Particles and Cosmology

16th Baksan School on Astroparticle Physics

Machine Learning in Astroparticle Physics

Oleg Kalashev

Institute for Nuclear Research, RAS

Lecture 4

April 10-18, 2019

What we have learned so far
fast review

Supervised Machine Learning Tasks

$x \in \mathbb{X}$ **- sample** $x = (x^1, \ldots, x^d)$ **- sample features known**

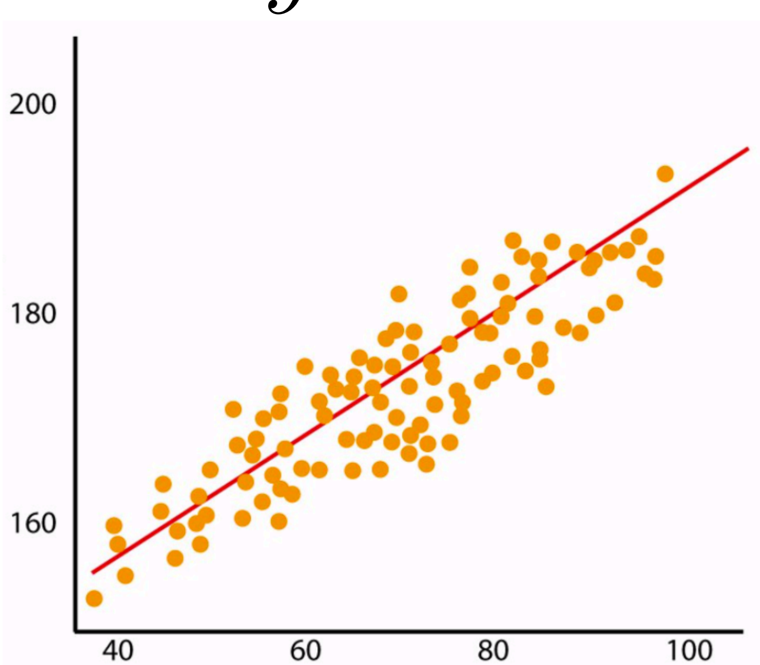
$y \in \mathbb{Y}$ **- answer (sample property which we want to predict)**

$X = (x_i, y_i)_{i=1}^{l}$ **training set is used to find a model** $y = a(x)\,, a \in \mathbb{A}$

by minimising loss (cost) function $C(a, X)$ **:** $a(x) = argmin_{a \in \mathbb{A}} C(a, X)$

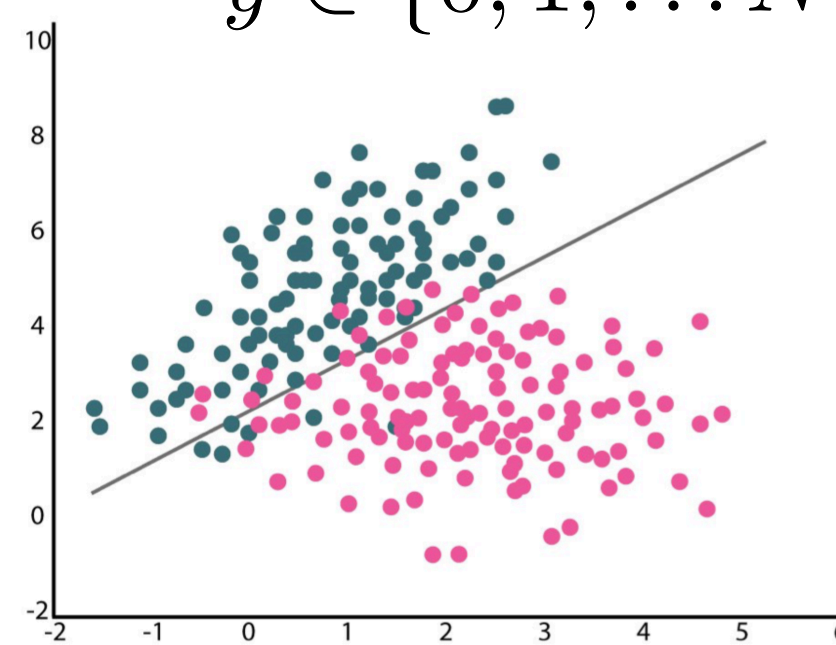
Regression: predict real value

$y \in \mathbb{R}$



Classification: predict class

$y \in \{0, 1, \ldots N\}$



Linear Regression

$$a(x) = w_0 + \sum_{j=1}^{d} w_j x^j$$ **or** $$a(x) = \sum_{j=0}^{d} w_i x^j = \langle w, x \rangle$$ **where** $x_0 \equiv 1$

$$C(a, X) = \frac{1}{l} \sum_{i=1}^{l} (a(x_i) - y_i)^2$$

$$C(a, X) = \frac{1}{l} \|Xw - y\|^2 \to \min_w$$

$$X = \begin{pmatrix} x_{11} & \dots & x_{1d} \\ \dots & \dots & \dots \\ x_{\ell 1} & \dots & x_{\ell d} \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ \dots \\ y_\ell \end{pmatrix}$$

gradient decent optimisation:

- **start with some random or zero** w
- **at step *t-1* calculate loss function gradient** $\nabla_w C(w, X) = \frac{2}{l} X^T (Xw - y)$
- **update weights** $w^t = w^{t-1} - \eta \nabla_w C(w^{t-1}, X)$ **and repeat**

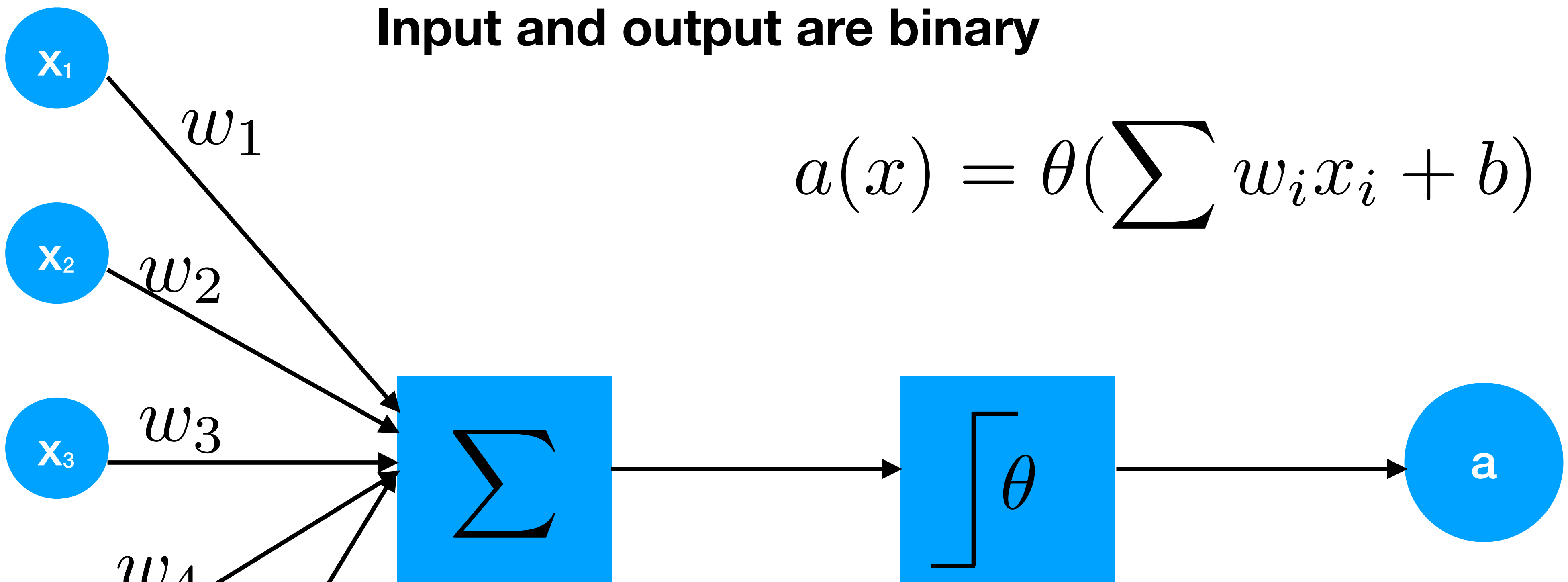
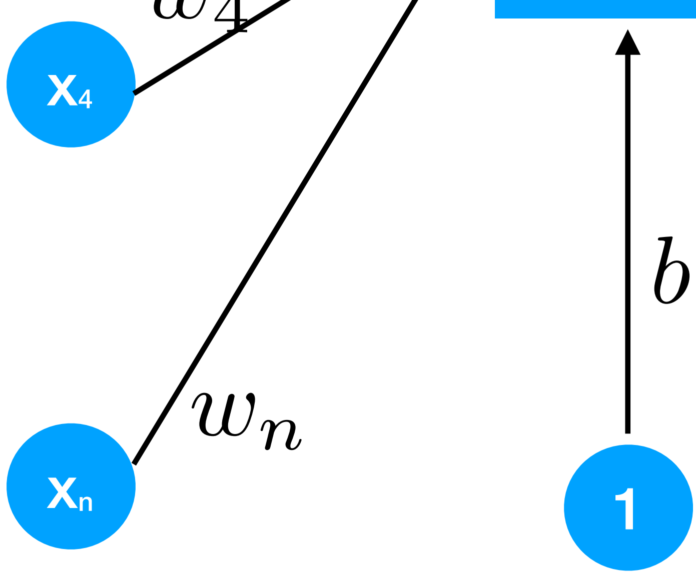
Biological Neural Networks

- Perceptron model.

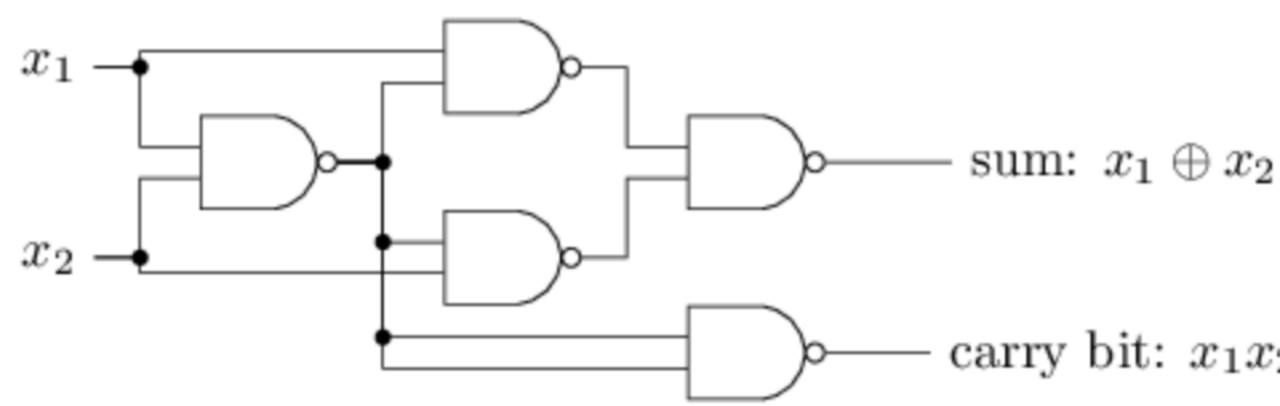
Walter Pitts, Warren McCulloch (1943)

Input and output are binary

$$a(x) = \theta(\sum w_i x_i + b)$$

Network of perceptrons can implement binary logic



Artificial Neural Networks

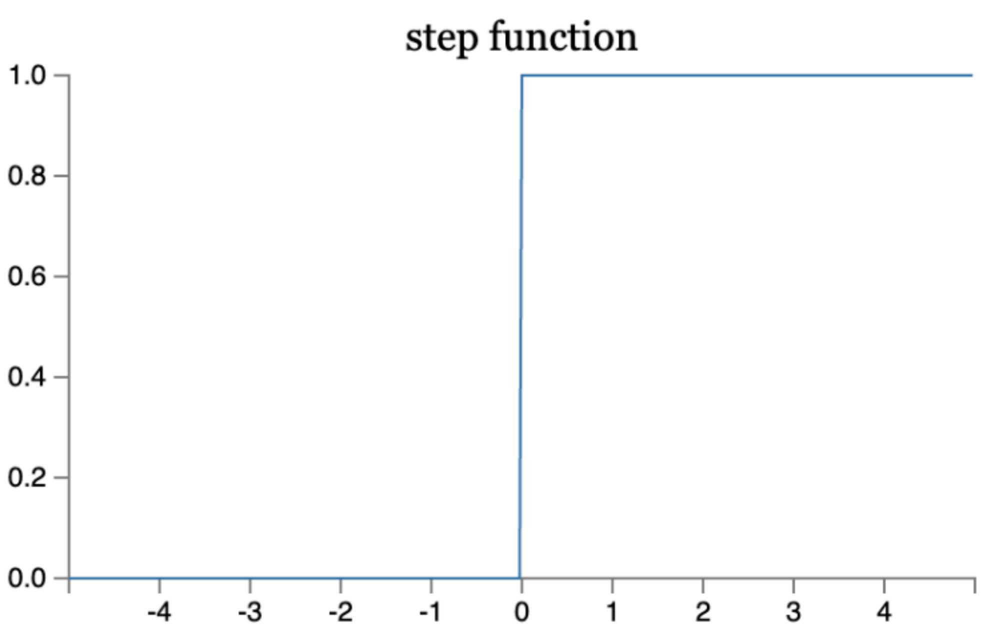
- Sigmoid neuron

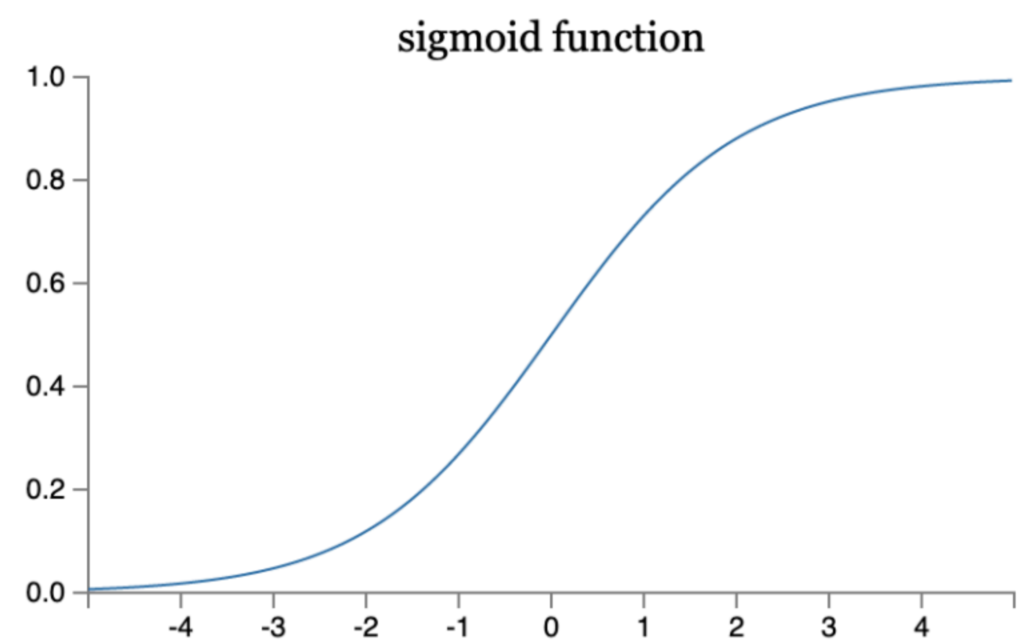
$$a(x) = \theta(\sum w_i x_i + b)$$

We want to train the weights.

- **continuous input/output instead of binary**
- **replace step by a smooth function**

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}$$





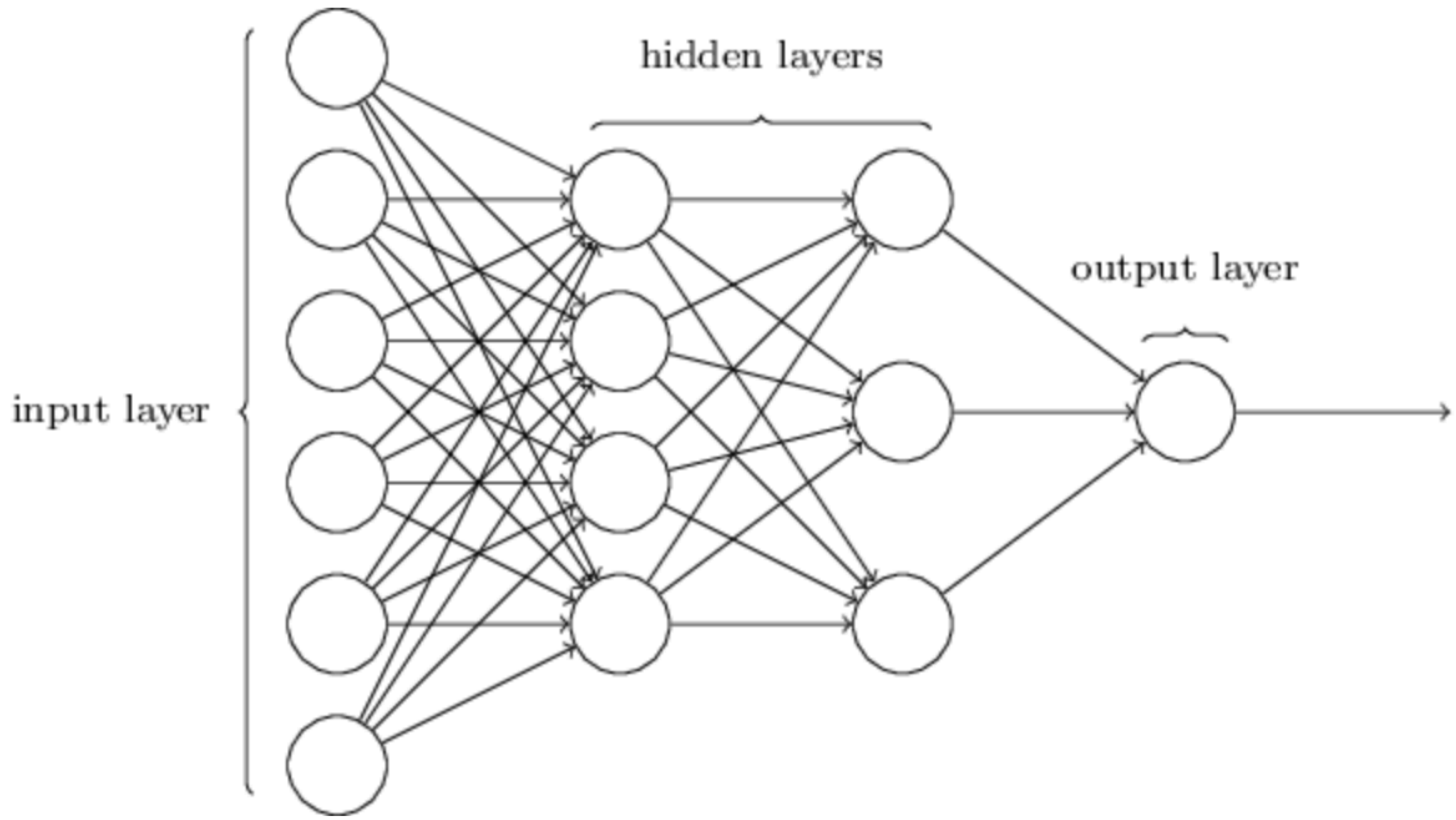
- **define lost (cost) function**

$$C(w,b) = \frac{1}{2n}\sum_x (y(x) - a)^2$$

note: if we used $\sigma(z) = z$
we would get linear regression

Artificial Neural Networks

- Multilayer Perceptron



Signal propagation: $$a_j^l = \sigma(z_j^l), \;\; z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l$$

Vector form: $$a^l = \sigma(w^l a^{l-1} + b^l)$$

Theorem (K. Hornik, 1991): ANY continuous function can be approximated with ANY precision by MLP

Back-propagation algorithm

Forward pass:

$$a_j^l = \sigma(z_j^l), \;\; z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l$$

Cost:

$$C(w,b) = \frac{1}{2n}\sum_x (y(x) - a^L)^2$$

1. calculate the error in the last layer δ_j^L using the chain rule

$$\delta_j^L = \sum_k \frac{\partial C}{\partial a_k^L}\frac{\partial a_k^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L),$$

2. calculate the error in the intermediate layer δ_j^l using the chain rule

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l}\delta_k^{l+1}$$

$$z_k^{l+1} = \sum_j w_{kj}^{l+1}\sigma(z_j^l) + b_k^{l+1}$$

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1}\sigma'(z_j^l)$$

$$\delta_j^l = \sum_k w_{kj}^{l+1}\delta_k^{l+1}\sigma'(z_j^l)$$ - recurrent expression

3. express $\delta C/\delta b_j^l$ and $\delta C/\delta w_{jk}^l$ via δ_j^l:

$$\frac{\partial C}{\partial w_{jk}^l} = \sum_i \frac{\partial C}{\partial z_i^l}\frac{\partial z_i^l}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}a_k^{l-1} = \delta_j^l a_k^{l-1}$$

$$\frac{\partial C}{\partial b_j^l} = \sum_k \frac{\partial C}{\partial z_k^l}\frac{\partial z_k^l}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} = \delta_j^l$$

ANN optimization

- speeding up learning by adjusting loss function
- weight initialization
- avoid overfitting
  - L1, L2 regularization
  - Dropout
- gradient decent optimization (momentum and adaptive learning rate)

Reading:

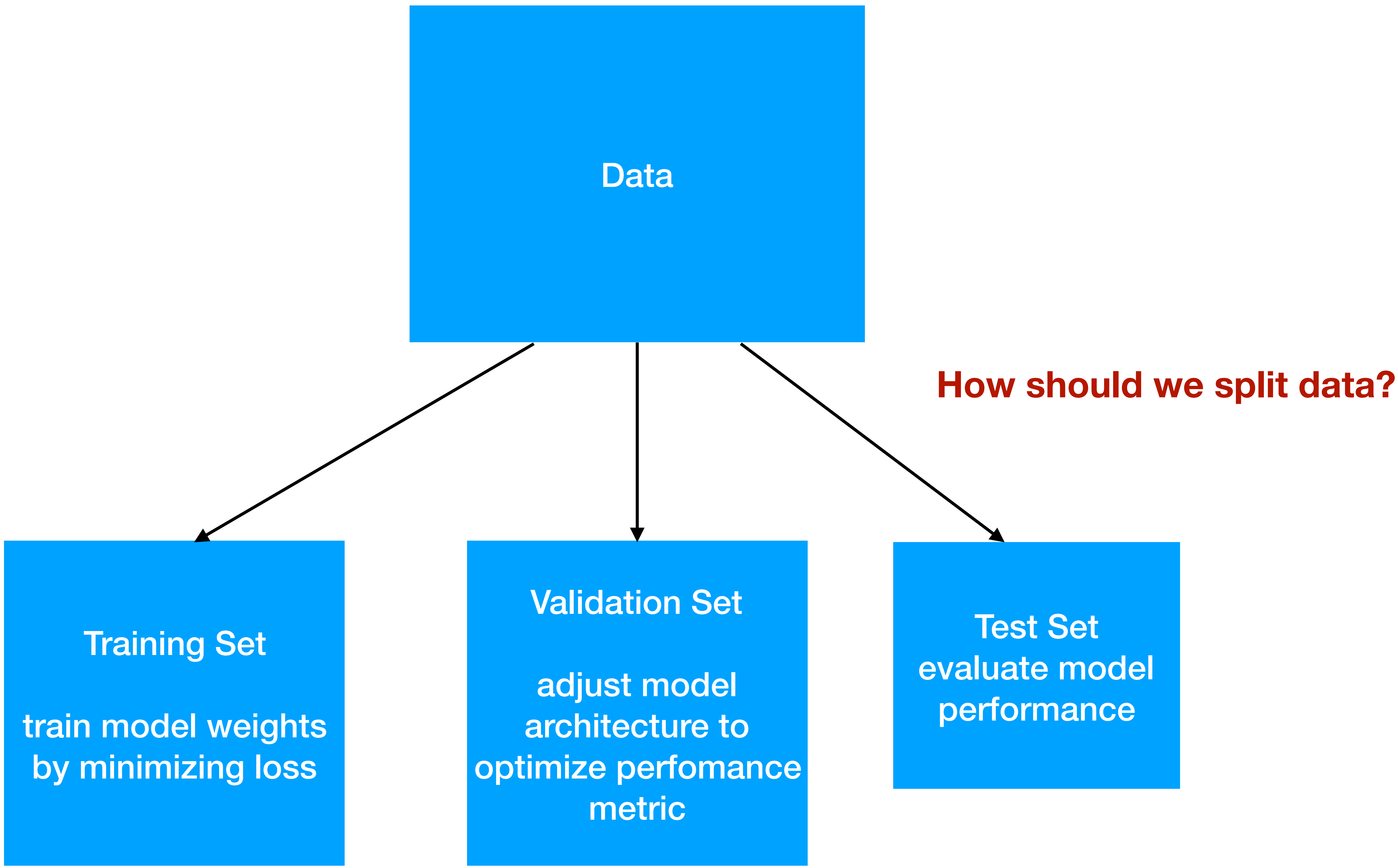
Michael Nielsen - Neural Networks and Deep Learning
http://neuralnetworksanddeeplearning.com/index.html

Andrew Ng
Deep Learning Specialization on Coursera
https://www.coursera.org/specializations/deep-learning

In Russian
Yandex Машинное обучение и анализ данных Specialization on Coursera
https://www.coursera.org/specializations/machine-learning-data-analysis

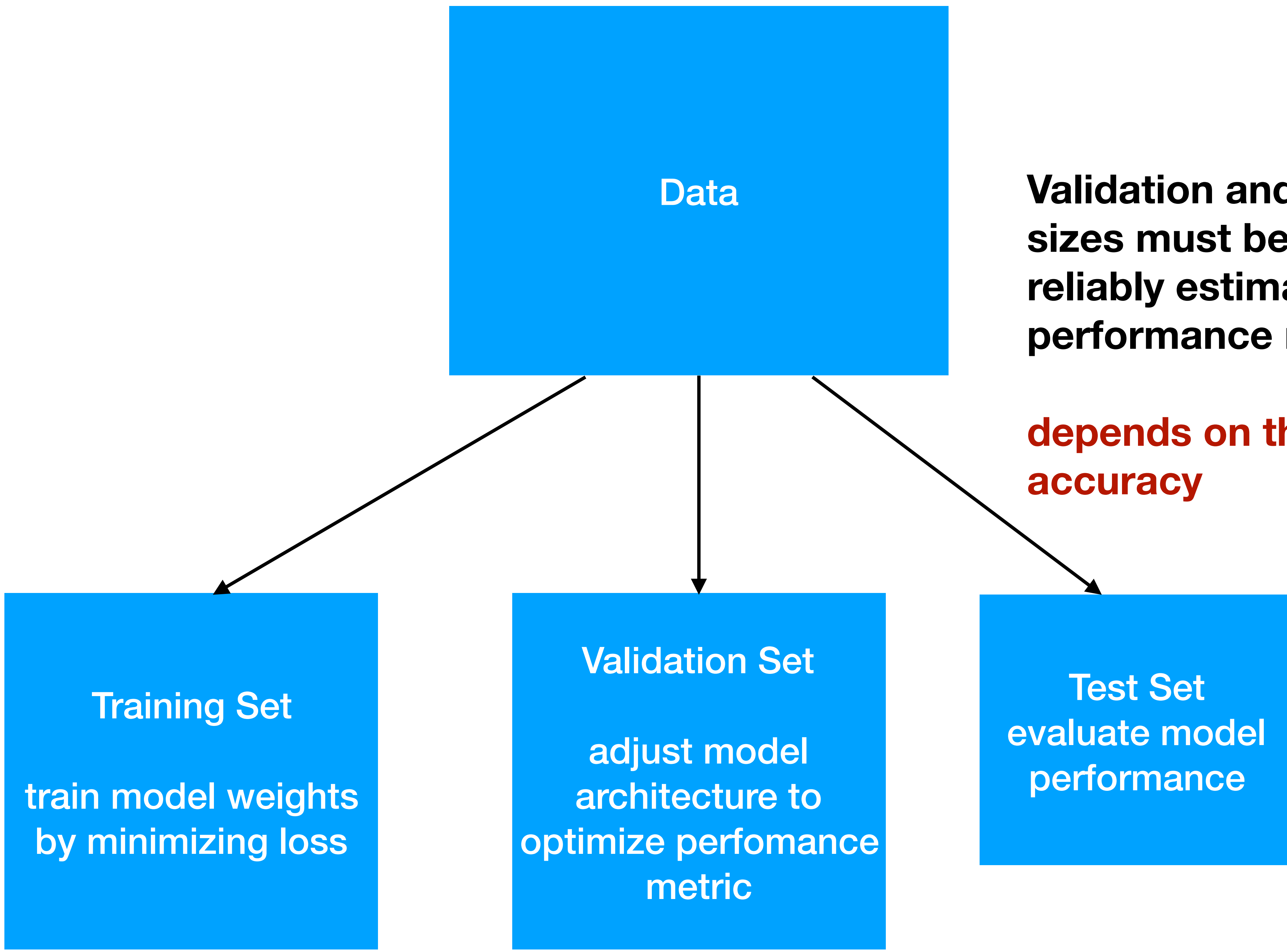
Optimizing ML model

splitting the data



Optimizing ML model

splitting the data



Validation and training set sizes must be just enough to reliably estimate model performance metric

depends on the required accuracy

Optimizing ML model

Strategy

Define metric for evaluation (not necessary the same as loss function)

e.g.

- **precision for classification task**
- **mean absolute error for regression task**
- **metrics may also depend e.g. on calculation time**

Optimizing ML model

Strategy

Define metric for evaluation (not necessary the same as loss function)

e.g.

precision for classification task
mean absolute error for regression task
explained variance score

$$EV(y, \hat{y}) = 1 - \frac{Var(y - \hat{y})}{Var(y)}$$

y - true value of quantity being predicted

$\hat{y}$ - model estimate of y

Optimizing ML model

Strategy

Starting strategy:

- **first challenge is to get any non-trivial learning, i.e., for the network to achieve results better than chance**

- **start with simple model and possibly cut your training set to speed up training and making rapid experiments with network architecture**

- **analyse errors**

ϵ : error on training data (sometimes called ‘bias’)

Δ : *[error on validation data]*$-\epsilon$ (sometimes called ‘variance’):

$\hat{\epsilon}$: optimal error rate (e.g. “unavoidable bias” defined by stochastic nature of the problem), $\hat{\epsilon}$ or upper limit may be known from independent study

Error analysis

Techniques for reducing avoidable bias $\epsilon - \hat{\epsilon}$

- **Increase the model size (such as number of neurons/layers)**
- **Modify input features based on insights from error analysis, possibly add extra features**
- **Reduce or eliminate regularization (L1,L2 regularization, dropout)**

Techniques for reducing variance Δ

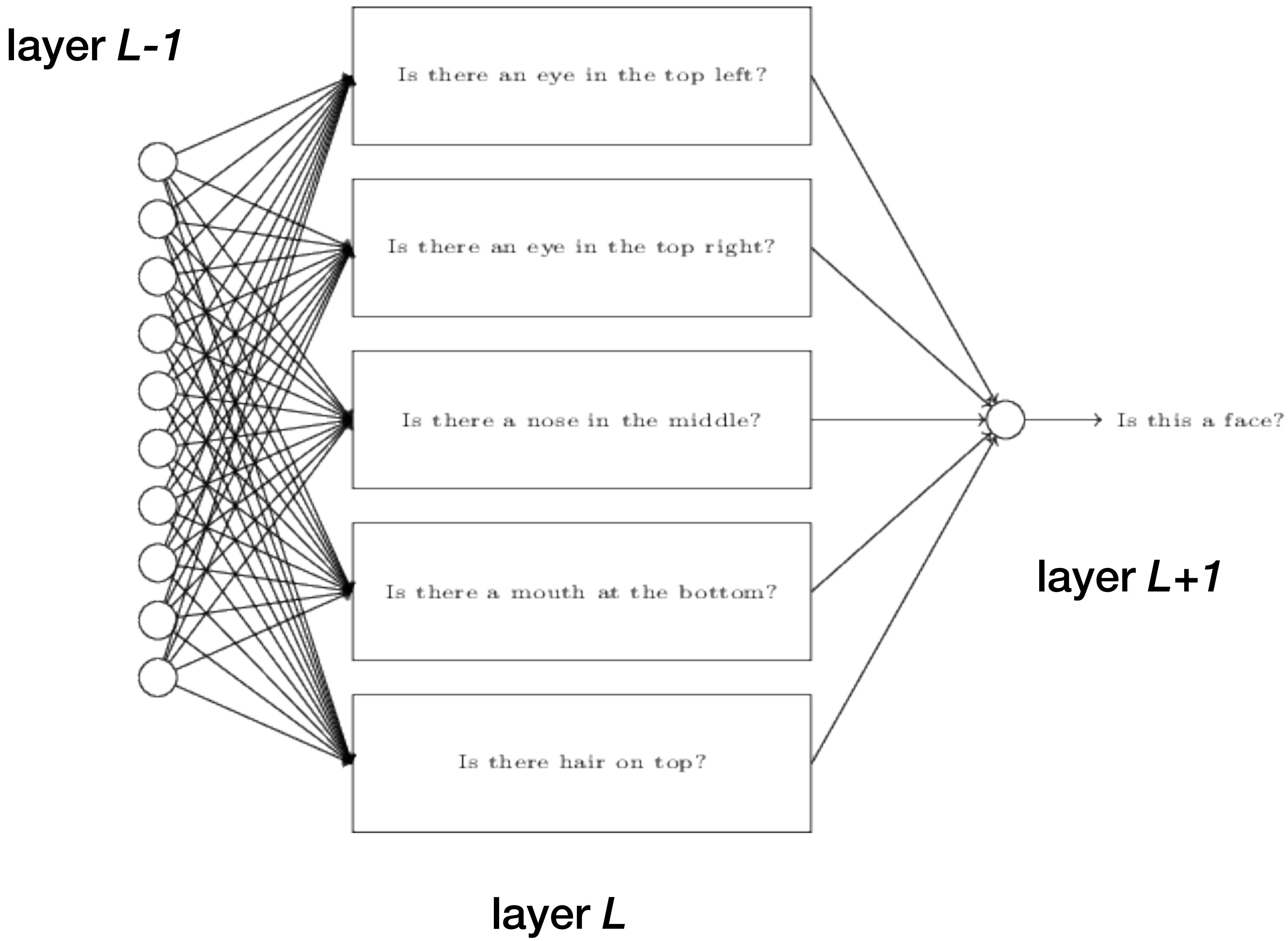
- **Add more training data**
- **Add regularization**
- **Add early stopping**
- **Feature selection to decrease number/type of input features**
- **Decrease the model size (such as number of neurons/layers)**
- **Reduce or eliminate regularization (L1,L2 regularization, dropout)**

Architecture choice. Deep vs shallow ANN.

Deep ANN:

implement more complicated logic

reason: each layer adds a level of abstraction

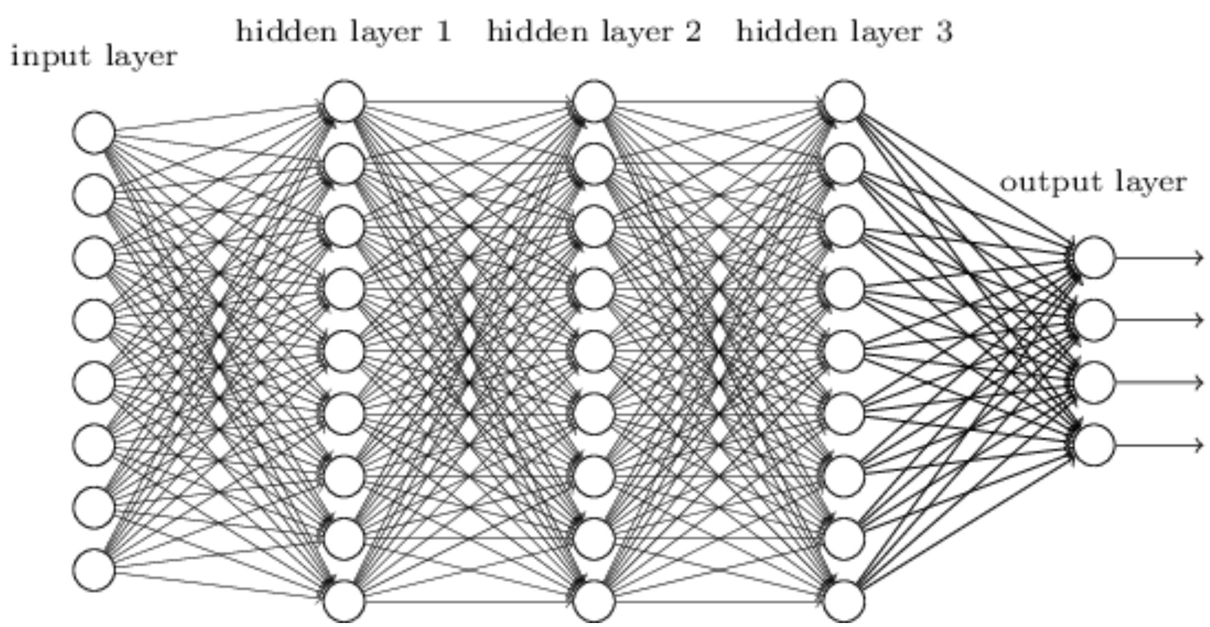


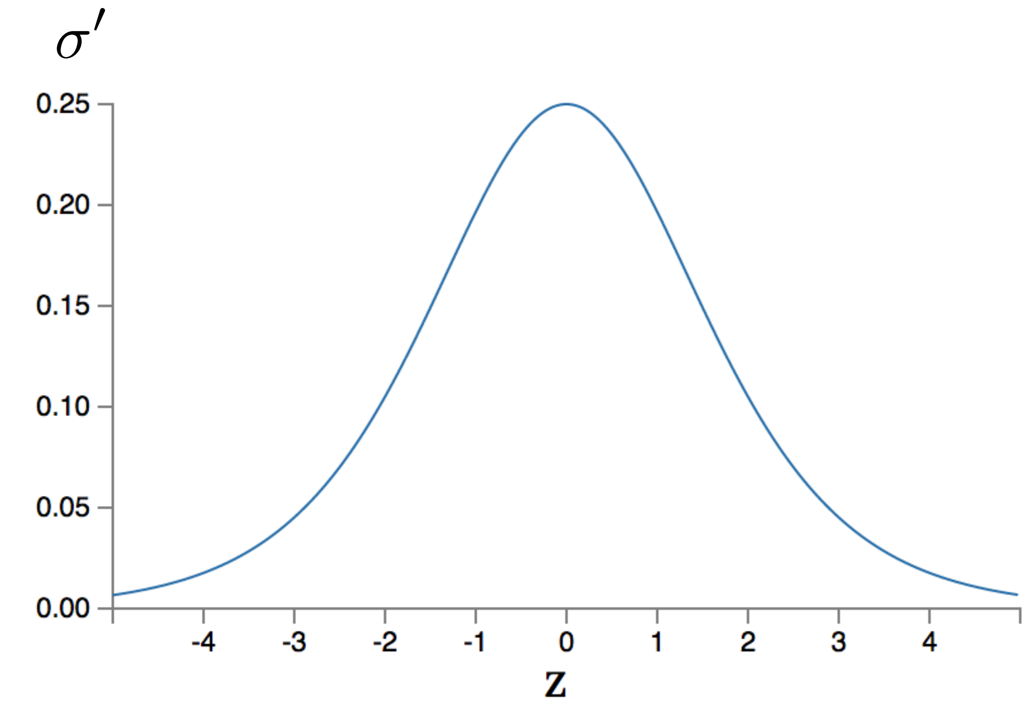
Architecture choice — deep vs shallow ANN.

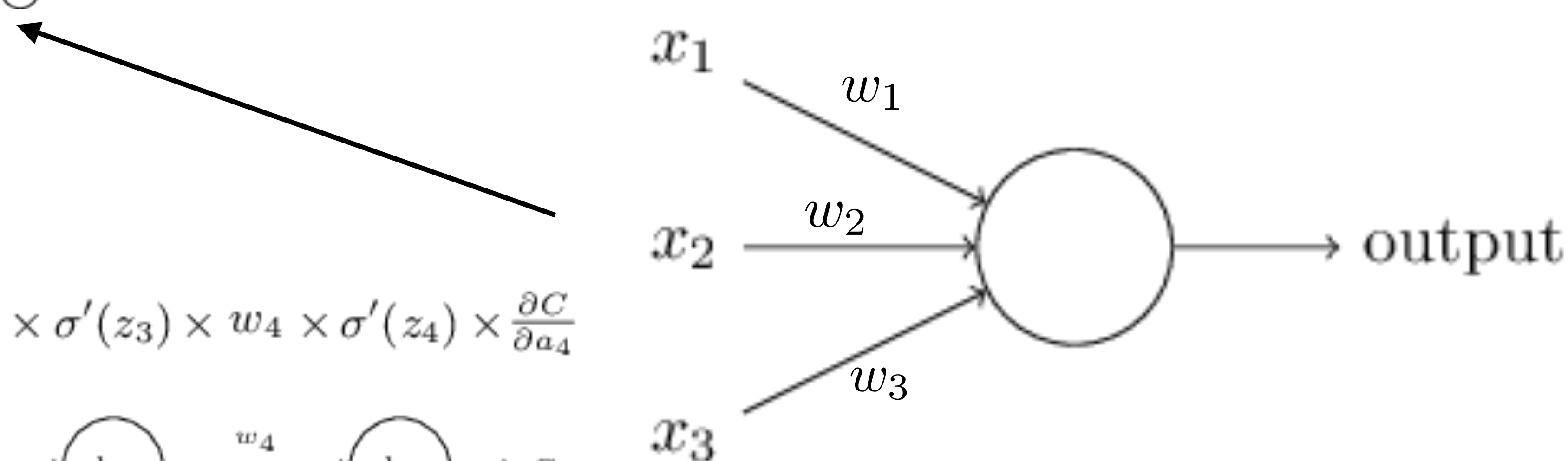
Deep ANN:

implement more complicated logic

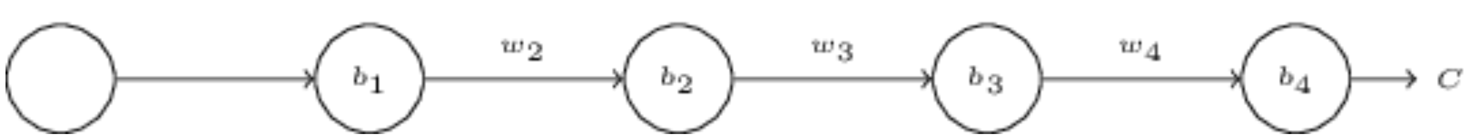
harder to train (vanishing gradient, rounding error)







$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$

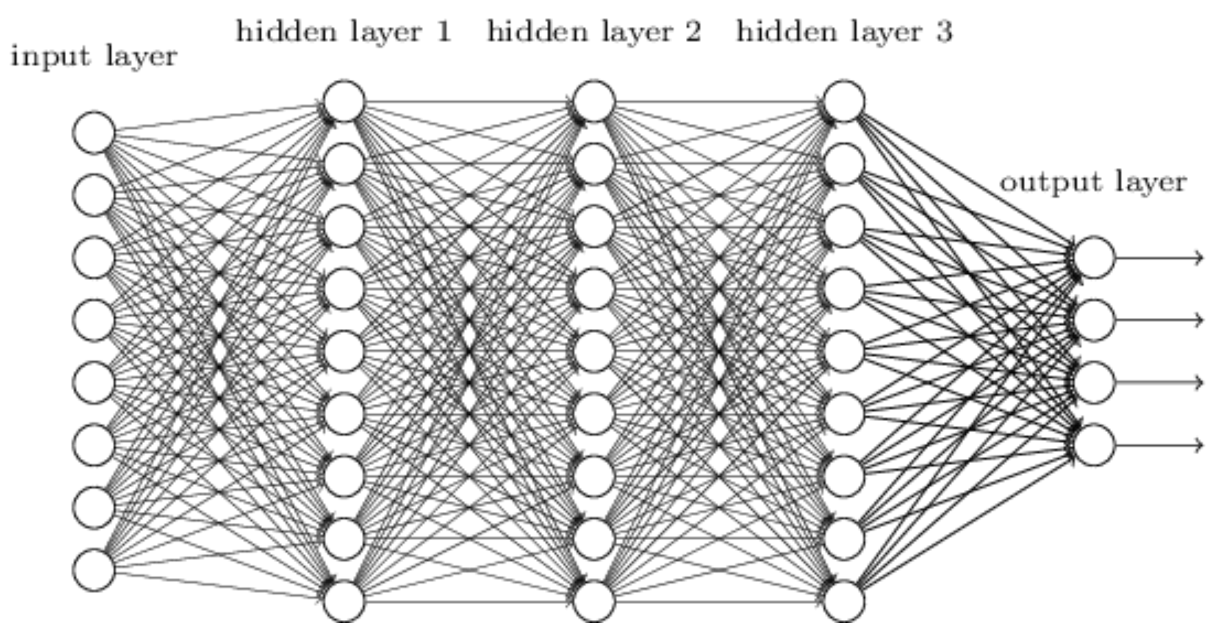


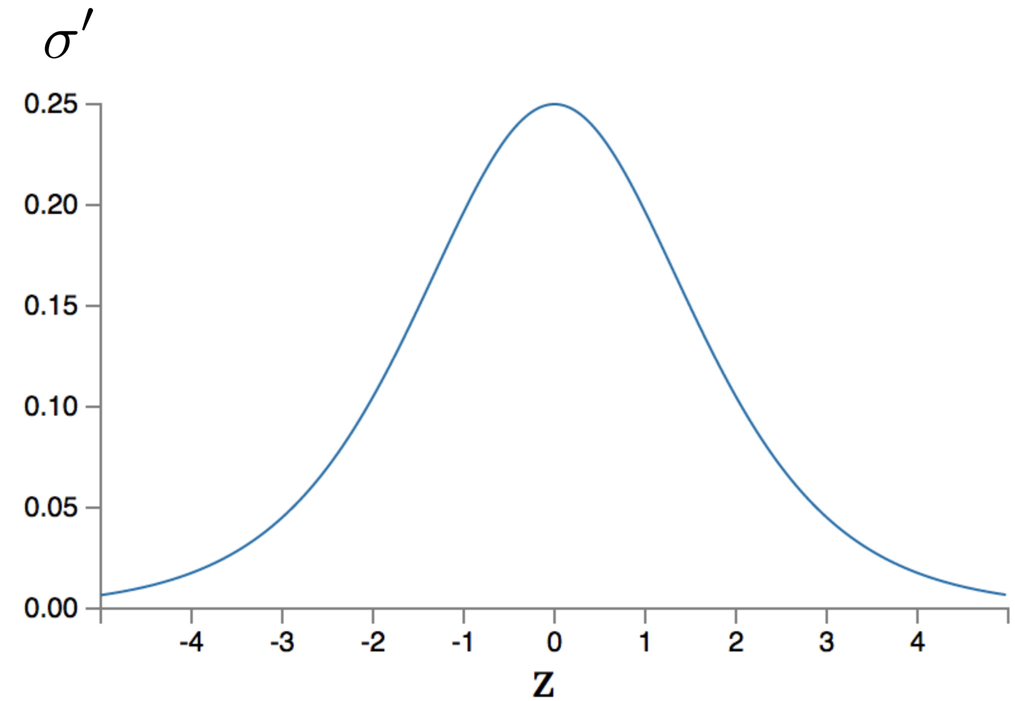
Architecture choice — deep vs shallow ANN.

Deep ANN:

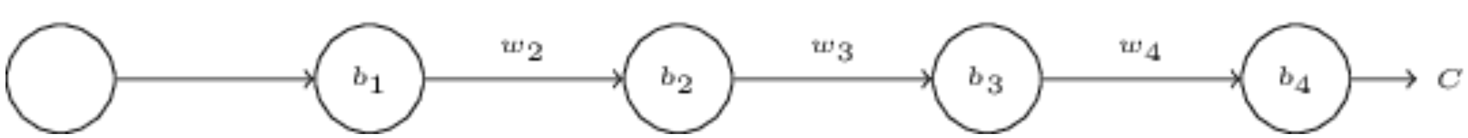
implement more complicated logic

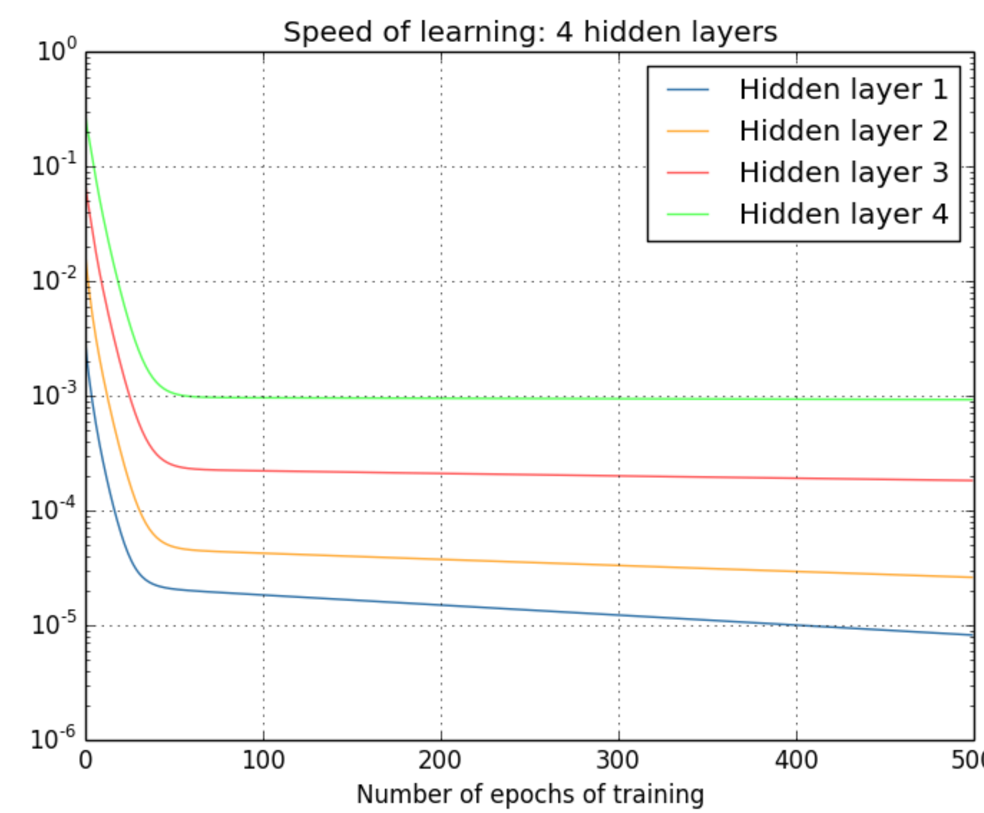
harder to train (vanishing gradient, rounding error)





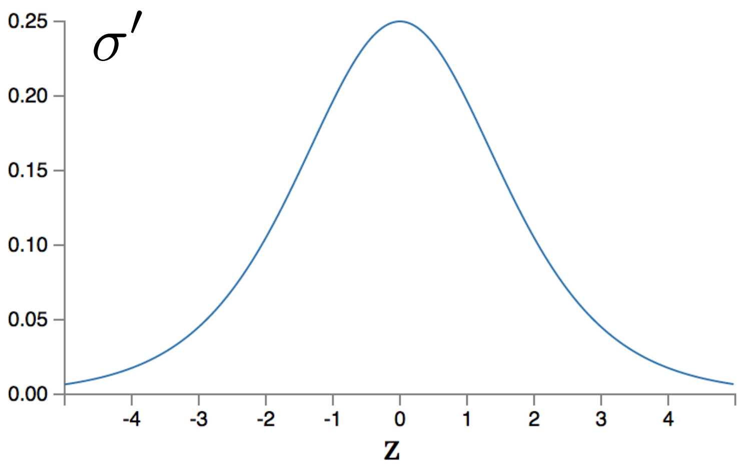
$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



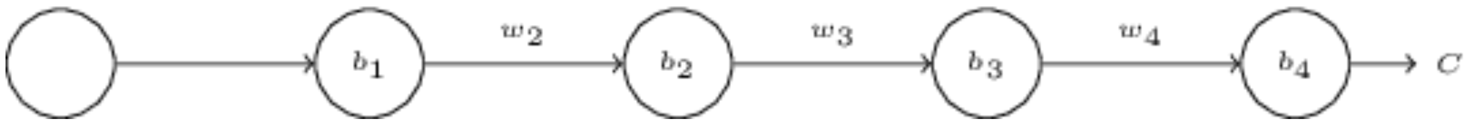


Vanishing gradient

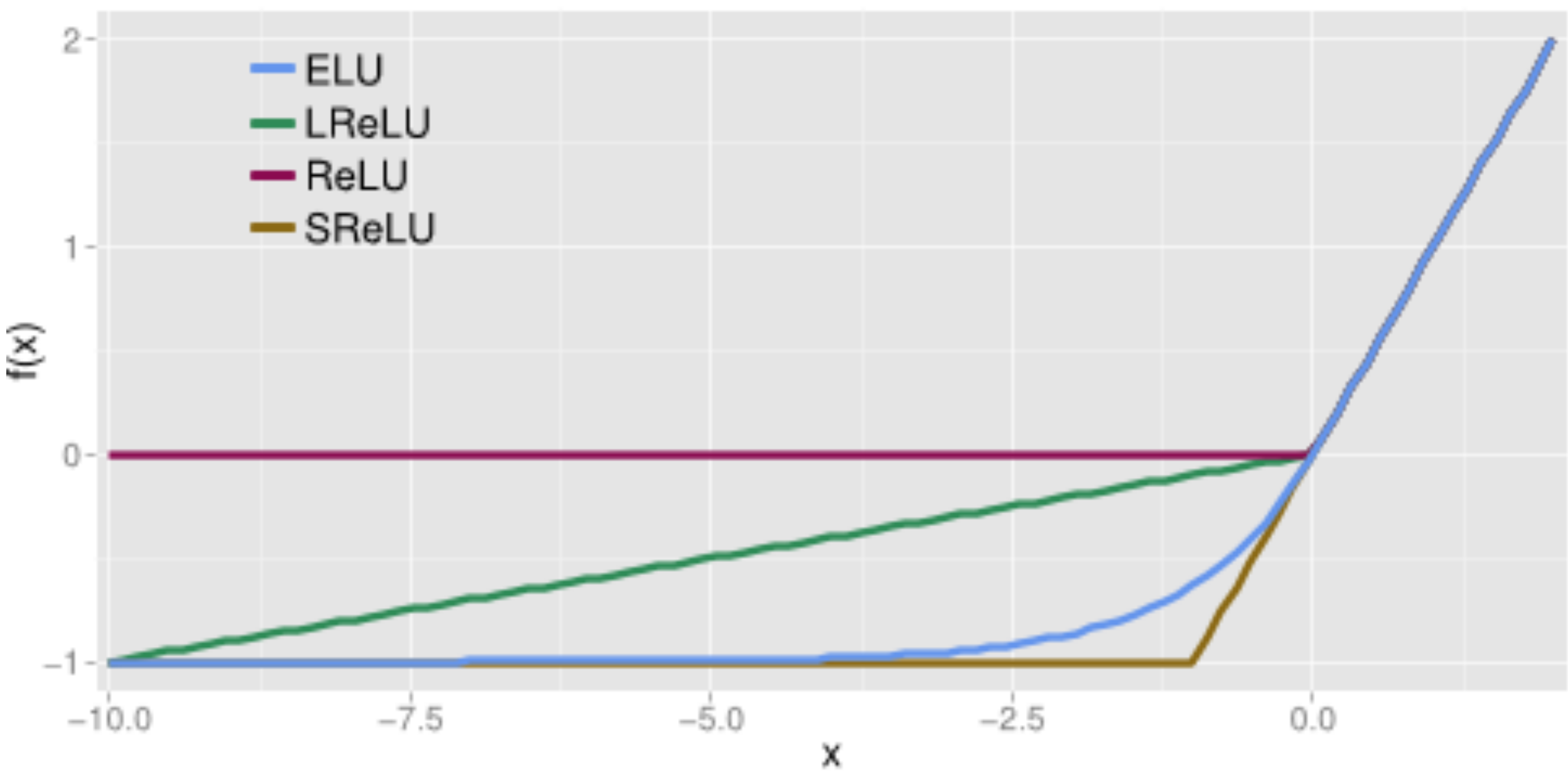
possible solutions



$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



1. Alternative activation function



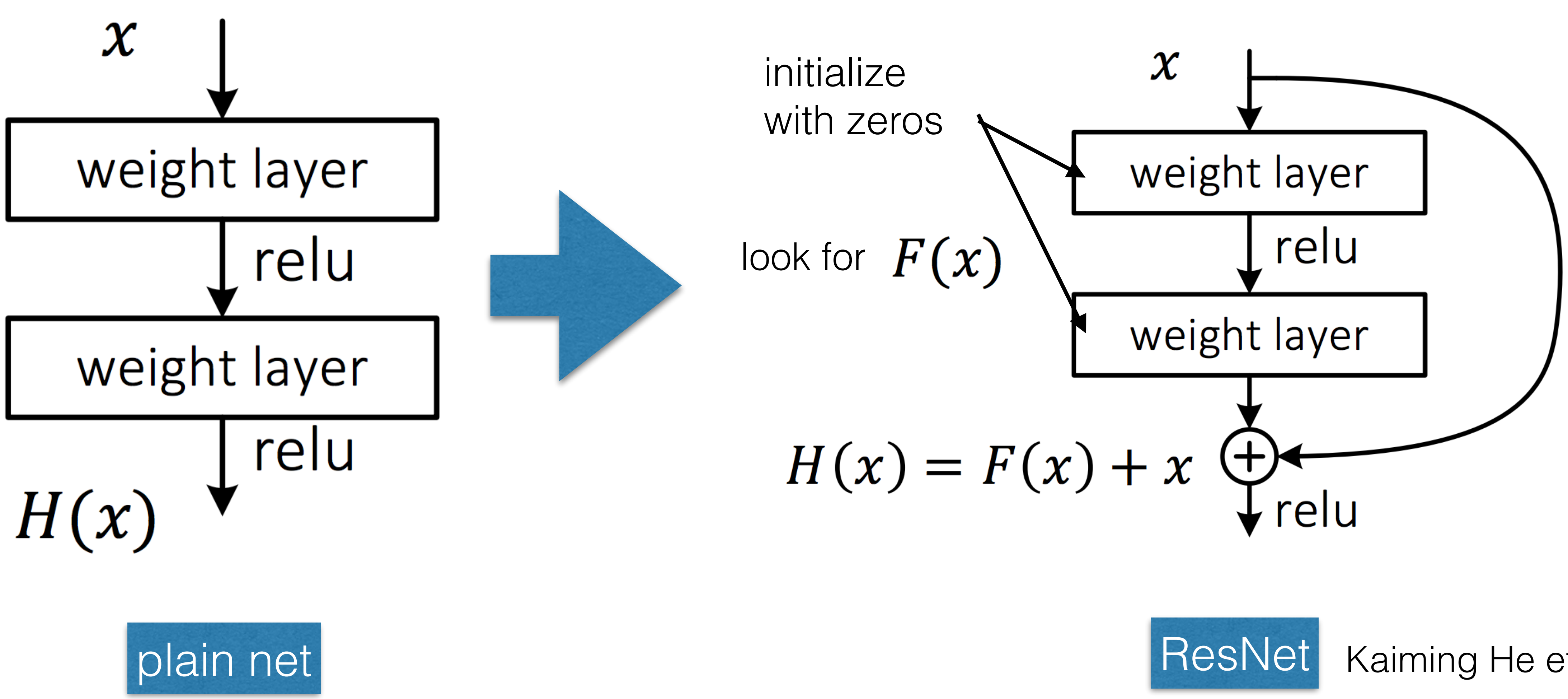
$$f' \simeq 1$$

Vanishing gradient

possible solutions

1. Alternative activation function
2. Residual layers

Suppose x is layer output of a pretrained shallow model and we want enhance the model by inserting two extra layers. The new model $H(x)$ can be trained either from scratch or as a correction:



Kaiming He et al 2015

Time to open jupyter notebook

Hackathon Problem

mass composition of Ultra-High Energy Cosmic Rays

Cosmic rays

Different Physics in each energy range:

Solar modulation:

$$10^8 \text{eV} \leq E \leq 10^{11} \text{eV}$$

Galactic sources:

$$10^{11} \text{eV} \leq E \leq 10^{18} \text{eV}$$

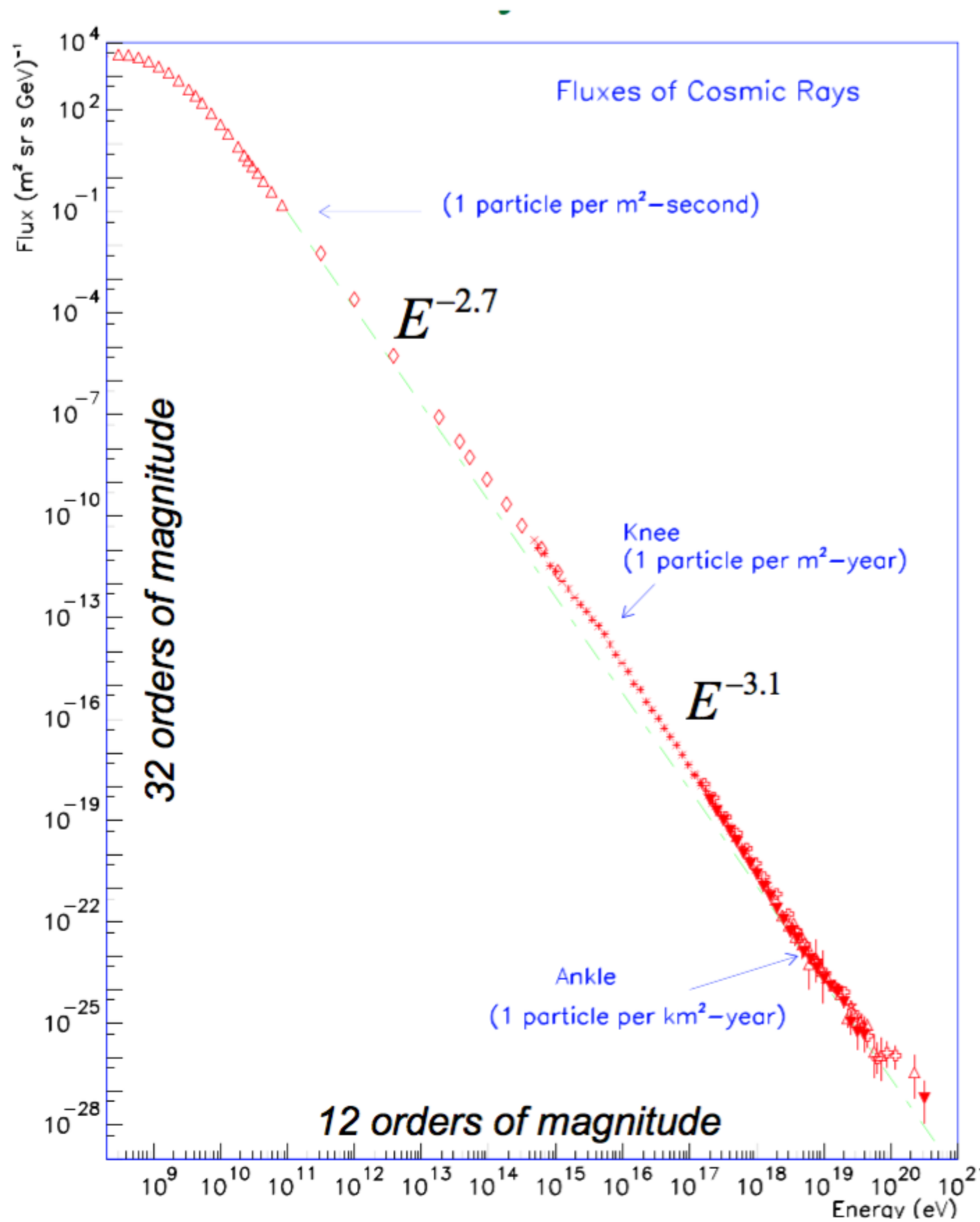
Extragalactic sources:

$$E > 10^{18} \text{eV}$$

GZK cut-off $E \simeq 10^{20} eV$

Problem:

Extremely small flux, hard to observe directly



Questions

- Sources (extragalactic)
- Production mechanism

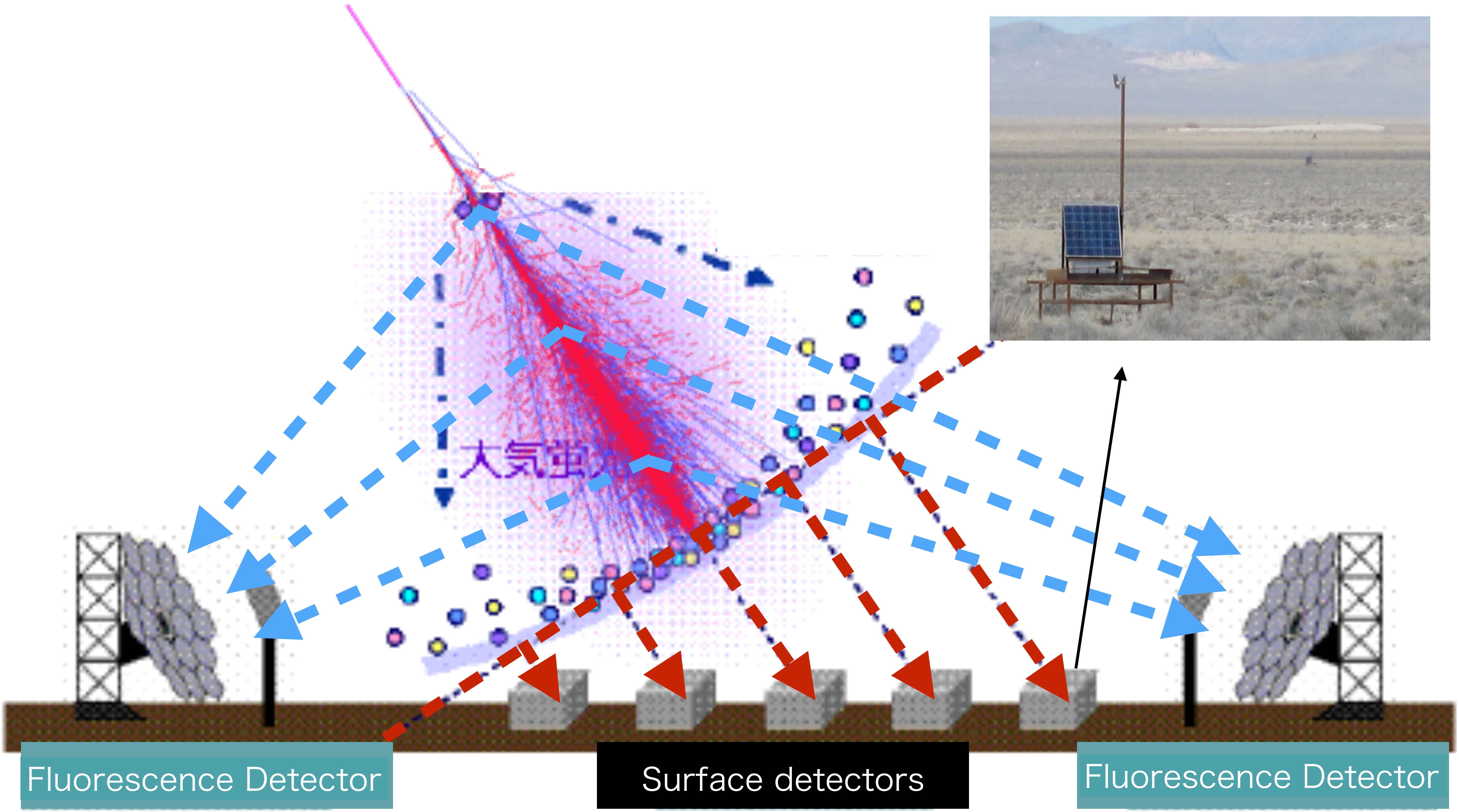
Observables (indirect)

- Energy spectrum
- Mass composition
- Arrival directions

Observables (direct)

- EAS properties observable from Earth (density profile on SD, fluorescence light on FD)

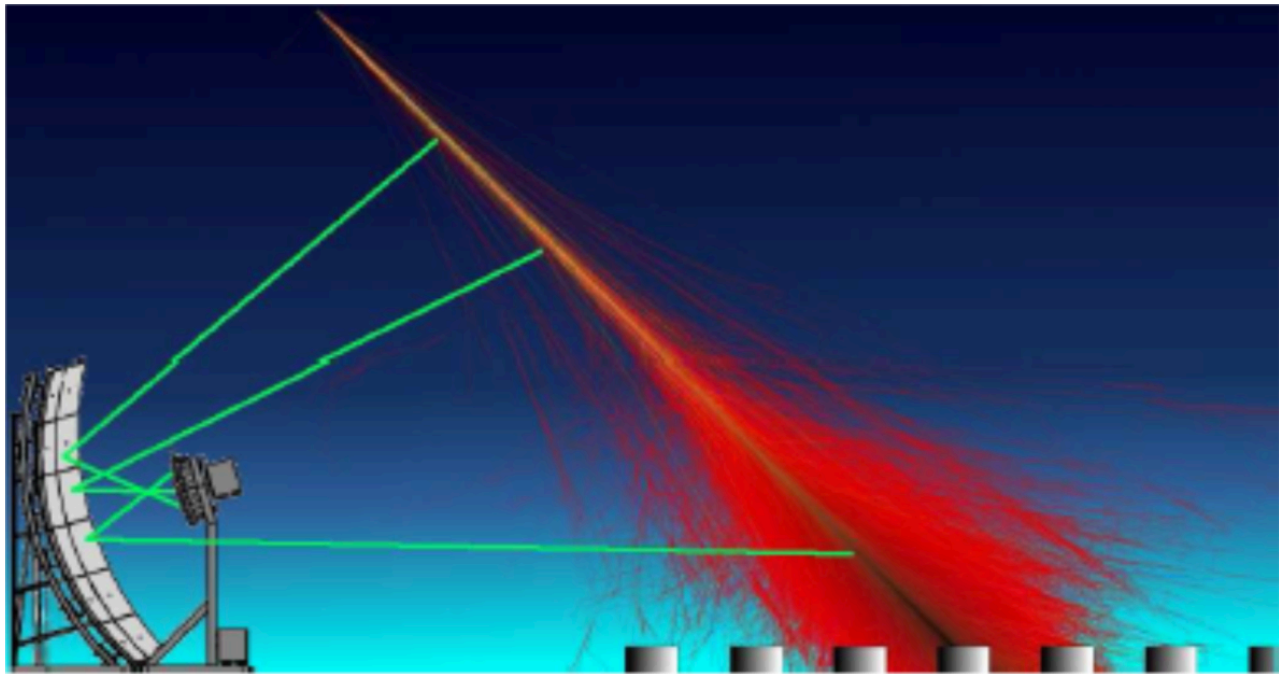
UHECR Detection Methods



Flourescence detectors:
Duty cycle ~ *10%*

Surface detectors:
Duty cycle ~ *95%*

Longitudinal Shower Profiles

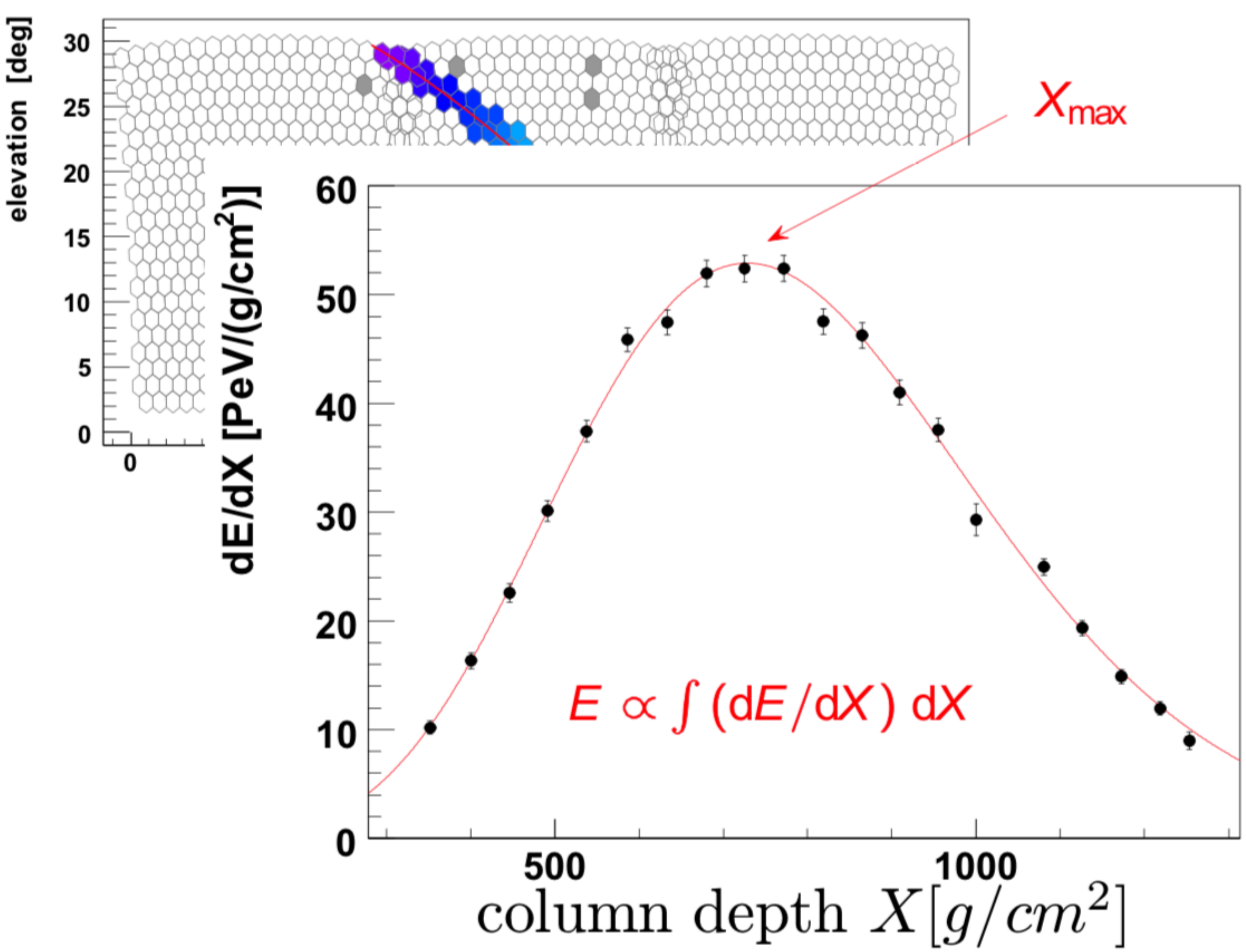


Depth of shower maximum - is good parameter for primary particle mass estimation

$$\langle X_{max} \rangle \propto log(E/A) + const$$

E **- energy**

A **- mass**

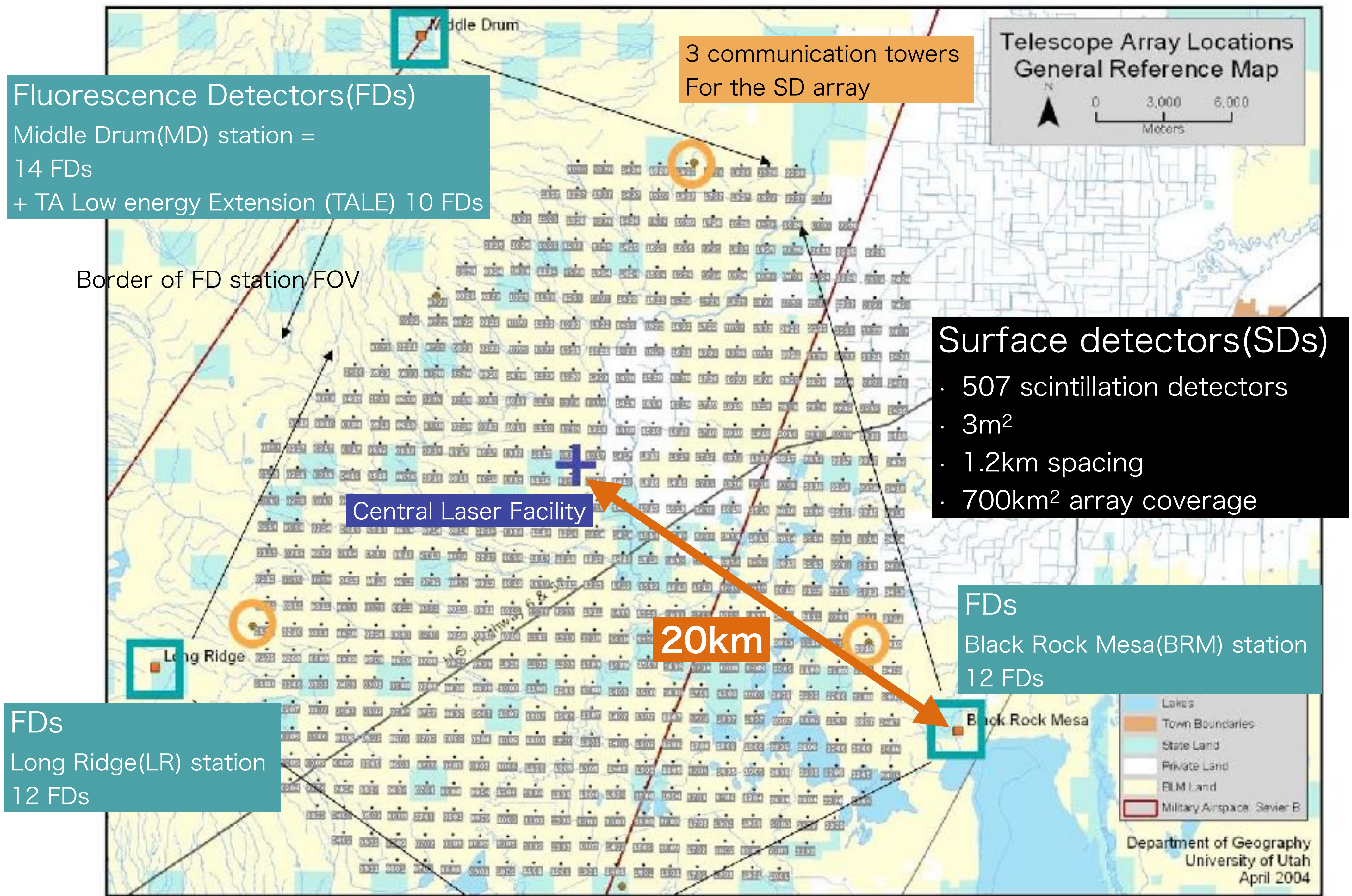


For the surface detector we don't know yet any particular observables as strongly dependent on particle mass as X_{max}

Telescope Array

- The biggest experiment in the northern hemisphere (Utah, USA).

USA,
Russia,
Japan,
Korea,
Belgium



Event reconstruction

standard parametric approach

- LDF

$$f\left(r\right)=\left(\frac{r}{R_m}\right)^{-1.2}\left(1+\frac{r}{R_m}\right)^{-(\eta-1.2)}\left(1+\frac{r^2}{R_1^2}\right)^{-0.6}$$

$$R_m=90.0\ \mathrm{m},\ R_1=1000\ \mathrm{m},\ R_L=30\ \mathrm{m},\quad \eta=3.97-1.79\left(\sec\left(\theta\right)-1\right),$$

$$r=\sqrt{\left(x_{\mathrm{core}}-x\right)^2+\left(y_{\mathrm{core}}-y\right)^2},$$

- Timing

$$t_r=t_o+t_{plane}+a\times\left(1+r/R_L\right)^{1.5}LDF\left(r\right)^{-0.5}$$

$$LDF\left(r\right)=f\left(r\right)/f\left(800\ \mathrm{m}\right)\qquad S\left(r\right)=S_{800}\times LDF\left(r\right)$$

Free parameters:

$x_{core},\ y_{core},\ \theta,\ \phi,\ S_{800},\ t_0,\ a$

Observables:

t_r - detector time

S_r - detector integral signal

Problem

Task 1: determine average mass and fractions of elements in a test set

Task 2: in a test set containing protons with small admixture of photons find photon candidate events

Training data: samples for primary H, He, N, Fe and γ containing 16 observables:

1. θ zenith angle
2. S800
3. number of detectors hit, #4 in Ref. [1],
4. number of detectors excluded from fit, #5 in in Ref. [1]
5. $\chi^2/n.d.f.$, #6 in Ref. [1]
6. shower front curvature *a*, #1 in Ref. [1]

7-8. Area-over-peak and it's slope, #2-3 in Ref. [1]

9-10. Sb, b=3 and b=4.5, #7-8 in Ref. [1]

11. The sum of the signals of all the detectors of the event, #9 in Ref. [1]
12. Asymmetry of the signal at the upper and lower layers of detectors, #10 in Ref. [1]
13. Total number of peaks within all FADC (flash analog-to-digital converter) traces, #11 in Ref. [1]
14. Number of peaks for the detector with the largest signal, #12 in Ref. [1]
15. Number of peaks present in the upper layer and not in the lower, #13 in Ref. [1]
16. Number of peaks present in the lower layer and not in the upper, #14 in Ref. [1]

[1] see Appendix A of http://arxiv.org/abs/arXiv:1808.03680