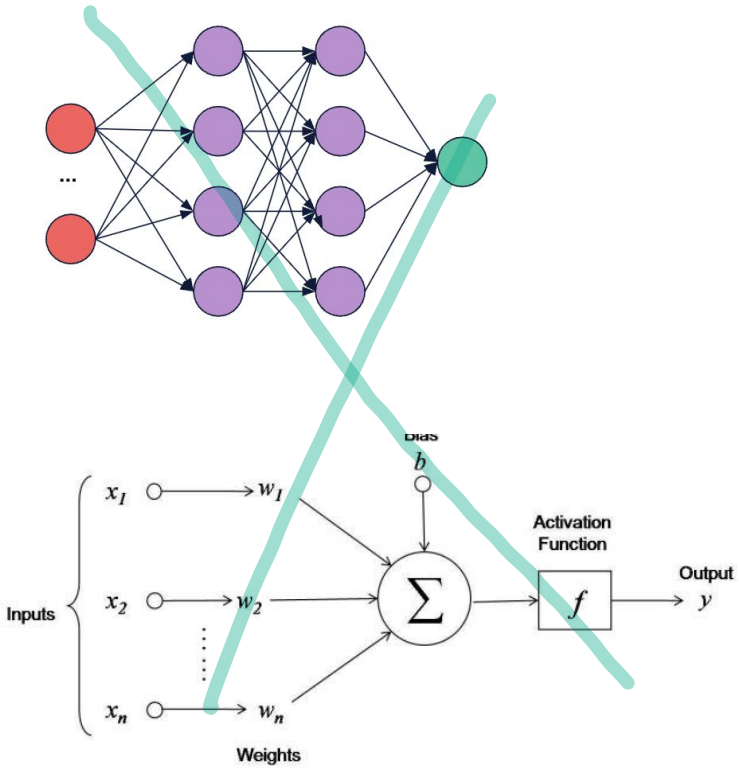
$$\theta^p_t = \theta^p_{t-1} - \eta \frac{\partial L}{\partial \theta^p}$$

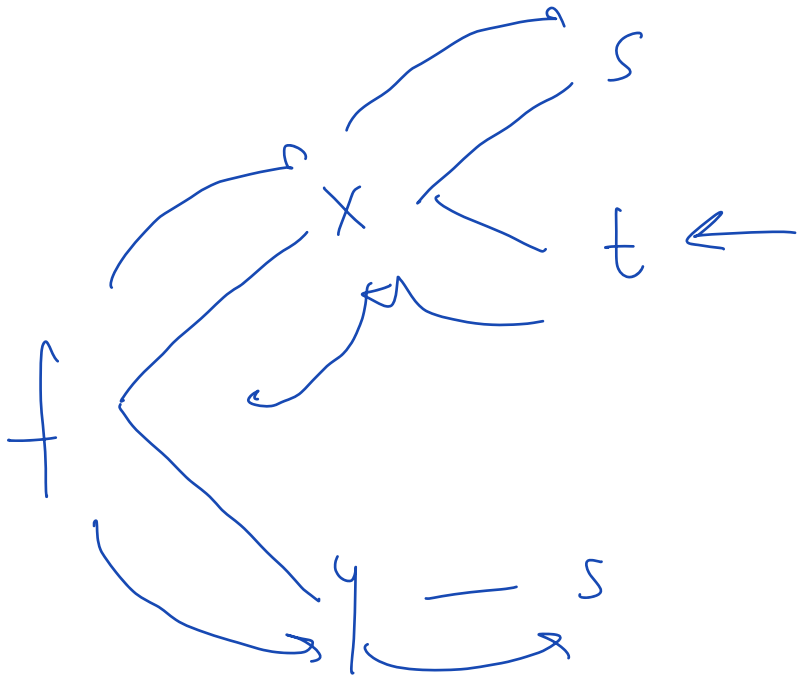
learning rate

Gradient Descent



Reminder: Chain Rule





$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s}$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial t}$$

Gradient Descent

$$p = \theta^p z^2 + b^p \leftarrow$$

$$L = \frac{1}{2}(p-y)^2$$

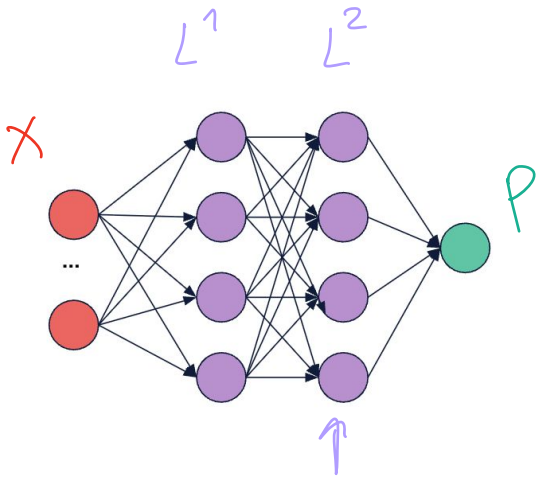
$$\theta_t^p = \theta_{t-1}^p - \eta \frac{\partial L}{\partial \theta^p}$$

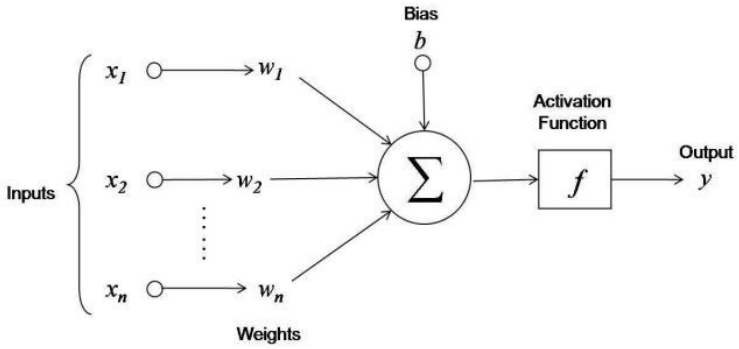
$$\Rightarrow \theta_t^p = \theta_{t-1}^p - \eta\left((p-y)z^2\right)$$

$$\frac{\partial L}{\partial \theta^p} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial \theta^p} = z^2$$

$$= 2 \cdot \frac{1}{2}(p-y)$$

How do we train a Neural Network?

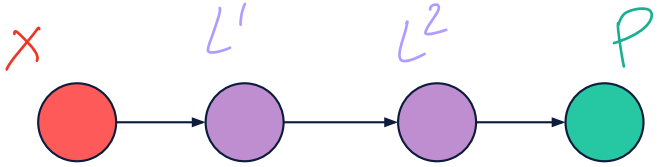




3) Backpropagation

$$\theta^2_t = \theta^2_{t-1} - \eta \frac{\partial L}{\partial \theta^2}$$

Backpropagation



$$\theta_t^2 = \theta_{t-1}^2 - \eta \frac{\partial L}{\partial \theta^2}$$

$$L = \frac{1}{2}(p-y)^2$$

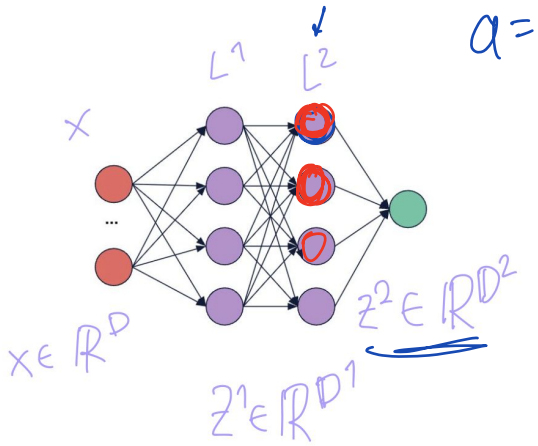
$$p = \theta^p z^2 + b^p$$

$$z^2 = f(a^2) \qquad a^2 = \theta^2 z^1 + b^2$$

$$\frac{\partial L}{\partial \theta^2} = \frac{\partial L}{\partial p} \cdot \frac{\partial p}{\partial z^2} \cdot \frac{\partial z^2}{\partial a^2} \cdot \frac{\partial a^2}{\partial \theta^2}$$

$$\frac{\partial L}{\partial p} = (p-y) \qquad \frac{\partial p}{\partial z^2} = \theta^p \qquad \frac{\partial z^2}{\partial a^2} = f'(a^2) \qquad \frac{\partial a^2}{\partial \theta^2} = z^1$$

Backpropagation



$$\theta_t^2 = \theta_{t-1}^2 - \eta \frac{\partial L}{\partial \theta^2}$$

$$L = \frac{1}{2}(p-y)^2$$

$$z^2 = f(a^2)$$

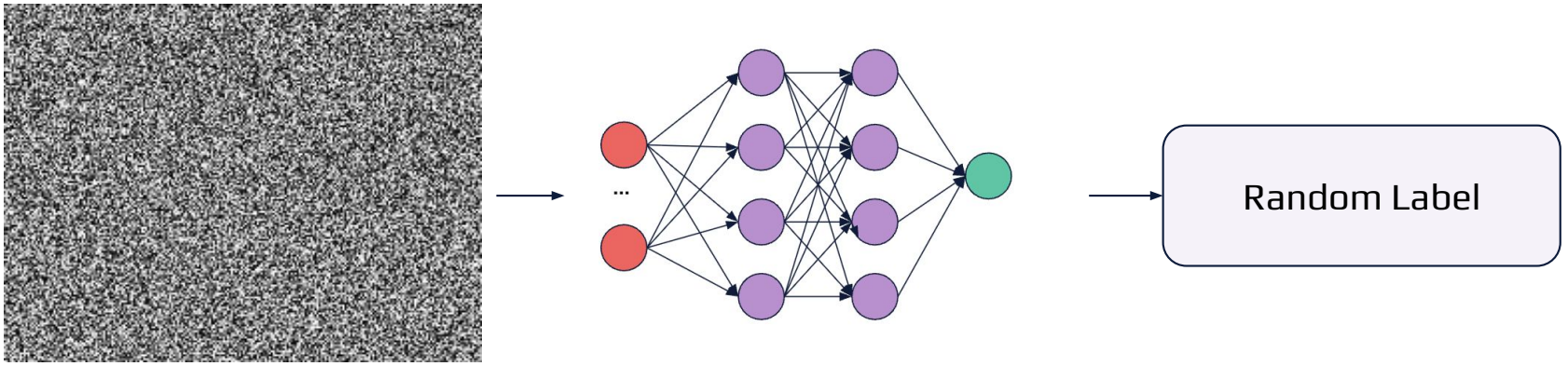
$$\frac{\partial L}{\partial \theta^2} = \frac{\partial L}{\partial p} \frac{\partial p}{\partial z^2} \frac{\partial z^2}{\partial a^2} \frac{\partial a^2}{\partial \theta^2}$$

$1 \times 1 \quad 1 \times D^2 \quad D^2 \times D^2 \quad D^2 \times (D^2 \times D^1)$

θ^2

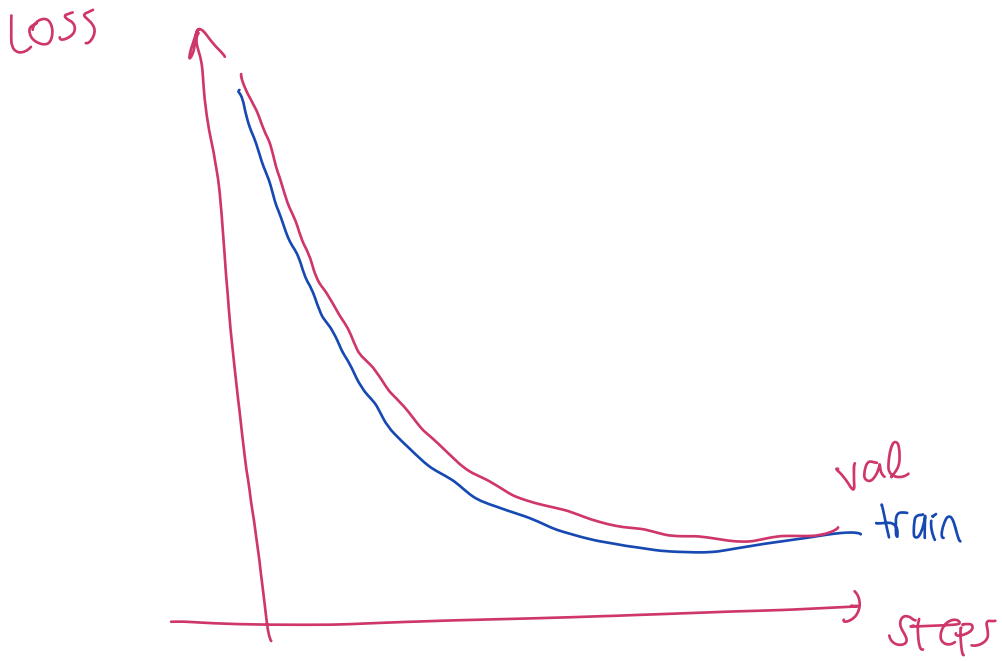
https://eli.thegreenplace.net/2018/backpropagation-through-a-fully-connected-layer/

Well trained Neural Networks can learn anything

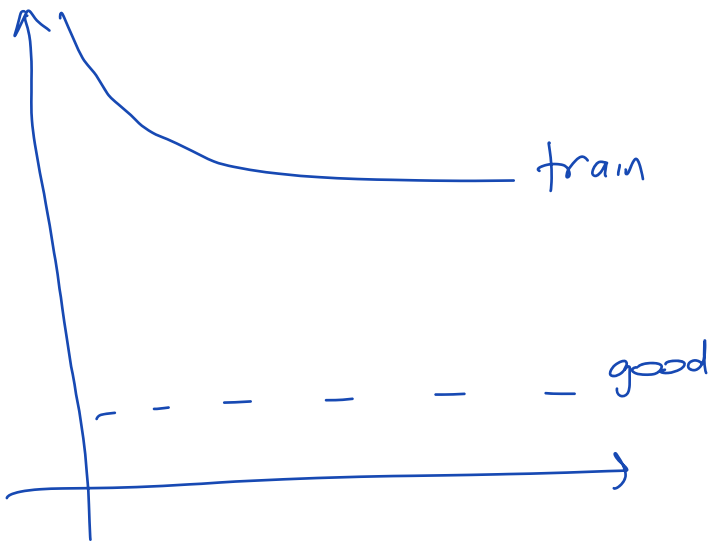


https://playground.tensorflow.org/
https://phiresky.github.io/neural-network-demo/

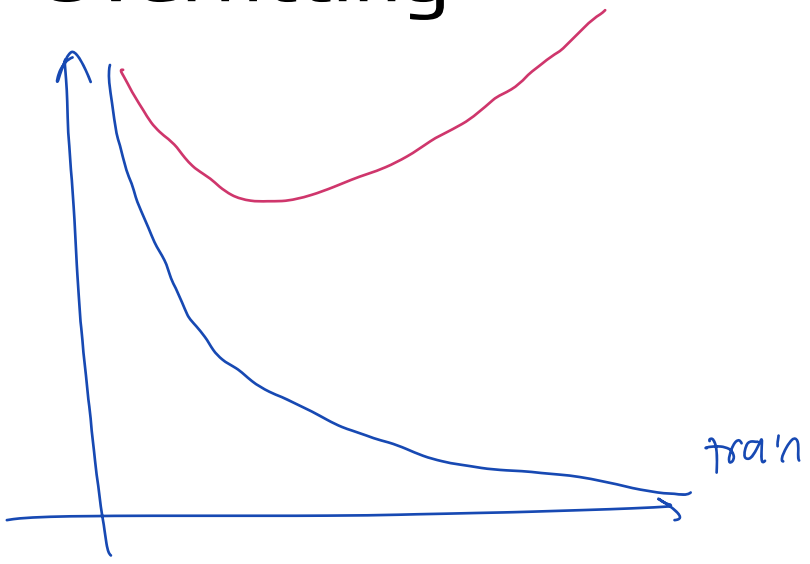
Healthy behavior

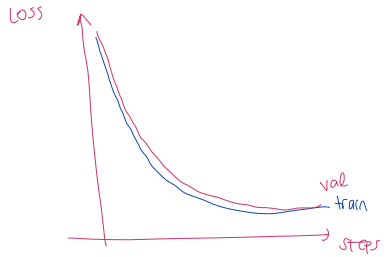


Underfitting



Overfitting



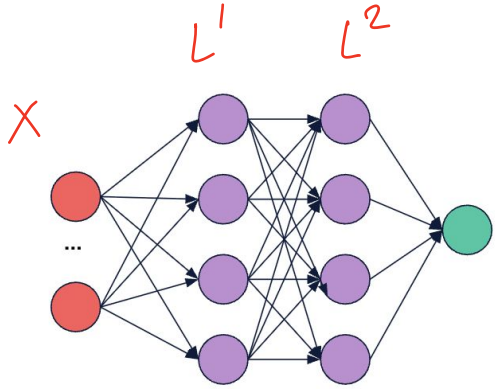


Underfitting

Main causes:

- Bad initialization
- Bad local minimum
- Bad optimization

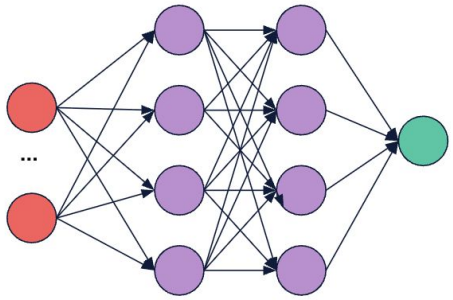
Will this network learn?



$\theta_j^{(i)} = 1 \quad \forall i, j$

Random init

Choosing scales of random initialization



$$\theta_t = \theta_{t-1} - \eta \frac{\partial L}{\partial \theta}$$

$$z^l = f(a^l) = f(\theta^l z^{l-1} + b^l) \quad \leftarrow \quad \frac{\partial z^l}{\partial z^{l-1}} = f'(\theta^l z^{l-1} + b^l)\theta^l$$

$$\frac{\partial L}{\partial \theta^1} = \frac{\partial L}{\partial z^L} \cdot \frac{\partial z^L}{\partial z^{L-1}} \cdot \frac{\partial z^{L-1}}{\partial z^{L-2}} \cdots \frac{\partial z^*}{\partial \theta^1}$$

$$\theta \in [10, 1000] \qquad (..)\theta^L(...)\theta^{L-1}\ldots\theta^{L-2}$$

$$\theta \in [0.000001, 0.00001] \qquad (..\pi..)\theta^L\theta^{L-1}\theta^{L-2}\ldots\theta^1$$

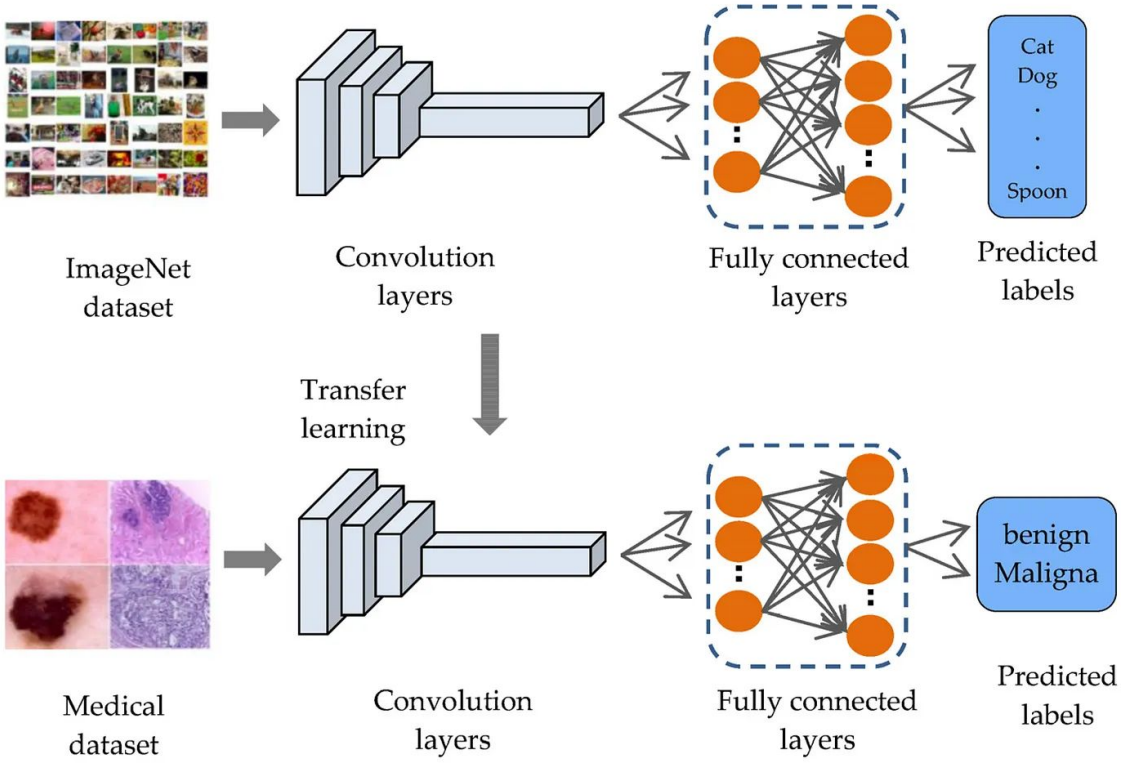
Choosing initialization

1. Break symmetry
2. Preserve scale

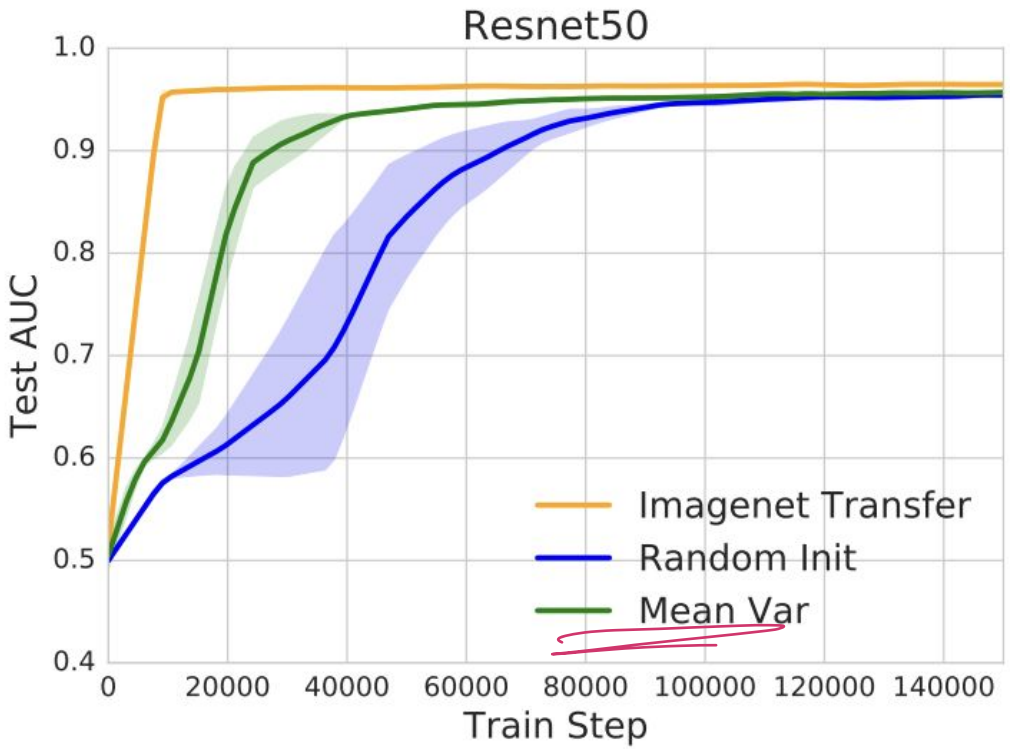
$\left\| \frac{\partial L}{\partial \theta} \right\|$ $\|z^l\|$

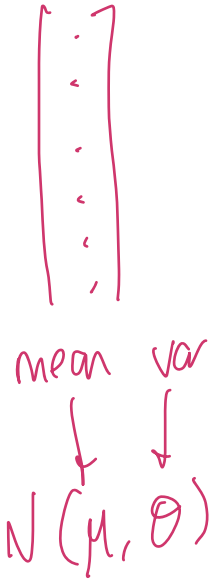
ReLu $\theta \sim N(0, \sqrt{2/D})$

Pretraining



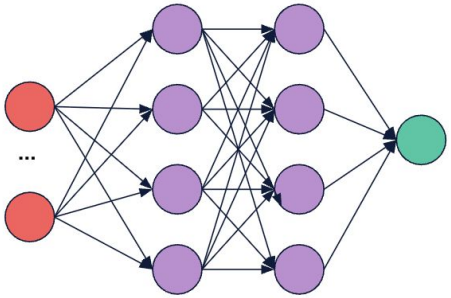
Surprising properties of transfer learning





Transfusion: Understanding Transfer Learning for Medical Imaging (2019)

Vanishing/Exploding gradients during training



BatchNorm to prevent activation collapse

$$\overline{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Residual Connections

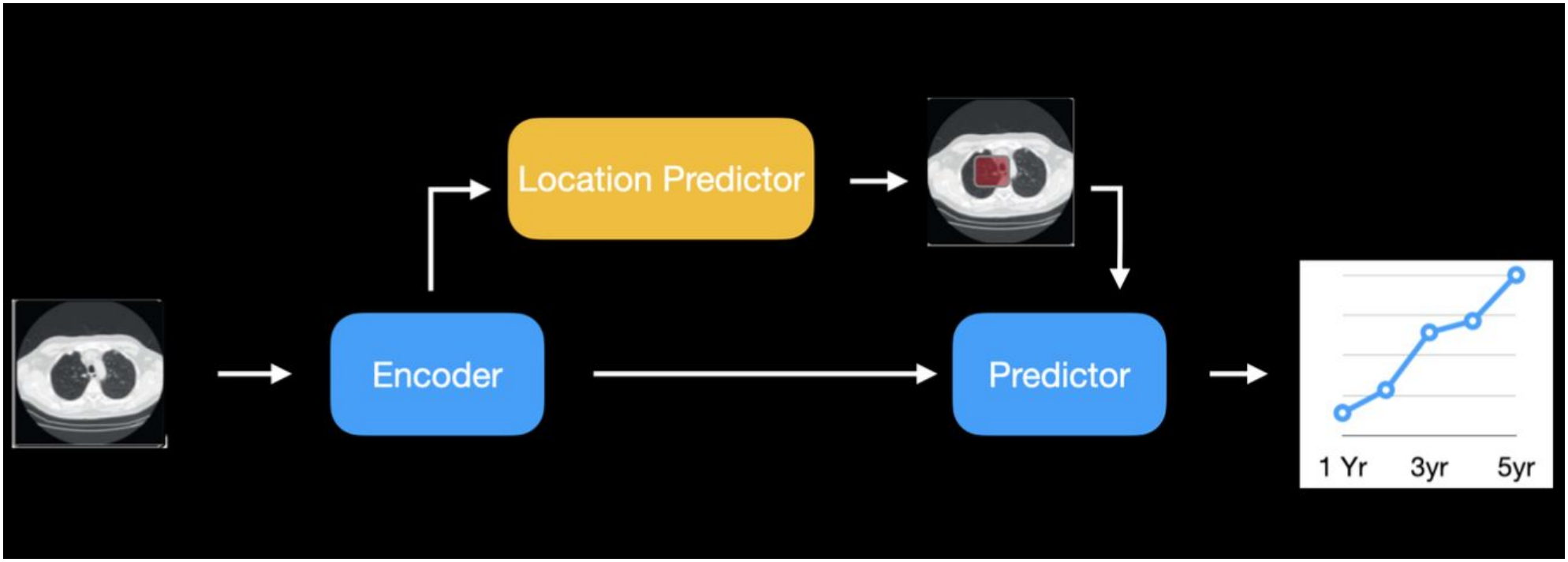
Optimizers

Overfitting

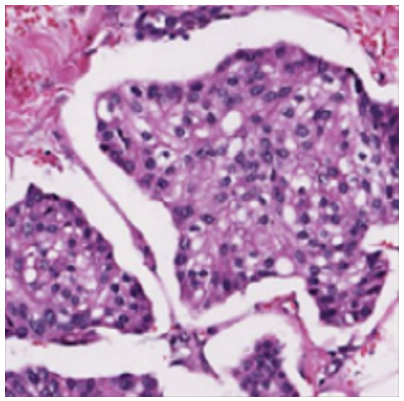
Main causes:

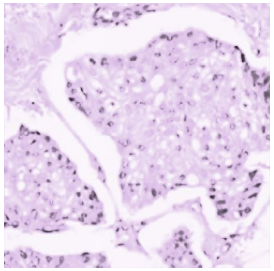
- Bias in the training data
- Not enough data
- Bad local optimum

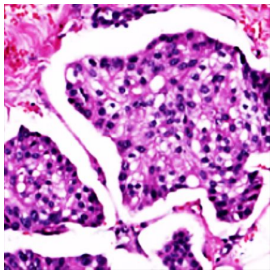
Additional Losses

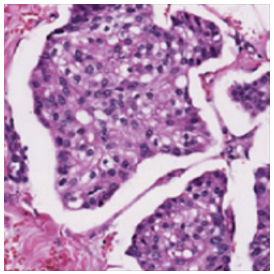


Data Augmentation









Model learning “well” depends on complex interaction between:

- Initialization
- Hypothesis Class
- Optimization
- Data
- Learning Rate