Machine Learning 2/2

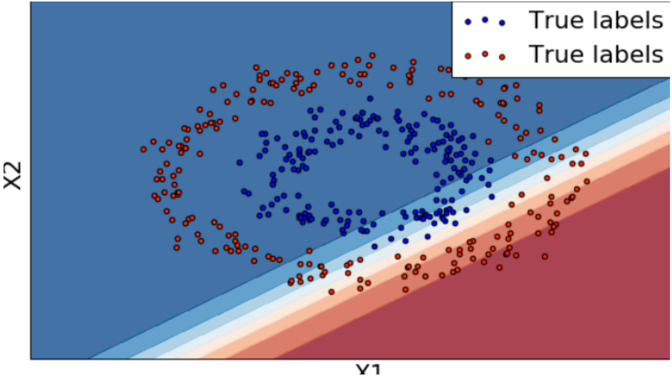
19th JINR-ISU Baikal Summer School on Physics of Elementary Particles and Astrophysics

Denis Derkach

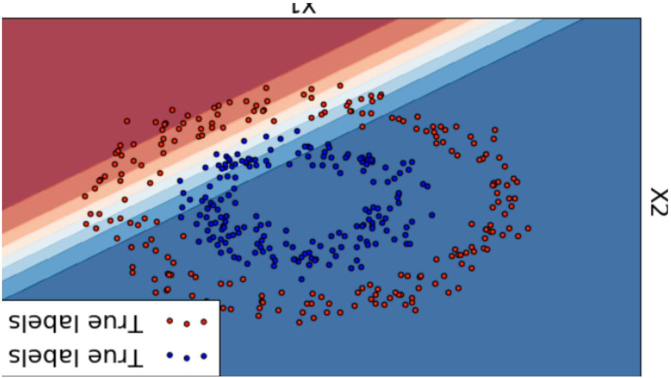
Contents

Neural Network Construction

The logistic regression model decision rule



The logistic regression model decision rule



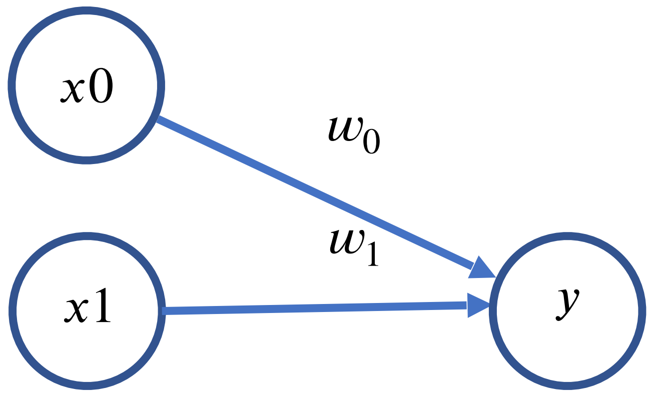
The decision boundary for this particular dataset could be put in different points.

How things work?

Remember a simple regression problem:

$$y = w_0 + w_1 x_1$$

Schematically, it can be written out as (let’s put x_0=1):

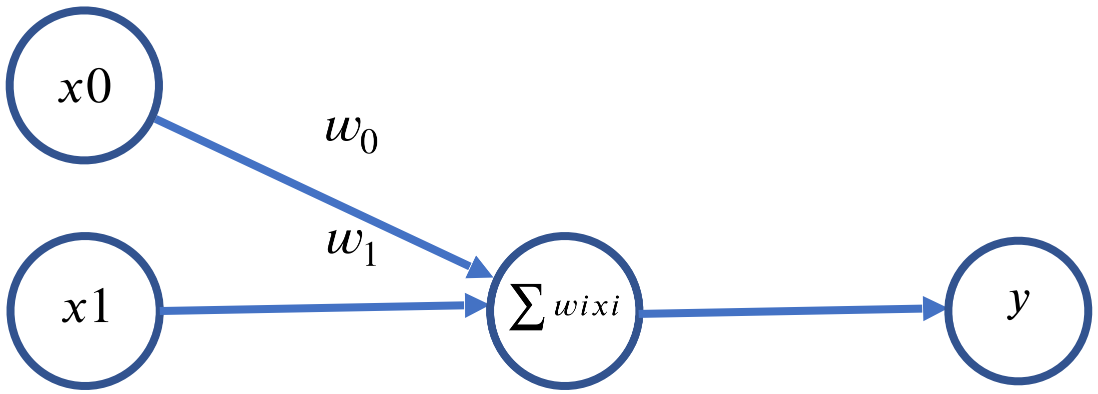


How things work?

Remember a simple regression problem:

$$y = w_0 + w_1 x_1$$

Schematically, it can be written out as (let’s put x_0=1):

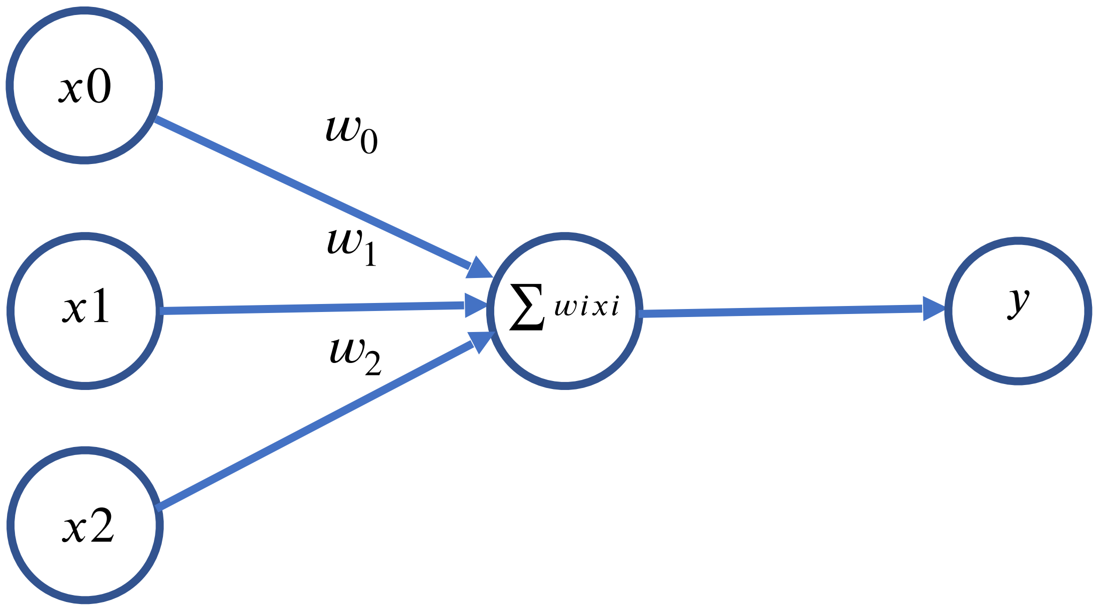


More observables?

What if regression will get multiple inputs.

$$y = w_0 + w_1 x_1 + w_2 x_2$$

Fairly easy, we can represent it in a graphical way:

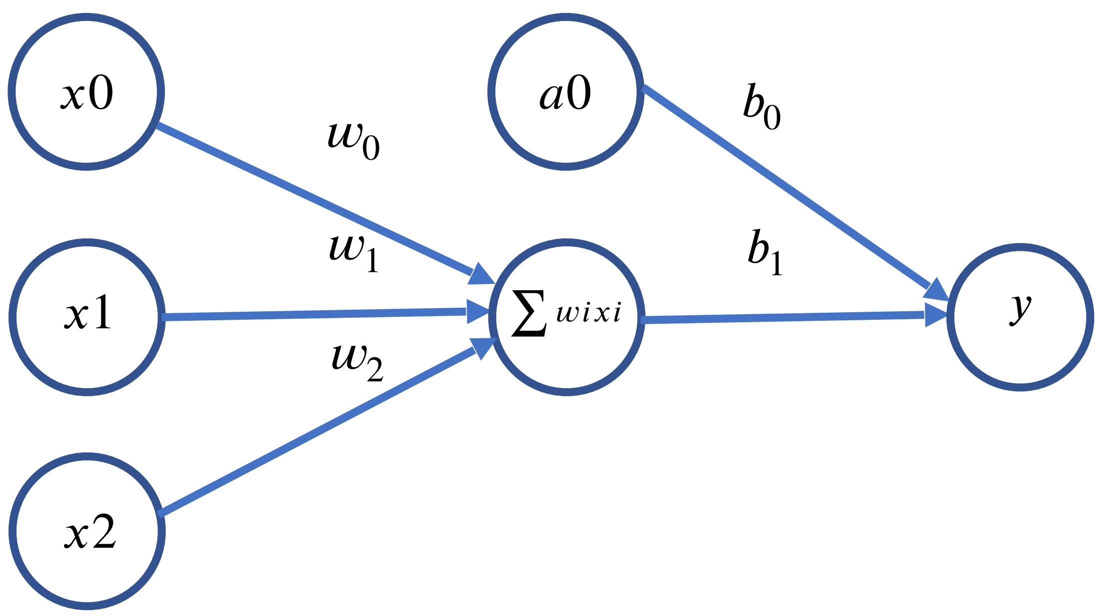


More Regressions?

We can add a similar to x_0 term a_0, we also assign weights v here.

$$y = b_1(w_0 + w_1x_1 + w_2x_2) + b_0$$

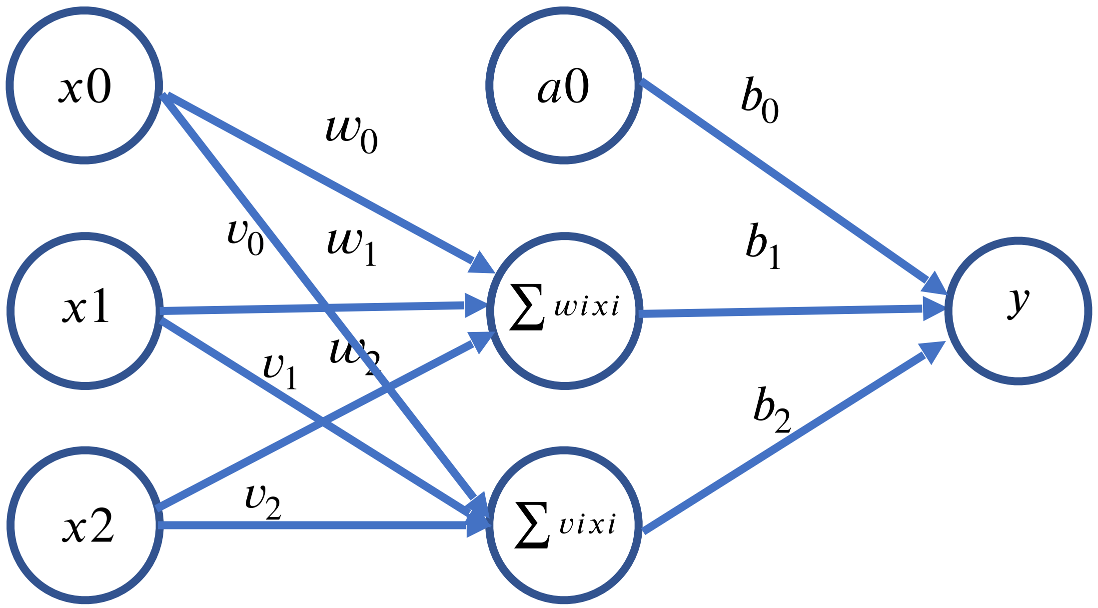
To represent a final calibration.



What else can be added?

We can add a second regression

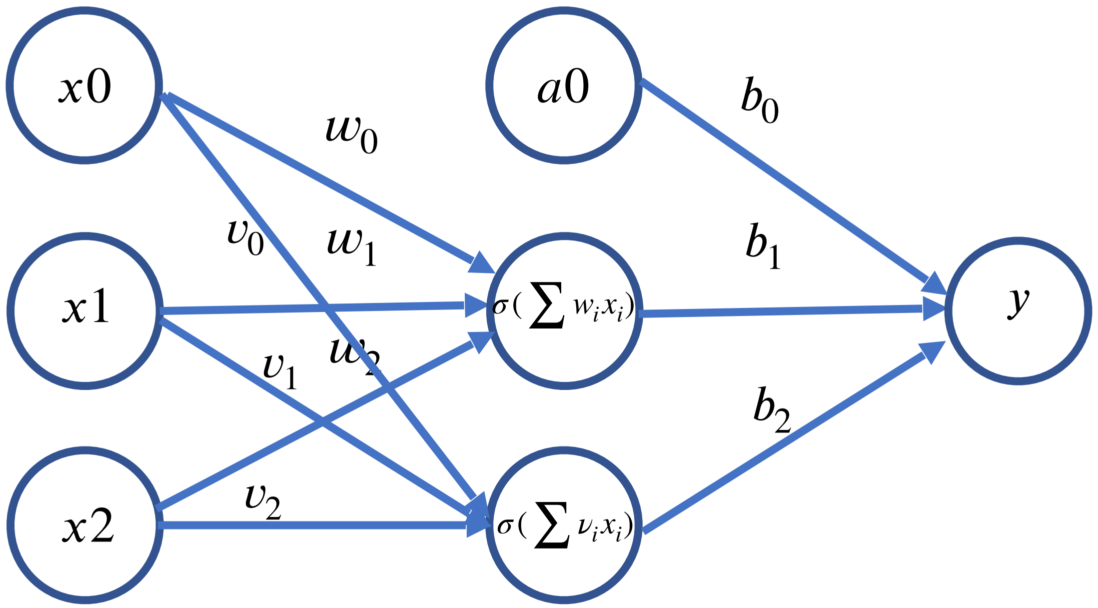
$$y = b_2(\nu_0 + \nu_1 x_1 + \nu_2 x_2) + b_1(w_0 + w_1 x_1 + w_2 x_2) + b_0$$



NonLinearities?

We can add a second regression

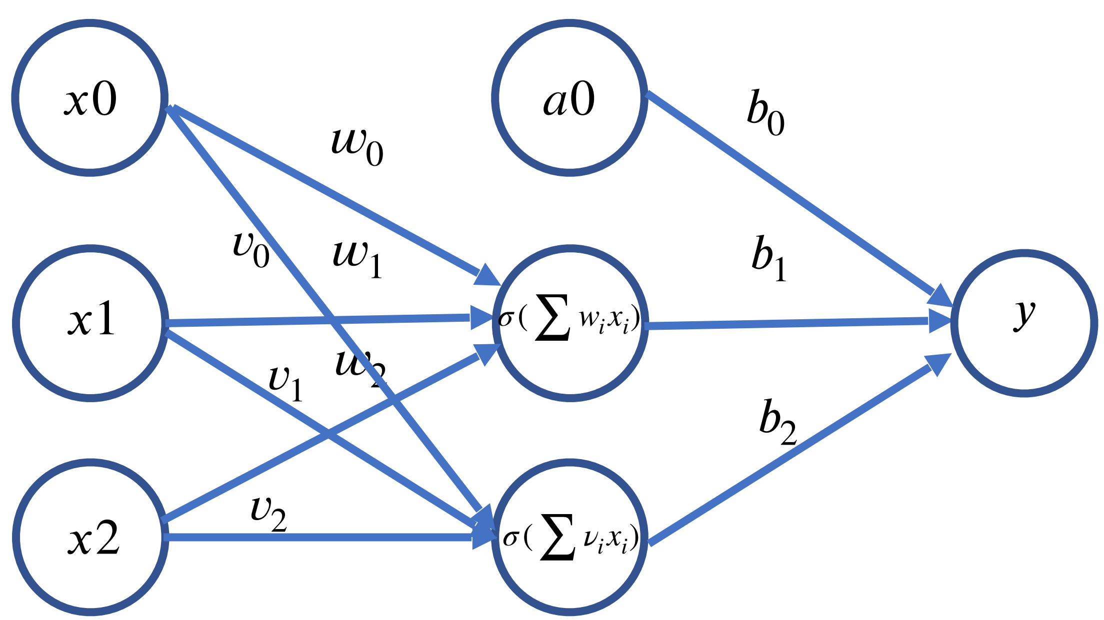
$$y = b_2\sigma(\nu_0 + \nu_1 x_1 + \nu_2 x_2) + b_1\sigma(w_0 + w_1 x_1 + w_2 x_2) + b_0$$



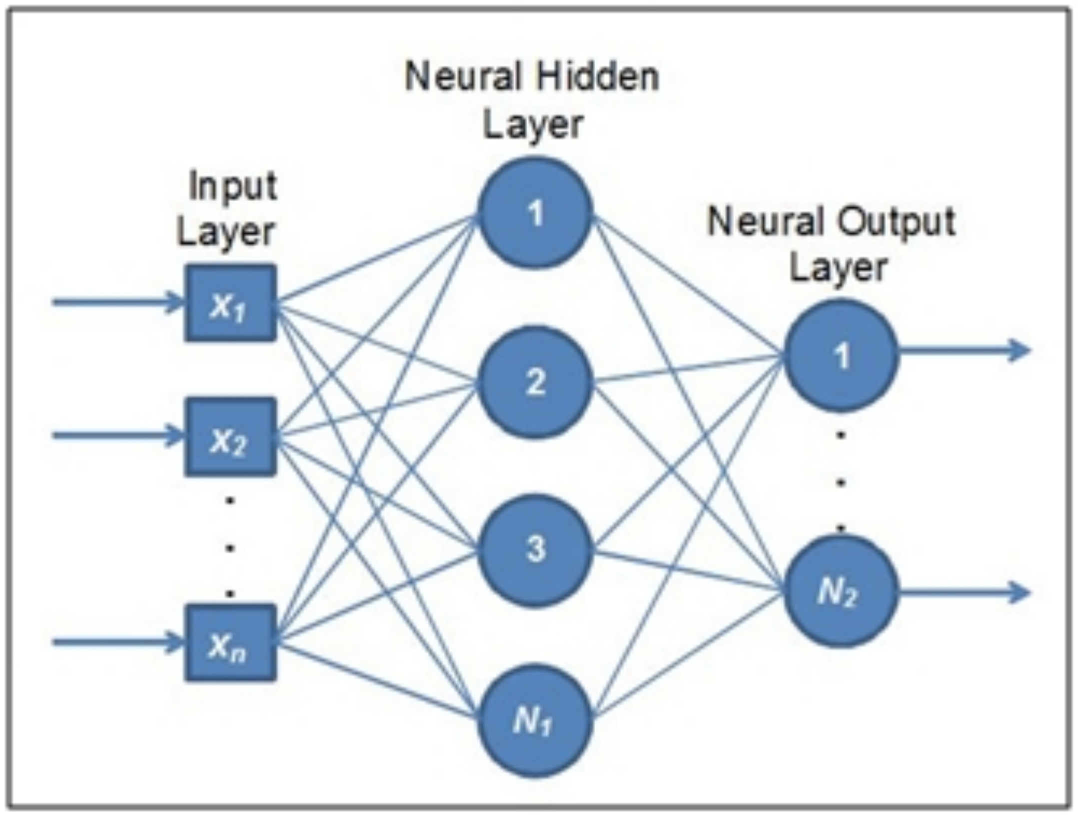
NonLinearities?

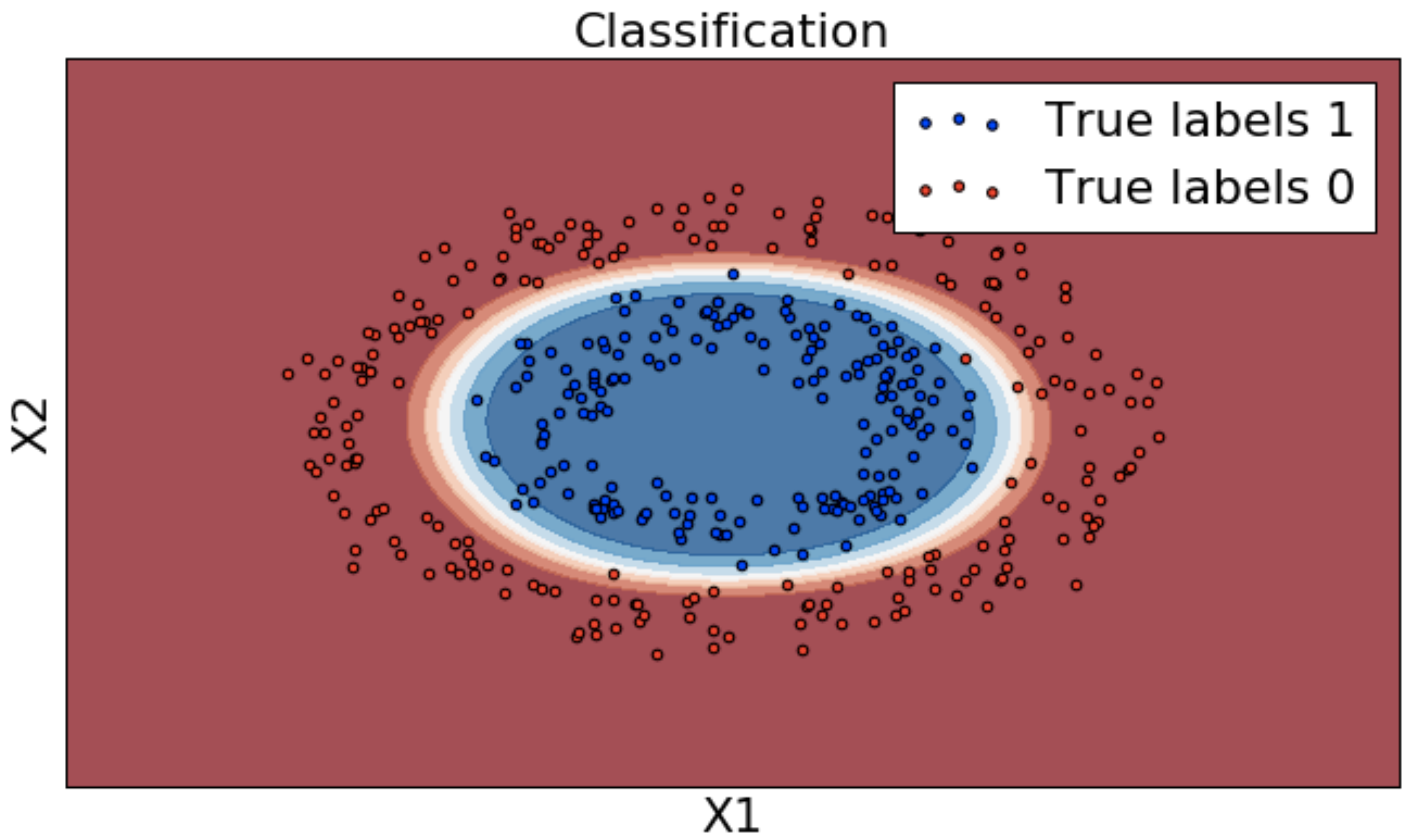
We can add a second nonlinearity?

$$y = \sigma(b_2\sigma(\nu_0 + \nu_1 x_1 + \nu_2 x_2) + b_1\sigma(w_0 + w_1 x_1 + w_2 x_2) + b_0)$$



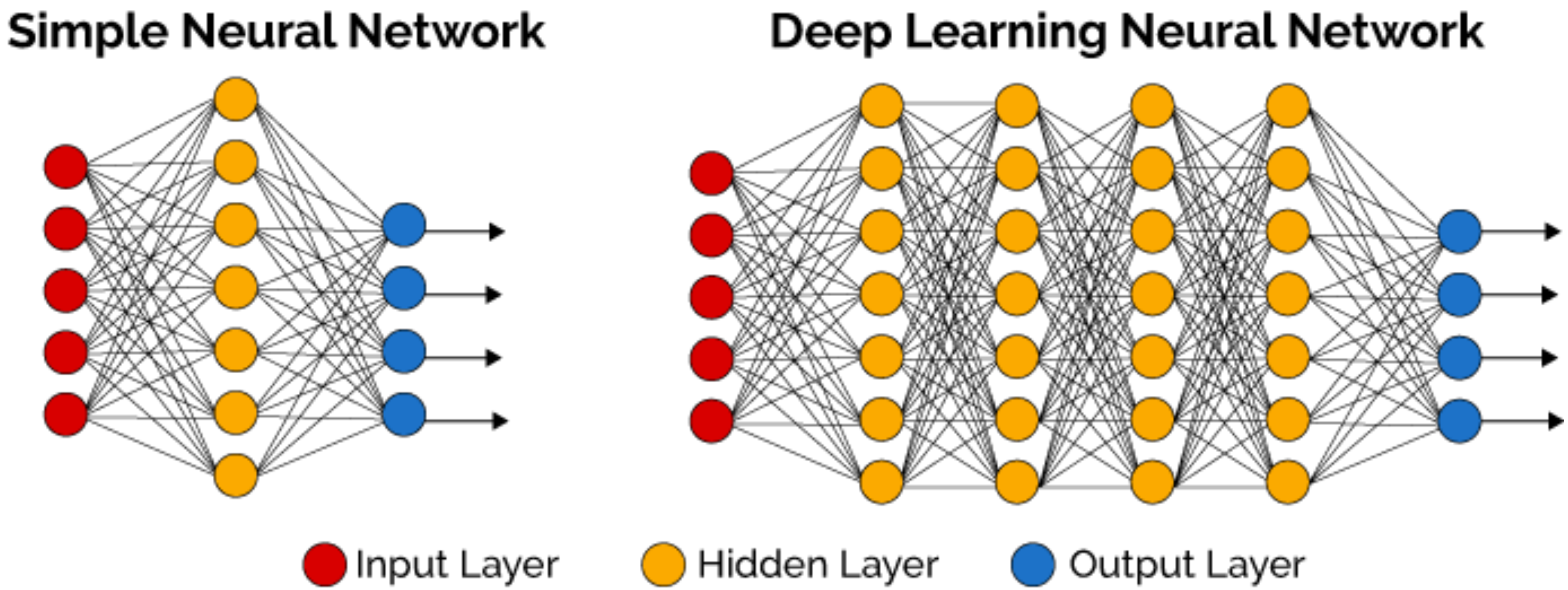
Neural Network!





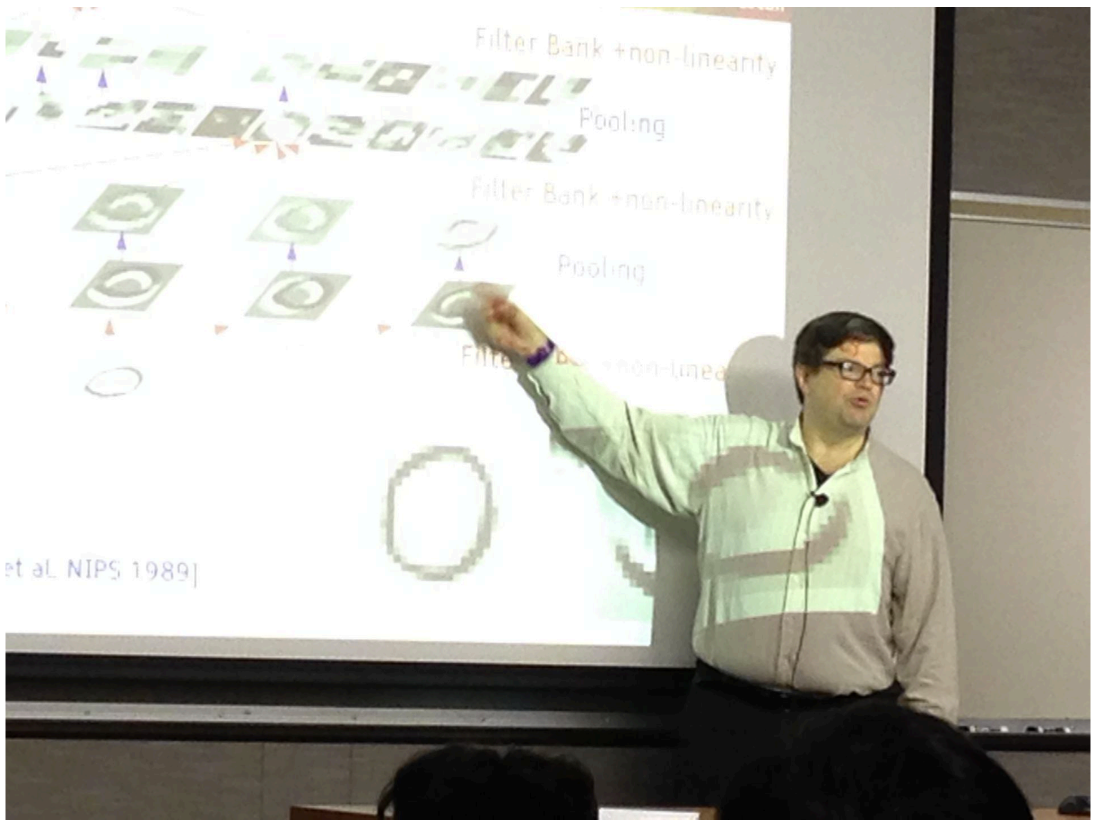
So we just need to have a very big hidden layer?
(and in fact just use many logistic regressions)?

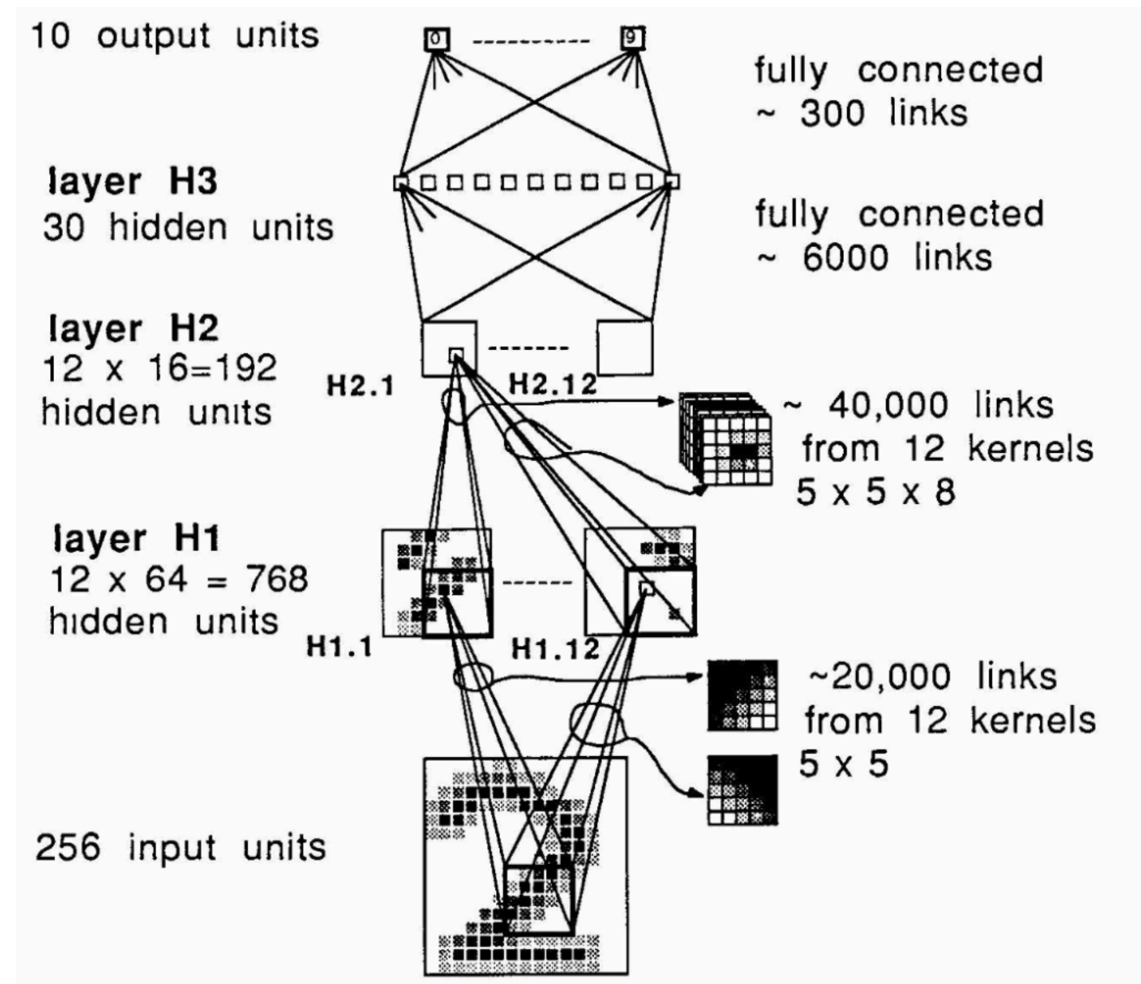
Growing Deeper



How to fit all the weights?

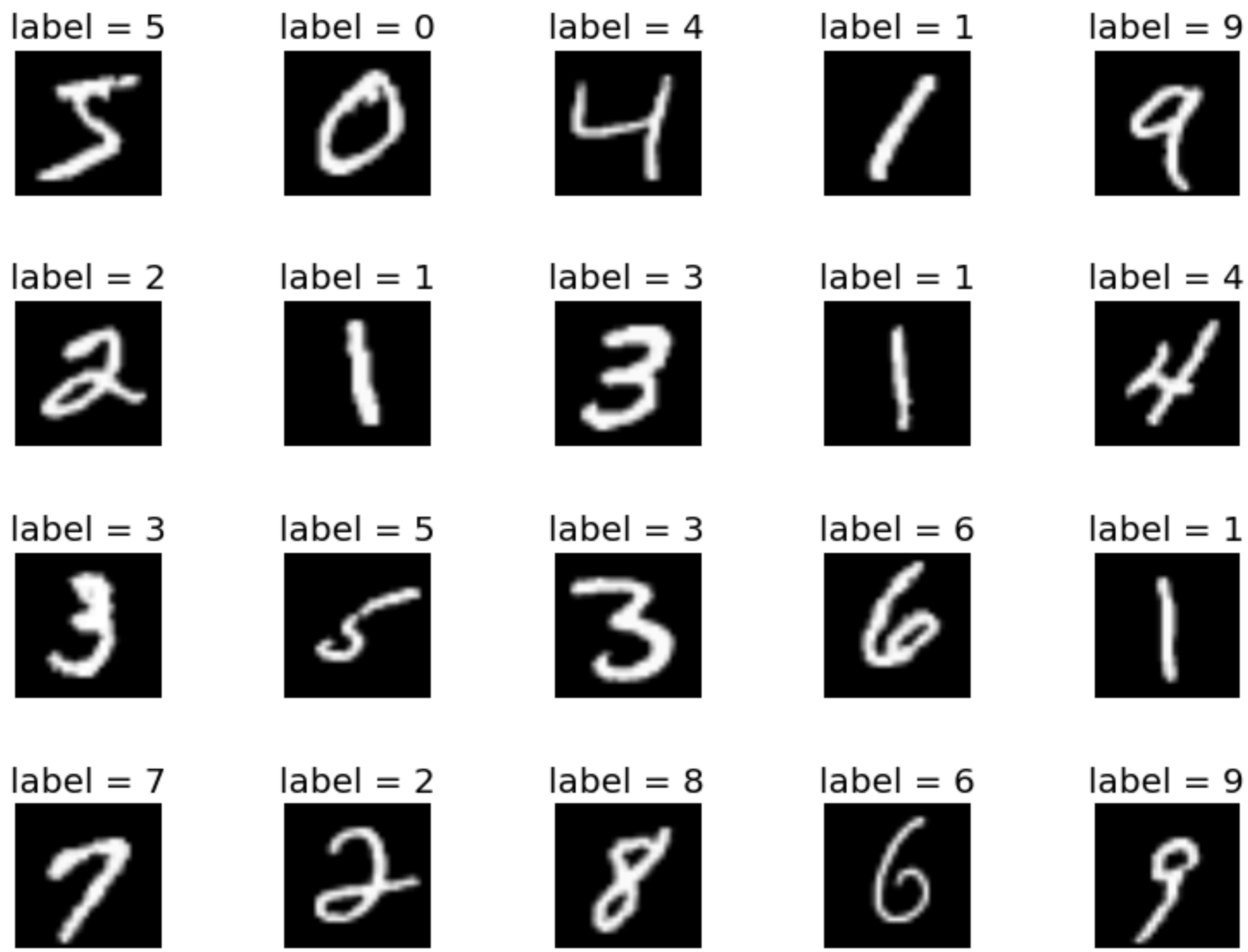
Training Neural Networks





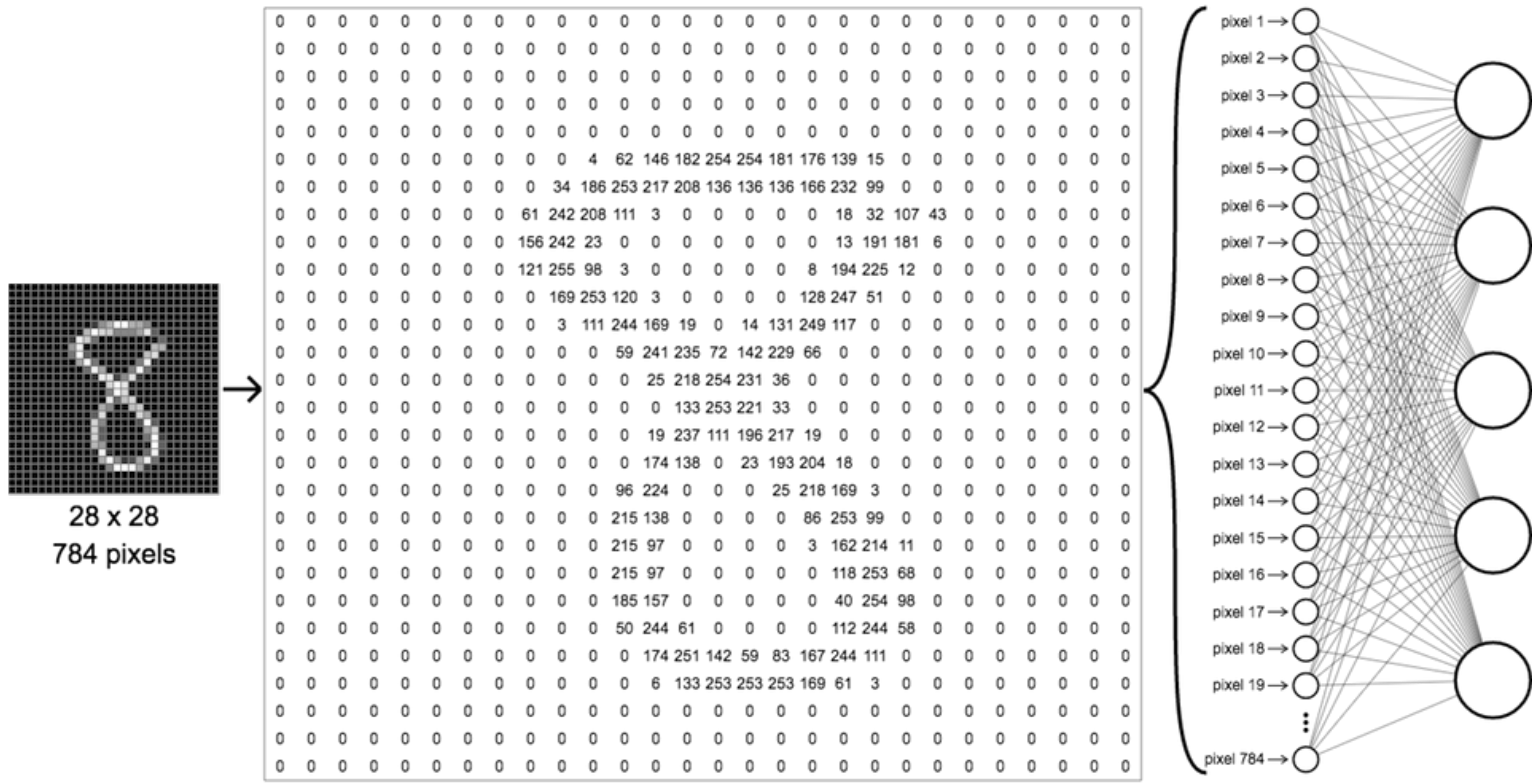
LeCun *et al.*, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, 1, pp. 541–551, 1989

Training Neural Networks



Most probably, we we will train a supervised classification, which means that we not only need images, but also labels.

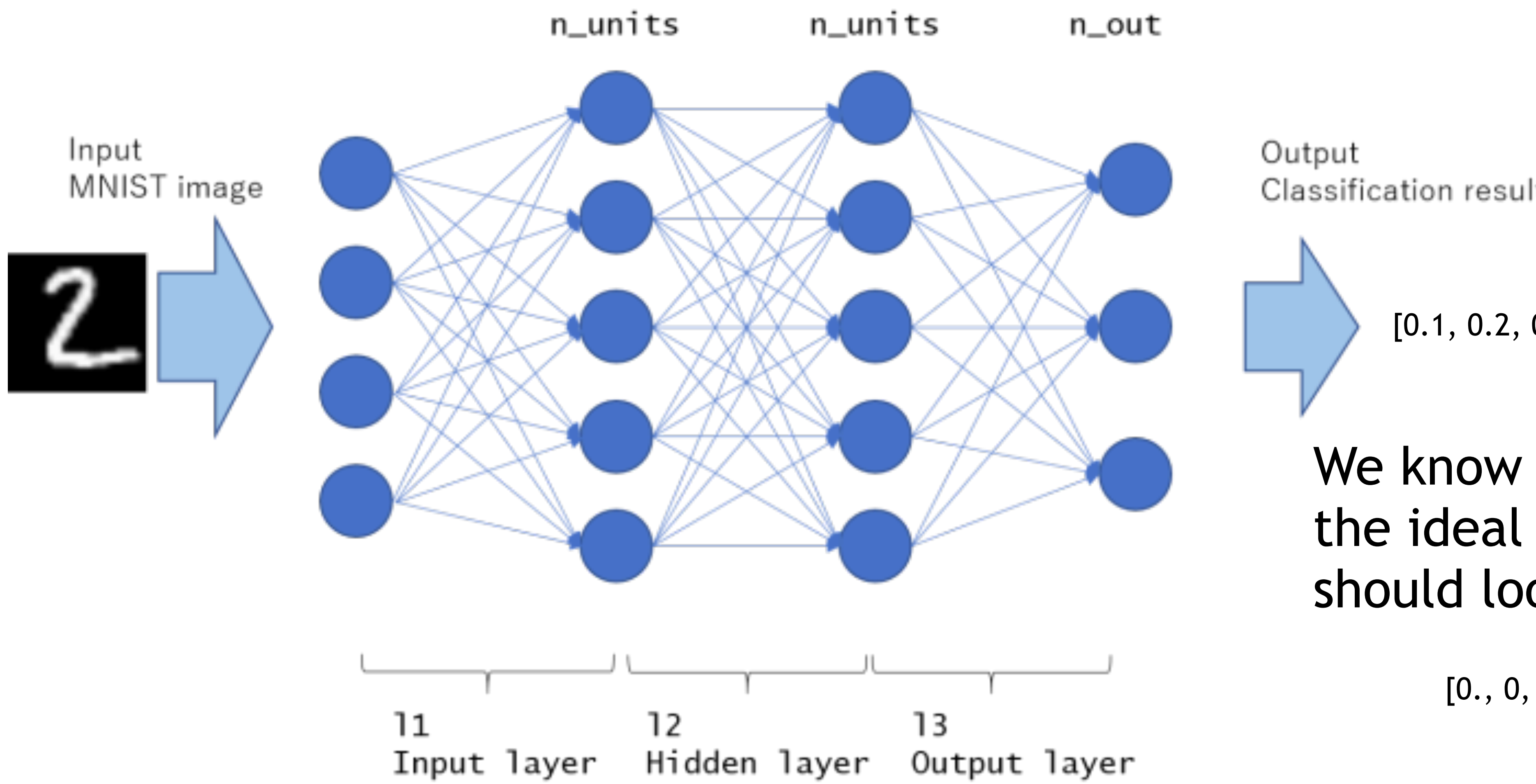
Single digit



Each digit is represented by 28X28 point picture with different brightness.
We can write it as a vector.

Forward propagation

We have an untrained network and **forward** propagate an image, which is known to be “2” (remember, we have labeled dataset)



[0.1, 0.2, 0.2 , 0.2, 0.1, 0.3, 0.5, 0.3, 0.2]

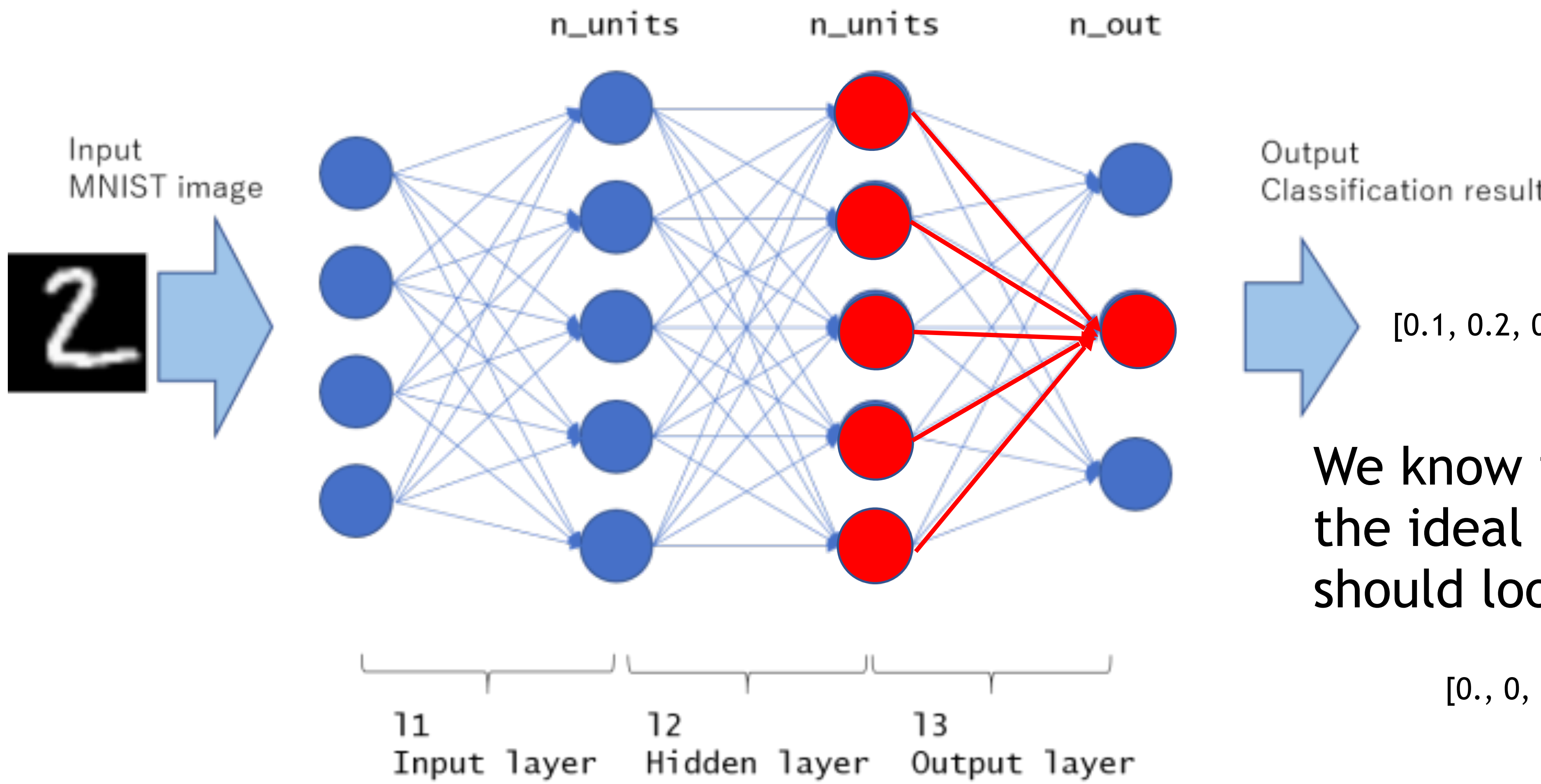
We know that that the ideal output should look like this:

[0., 0, 1, 0, 0. 0, 0, 0, 0]

https://www.youtube.com/watch?v=qtyiJTMcxao

Backpropagation

Remember that we started from regression.

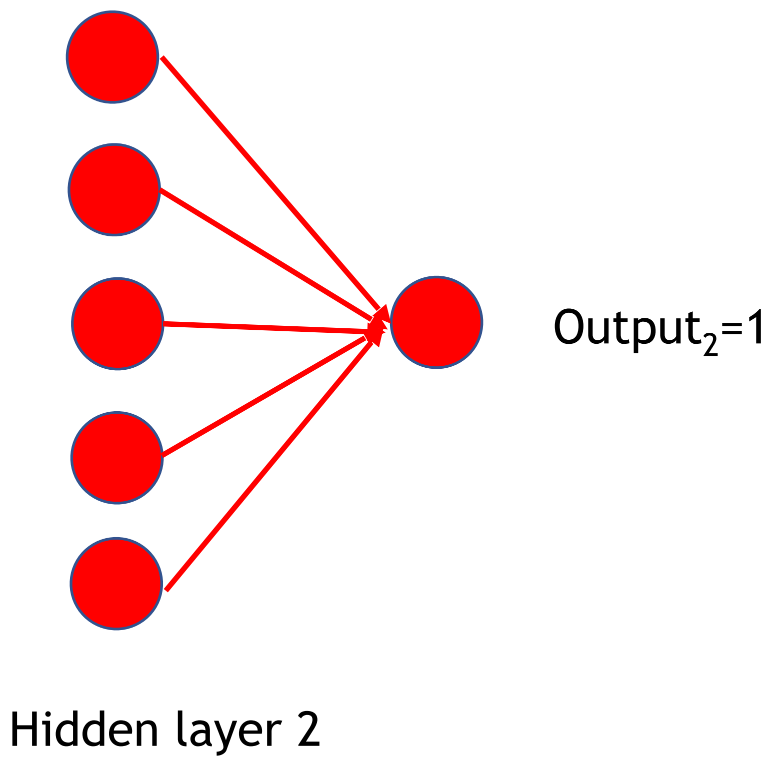


[0.1, 0.2, 0.2 , 0.2, 0.1, 0.3, 0.5, 0.3, 0.2]

We know that that the ideal output should look like this:

[0., 0, 1, 0, 0. 0, 0, 0, 0]

Backpropagation



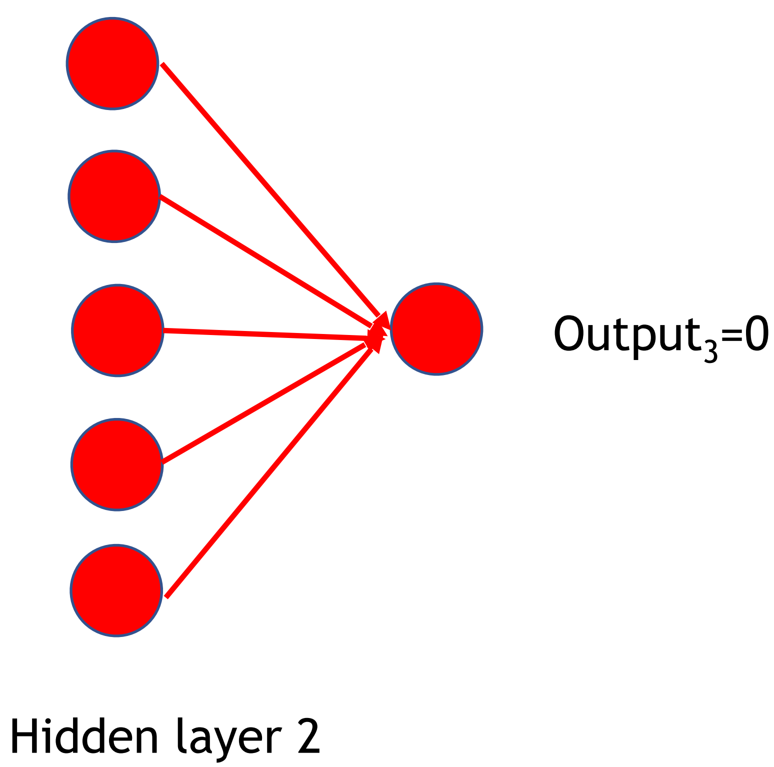
In fact, we know this shape, it looks like regression diagram.

We also know how to obtain a good regression and update weights.

But we know more than this.

We know that the image is in fact 2.

Backpropagation



In fact, we know this shape, it looks like regression diagram.

We also know how to obtain a good regression and update weights.

But we know more than this.

We know that the image is in fact 2.

We know even more than this: this number is not 3.

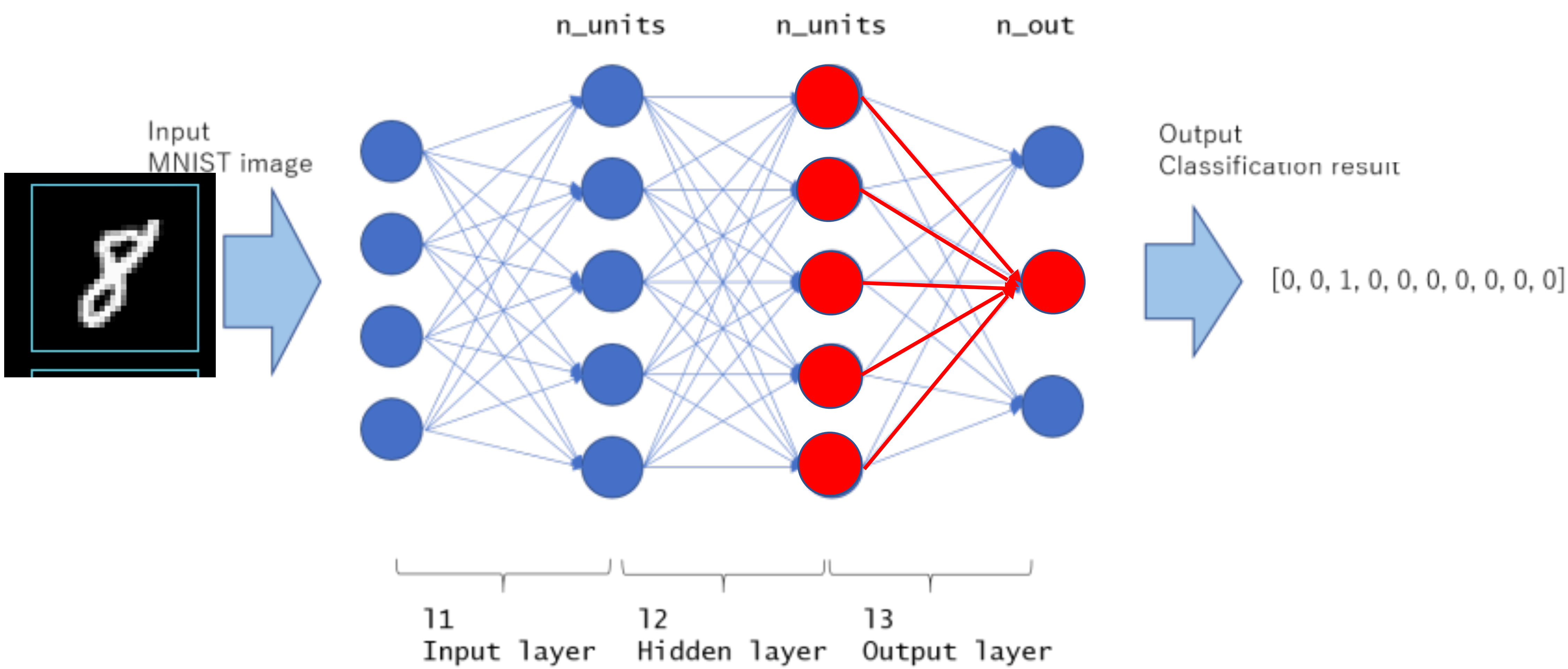
We thus can simultaneously update the weights using a rule:

$$\Delta w = \alpha \frac{\partial \mathscr{L}}{\partial w}$$

α is called a learning rate

Backpropagation

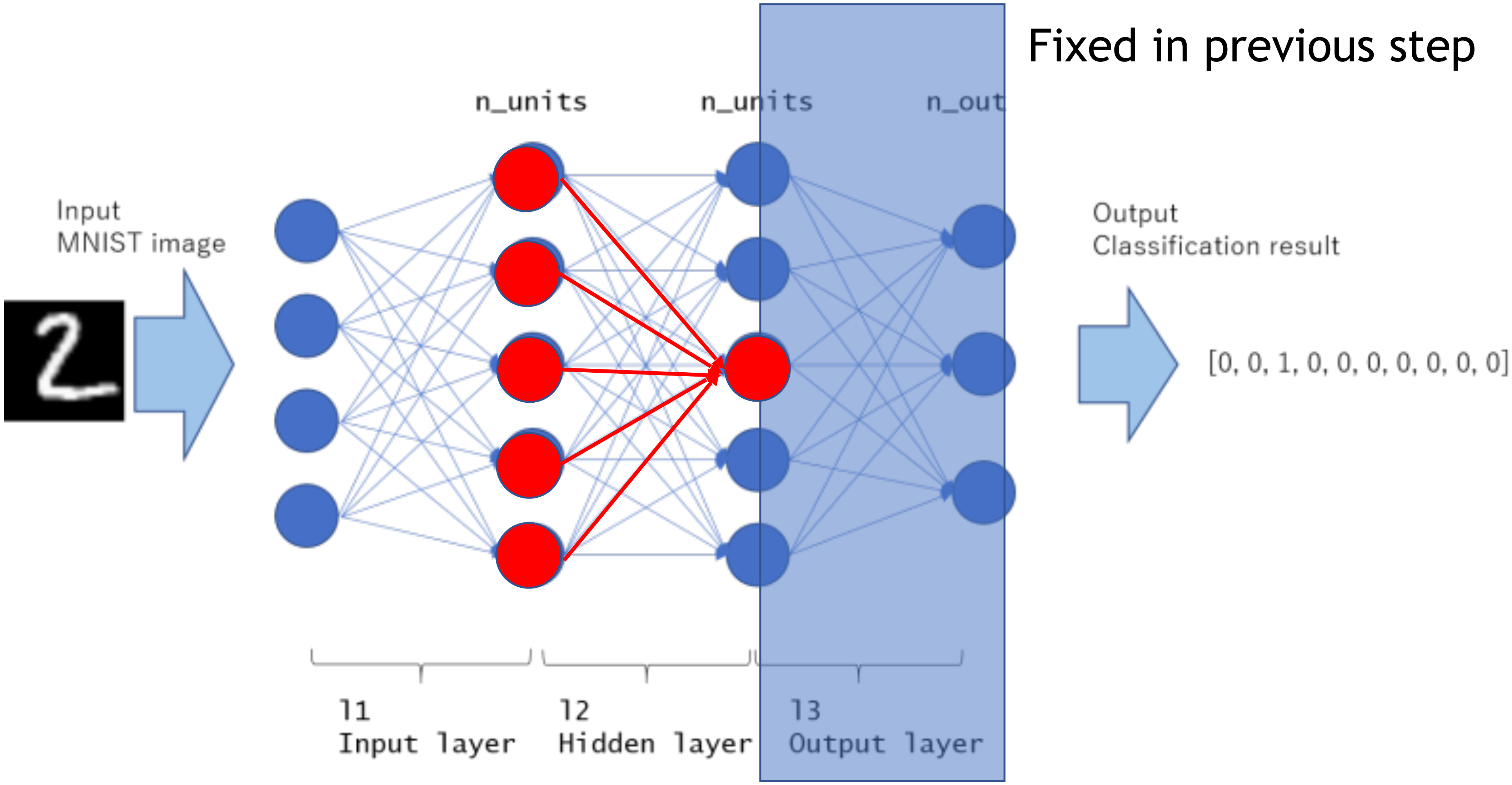
In fact, to avoid our neural network to be trained to give only ”2”, we need to insert several different digits.



We thus produce a new set of weights tor the last hidden layer.

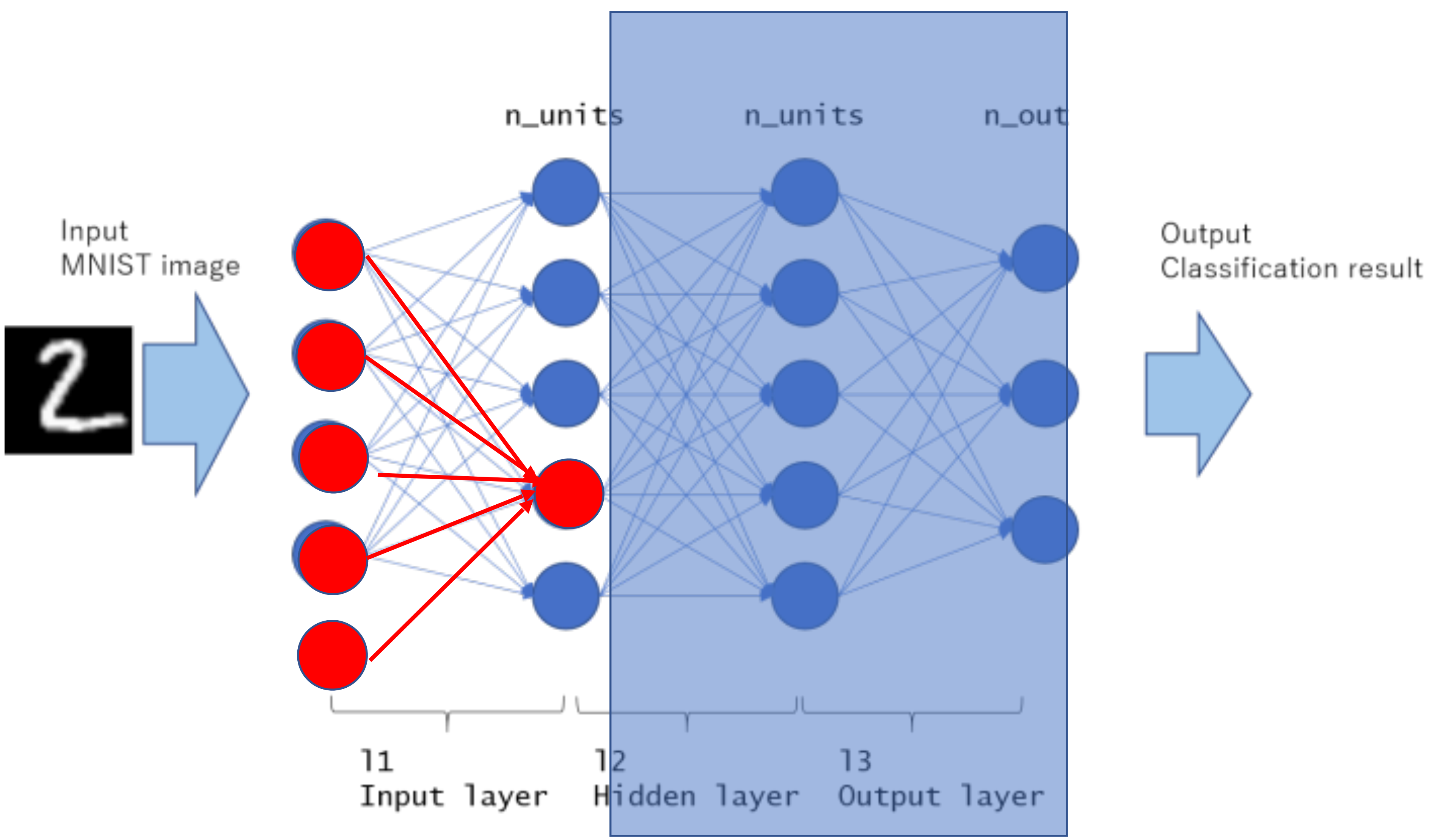
Backpropagation

Now with previous layer, we can update the values in the same manner we did before.



Backpropagation

Fixed in previous steps



Figures of merits

Classification quality evaluation: accuracy

- Given a labeled sample $X^\ell = \big\{(\mathbf{x}_i, y_i)\big\}_{i=1}^{\ell}$, $y_i \in \{-1, +1\}$, and some candidate h, how well does h perform on X^ℓ?

Classification quality evaluation: accuracy

- Given a labeled sample $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$, $y_i \in \{-1, +1\}$, and some candidate h, how well does h perform on X^ℓ?
- Let the thresholded decision rule be $a(x) = [h(x) > t]$ (t: hyperparameter)

Classification quality evaluation: accuracy

- Given a labeled sample $X^\ell = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{\ell}$, $y_i \in \{-1, +1\}$, and some candidate h, how well does h perform on X^ℓ?
- Let the thresholded decision rule be $a(x) = [h(x) > t]$ (t: hyperparameter)
- Obvious choice: accuracy

$$\text{accuracy}(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} [a(\mathbf{x}_i) = y_i]$$

Classification quality evaluation: confusion matrix

	Label $y = 1$	Label $y = -1$
Decision $a(x) = 1$	True Positive (TP)	False Positive (FP)
Decision $a(x) = -1$	False negative (FN)	True Negative (TN)

Classification quality evaluation: confusion matrix

	Label $y = 1$	Label $y = -1$
Decision $a(x) = 1$	True Positive (TP)	False Positive (FP)
Decision $a(x) = -1$	False negative (FN)	True Negative (TN)

- Rates are often more informative:

$$\text{False Positive Rate aka FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

$$\text{True Positive Rate aka TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

Classification quality evaluation: confusion matrix

	Label $y = 1$	Label $y = -1$
Decision $a(x) = 1$	True Positive (TP)	False Positive (FP)
Decision $a(x) = -1$	False negative (FN)	True Negative (TN)

- Rates are often more informative:

$$\text{False Positive Rate aka FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

$$\text{True Positive Rate aka TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

- While accuracy can be expressed, too

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Classification quality: the receiver operating curve

- Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$

Classification quality: the receiver operating curve

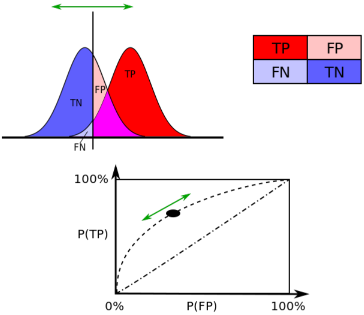
- Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$
- Consider two-dimensional space with coordinates (TPR(t), FPR(t)), corresponding to various choices of the threshold t

Classification quality: the receiver operating curve

- Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$
- Consider two-dimensional space with coordinates (TPR(t), FPR(t)), corresponding to various choices of the threshold t
- The plot TPR(t) vs. FPR(t) is called the receiver operating characteristic (ROC) curve

Classification quality: the receiver operating curve

- Often $h(\mathbf{x})$ is more valuable than its thresholded version $a(x) = [h(x) > t]$
- Consider two-dimensional space with coordinates (TPR(t), FPR(t)), corresponding to various choices of the threshold t
- The plot TPR(t) vs. FPR(t) is called the receiver operating characteristic (ROC) curve
- Area under curve (ROC-AUC) reflects classification quality



Source: Wikipedia

Classification quality: imbalanced data

- TPR(t) vs. FPR(t) / ROC is bad for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ ("treat everything as background") would yield:

Classification quality: imbalanced data

- TPR(t) vs. FPR(t) / ROC is bad for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ ("treat everything as background") would yield:
  - accuracy$(a, X^\ell) = 0.95$ (bad)

Classification quality: imbalanced data

- TPR(t) vs. FPR(t) / ROC is bad for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ ("treat everything as background") would yield:
  - accuracy$(a, X^\ell) = 0.95$ (bad)
  - TPR$(a, X^\ell) = 0.$ (OK)

Classification quality: imbalanced data

- TPR(t) vs. FPR(t) / ROC is bad for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ ("treat everything as background") would yield:
  - accuracy$(a, X^\ell) = 0.95$ (bad)
  - TPR$(a, X^\ell) = 0.$ (OK)
  - FPR$(a, X^\ell) = 0.$ (bad)

Classification quality: imbalanced data

- TPR(t) vs. FPR(t) / ROC is bad for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ ("treat everything as background") would yield:
  - accuracy$(a, X^\ell) = 0.95$ (bad)
  - TPR$(a, X^\ell) = 0.$ (OK)
  - FPR$(a, X^\ell) = 0.$ (bad)
- Criteria better suited for imbalanced problems:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Classification quality: imbalanced data

- The plot recall vs. precision is called the precision-recall (PR) curve

Classification quality: imbalanced data

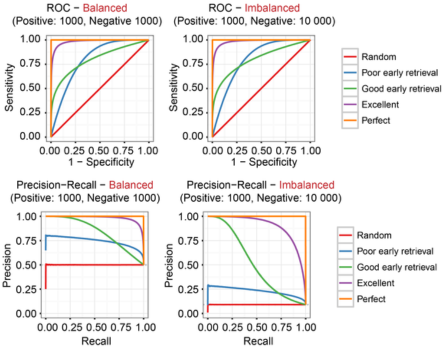
- The plot recall vs. precision is called the precision-recall (PR) curve
- Recall(t) vs. Precision(t) is good for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ would yield:

Classification quality: imbalanced data

- The plot recall vs. precision is called the precision-recall (PR) curve
- Recall(t) vs. Precision(t) is good for imbalanced data: for $\ell = 1000$, $n_- = 950$ (high background noise), $n_+ = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ would yield:
    - Recall$(a, X^\ell) = 0$. (OK)

Classification quality: imbalanced data

- The plot recall vs. precision is called the precision-recall (PR) curve
- Recall(t) vs. Precision(t) is good for imbalanced data: for $\ell = 1000$, $n_{-} = 950$ (high background noise), $n_{+} = 50$ (low signal), a trivial rule $h(\mathbf{x}) = -1$ would yield:
  - Recall$(a, X^{\ell}) = 0$. (OK)
  - Precision$(a, X^{\ell}) = 0$. (OK)



Source: classeval.wordpress.com

Overfitting

Generalization and overfitting

› Training set memorization: for seen $(\mathbf{x}, y) \in X^{\ell}$, $h(\mathbf{x}) = y$

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances
- How to assess model's generalization ability?

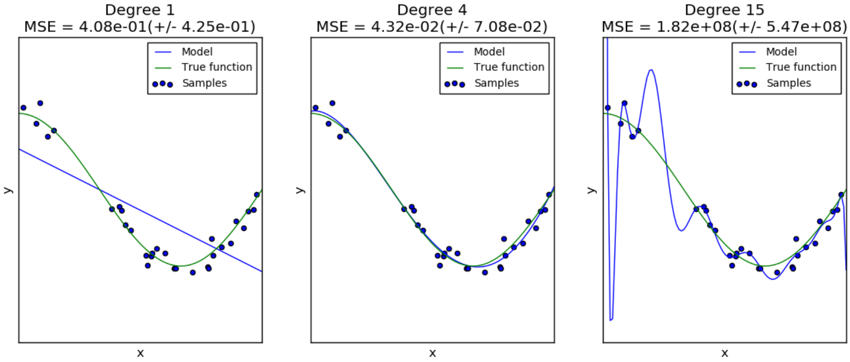
Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances
- How to assess model's generalization ability?
- Consider an example:
  - $y = \cos(1.5\pi x) + \mathcal{N}(0, 0.01)$, $x \sim \text{Uniform}[0, 1]$
  - Features: $\{x\}$, $\{x, x^2, x^3, x^4\}$, $\{x, \ldots, x^{15}\}$
  - The model is linear w. r. t. features: $f(\mathbf{x}) = \mathbf{w}^\intercal \phi(\mathbf{x})$

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances
- How to assess model's generalization ability?
- Consider an example:
  - $y = \cos(1.5\pi x) + \mathcal{N}(0, 0.01)$, $x \sim \mathrm{Uniform}[0, 1]$
  - Features: $\{x\}$, $\{x, x^2, x^3, x^4\}$, $\{x, \ldots, x^{15}\}$
  - The model is linear w. r. t. features: $f(\mathbf{x}) = \mathbf{w}^\intercal \phi(\mathbf{x})$
- How well do the regression models perform?

Polynomial fits of different degrees



Model validation and selection

- We have free parameters in models:

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?
- Can we use entire dataset to fit the model?

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?
- Can we use entire dataset to fit the model?
- Yes, but we will likely get overly optimistic performance estimate

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?
- Can we use entire dataset to fit the model?
- Yes, but we will likely get overly optimistic performance estimate
- The solution: rely on held-out data to assess model performance

Train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

Train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}

Train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}

Train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\mathsf{TRAIN}} \cup X^\ell_{\mathsf{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}
- Assess quality using $Q(h, X^\ell_{\mathsf{VAL}})$

Train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

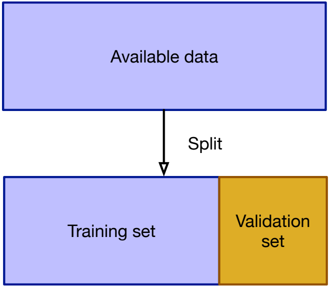
- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}
- Assess quality using $Q(h, X^\ell_{\text{VAL}})$
- Data-hungry: can we afford the ”luxury” of setting aside a portion of the data for testing?

Train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}
- Assess quality using $Q(h, X^\ell_{\text{VAL}})$
- Data-hungry: can we afford the ”luxury” of setting aside a portion of the data for testing?
- May be imprecise: the holdout estimate of error rate will be misleading if we happen to get an ”unfortunate” split



Assessing generalization ability: cross-validation

- Split training set into subsets of equal size $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$

Assessing generalization ability: cross-validation

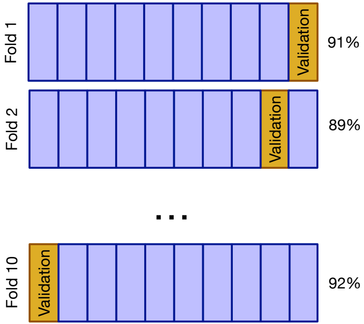
- Split training set into subsets of equal size $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$
- Train K models $h_1, \ldots, h_K$ where each model h_k is trained on all subsets but X_k^ℓ

Assessing generalization ability: cross-validation

- Split training set into subsets of equal size $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$
- Train K models $h_1, \ldots, h_K$ where each model h_k is trained on all subsets but X_k^ℓ
- Assess quality using $\mathrm{CV} = \frac{1}{K}\sum_{k=1}^{K} Q(h_k, X_k^\ell)$ (K-fold)

Assessing generalization ability: cross-validation

- Split training set into subsets of equal size $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$
- Train K models $h_1, \ldots, h_K$ where each model h_k is trained on all subsets but X_k^ℓ
- Assess quality using
  $$\mathrm{CV} = \frac{1}{K} \sum_{k=1}^{K} Q(h_k, X_k^\ell) \text{ (K-fold)}$$
- Leave-one-out cross-validation: $X_k^\ell = \{(\mathbf{x}_k, y_k)\}$ (yes, train ℓ models!)



Cross-validation method: drawbacks

$$\mathrm{CV} = \frac{1}{K}\sum_{k=1}^{K} Q(h_k, X_k^\ell)$$

Many folds:

- Small bias: the estimator will be very accurate
- Large variance: due to small split sizes
- Costly: many experiments, large computational time

Cross-validation method: drawbacks

$$\mathrm{CV} = \frac{1}{K}\sum_{k=1}^{K} Q(h_k, X_k^\ell)$$

Many folds:

- Small bias: the estimator will be very accurate
- Large variance: due to small split sizes
- Costly: many experiments, large computational time

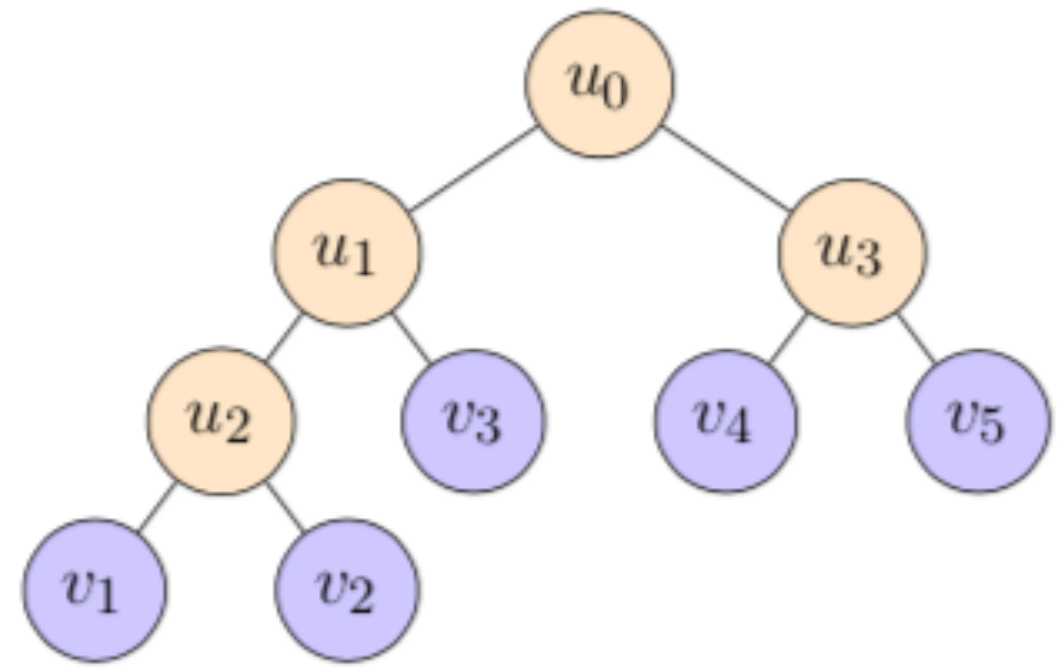
Few folds:

- Cheap, computationally effective: few experiments
- Small variance: average over many samples
- Large bias: estimated error rate conservative or smaller than the true error rate

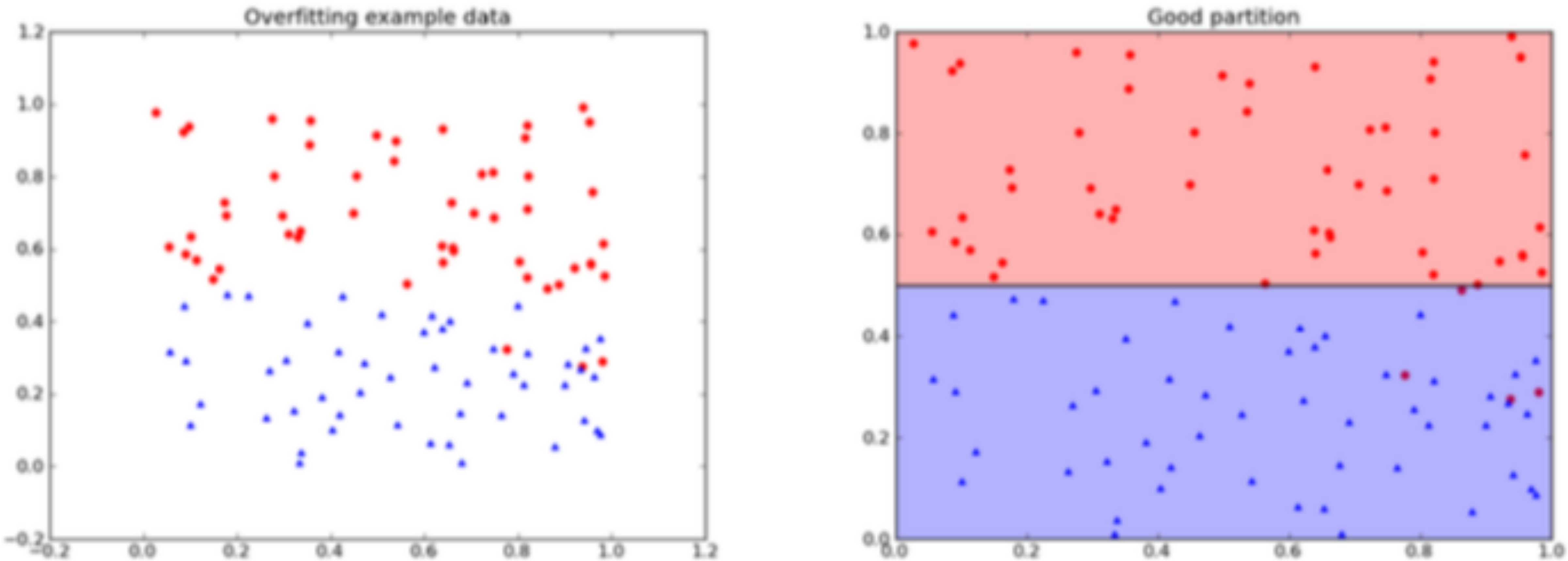
Decision trees

Decision tree formalism

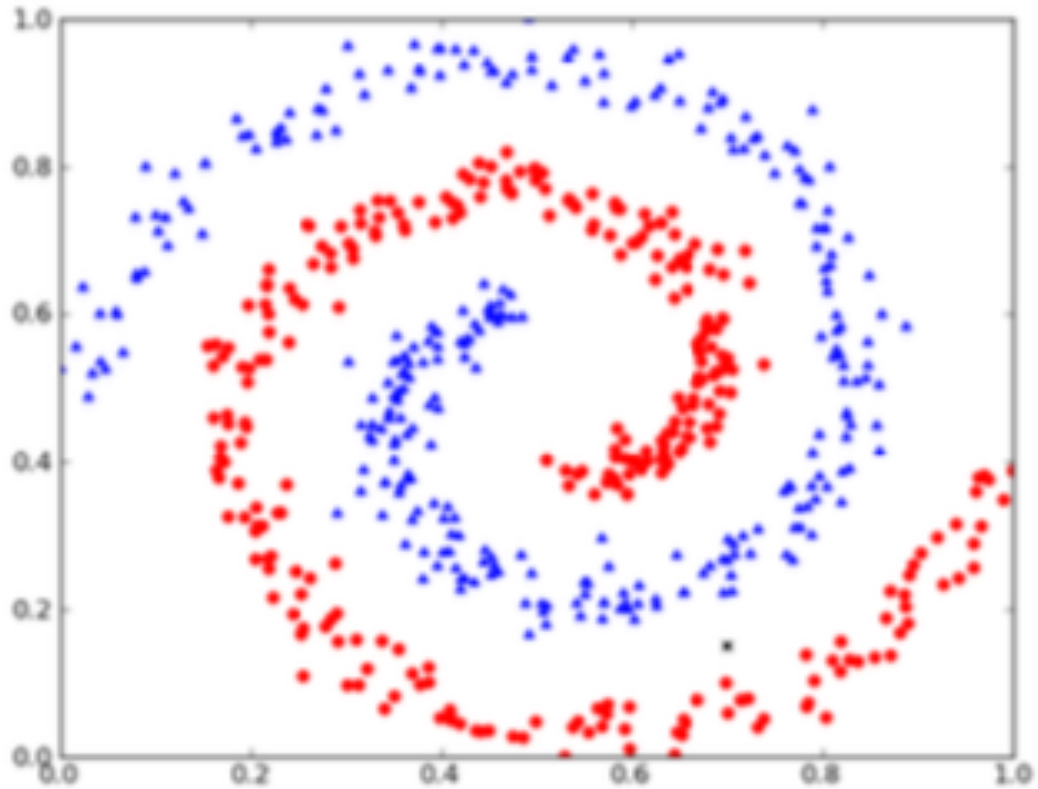
- Decision tree is a binary tree V
- Internal nodes $u \in V$: predicates $\beta_u : \mathbb{X} \to \{0, 1\}$
- Leafs $v \in V$: predictions x
- Algorithm $h(\mathbf{x})$ starts at $u = u_0$
  - Compute $b = \beta_u(\mathbf{x})$
  - If $b = 0$, $u \leftarrow \mathrm{LeftChild}(u)$
  - If $b = 1$, $u \leftarrow \mathrm{RightChild}(u)$
  - If u is a leaf, return b
- In practice: $\beta_u(\mathbf{x}; j, t) = [\mathbf{x}_j < t]$

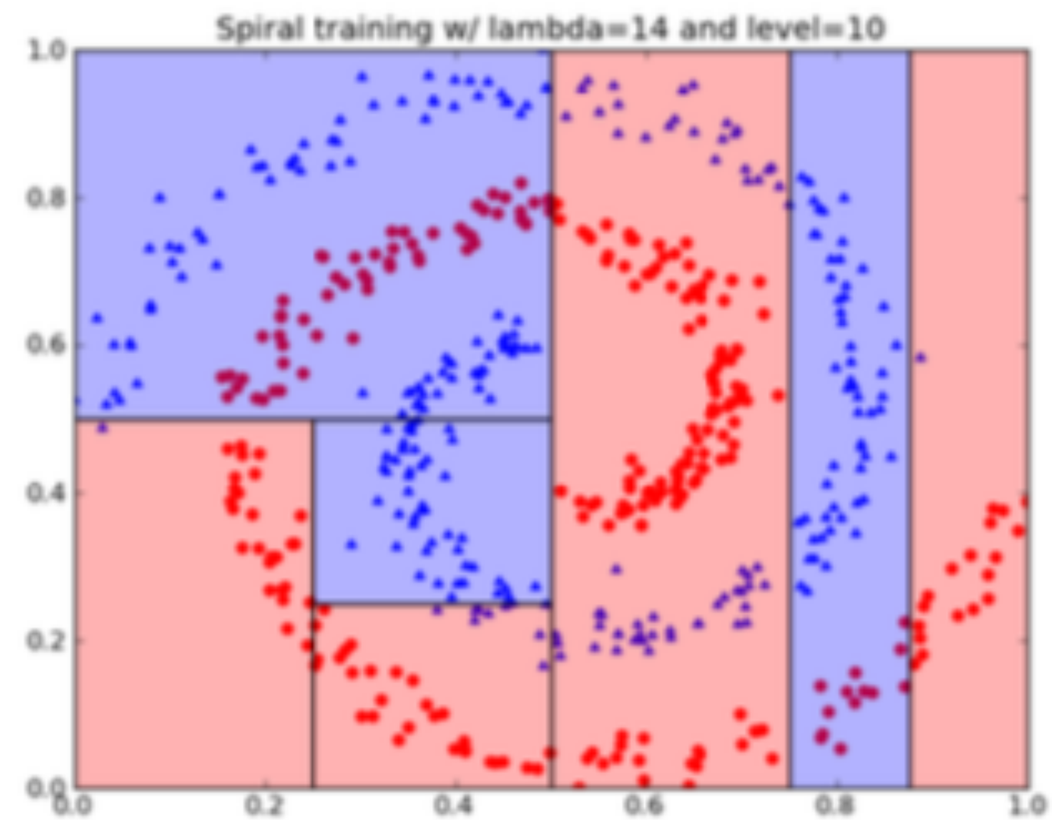


Greedy tree learning for binary classification



Greedy tree learning for binary classification





Ensembling

One could organize the trees into ”collection”.
Ensembles:

- Single Decision Tree.
- Random Forest: mean of N decision trees predictions.
- AdaBoost: set of N trees. A new tree is trained on mistakes of previous built trees. Prediction is weighted mean of predictions of the single trees.
- Gradient Boosting: set of N trees. Prediction is weighted mean of predictions of the single trees. Weights are selected to minimize the loss function.

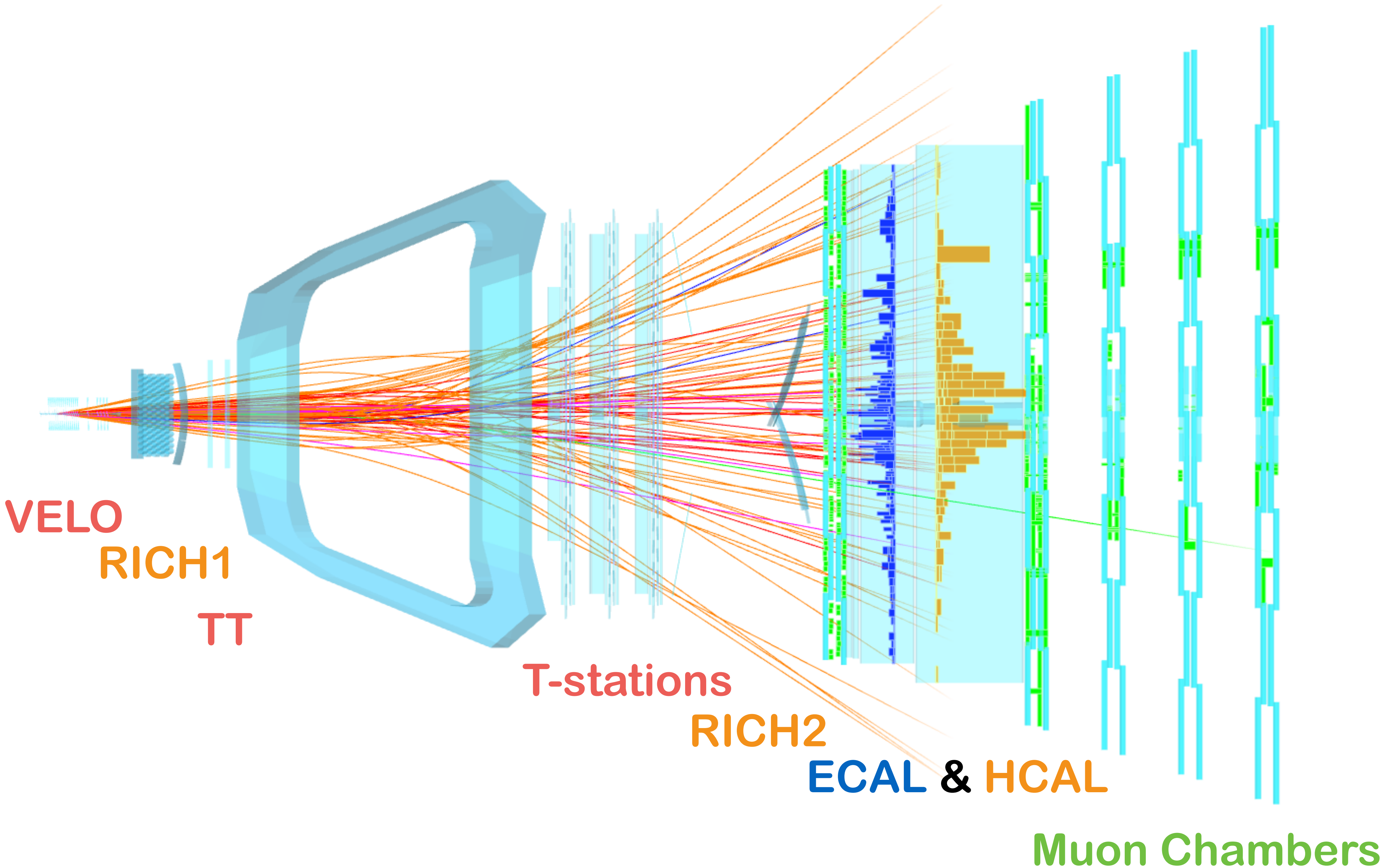
These algorithms are easy to train and provide good predictive power.

Summary so far

- We covered only ”supervised” machine learning: regression and classification. This type of learning needs labeled datasets (which we normally have from simulation).
- There is also a big part, which will not be covered here: ”unsupervised” learning (clustering, some anomaly detection) and ”reinforcement” learning (agent behaviour in medium).

Classification in High-Energy Physics

LHCb layout



PID at LHCb

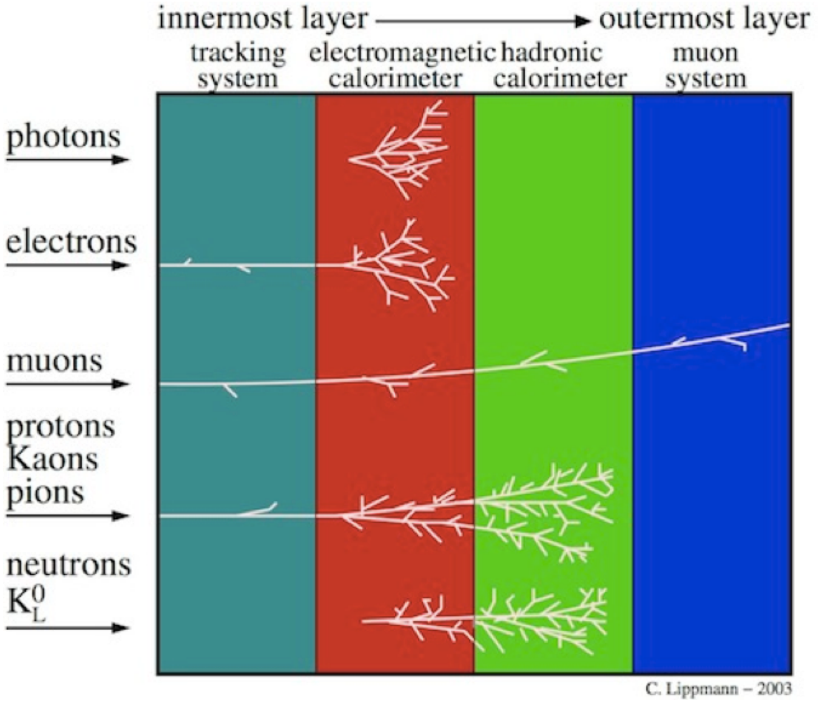
Problem: identify particle type associated with a track/energy deposited in the subdetectors

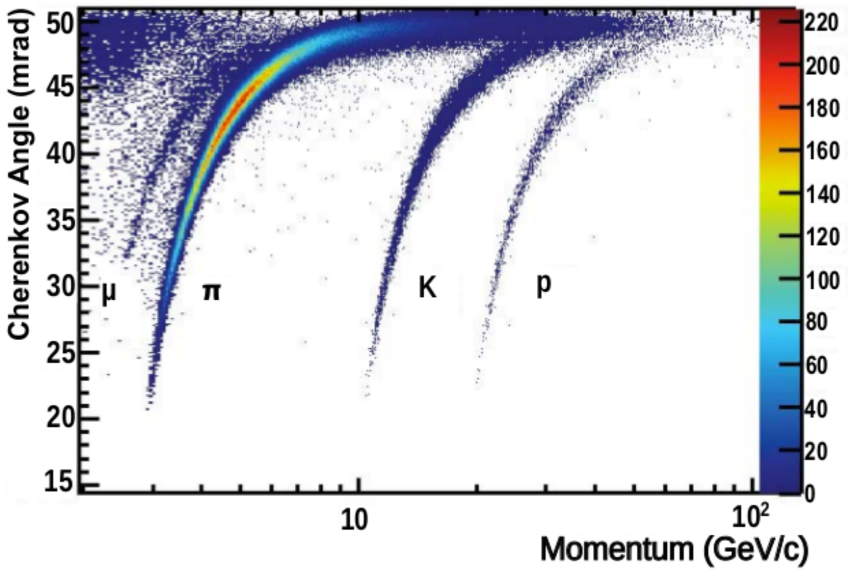
- Charged: π, e, μ, K, p
- Neutral: π^0, γ, n

Better PID performance → better bkg rejection → more precise results.

PID also used for trigger (in particular for upgrade): less background → less resources (less bandwidth)

High-level info from subdetectors + track quality info → multi-class classification in machine learning



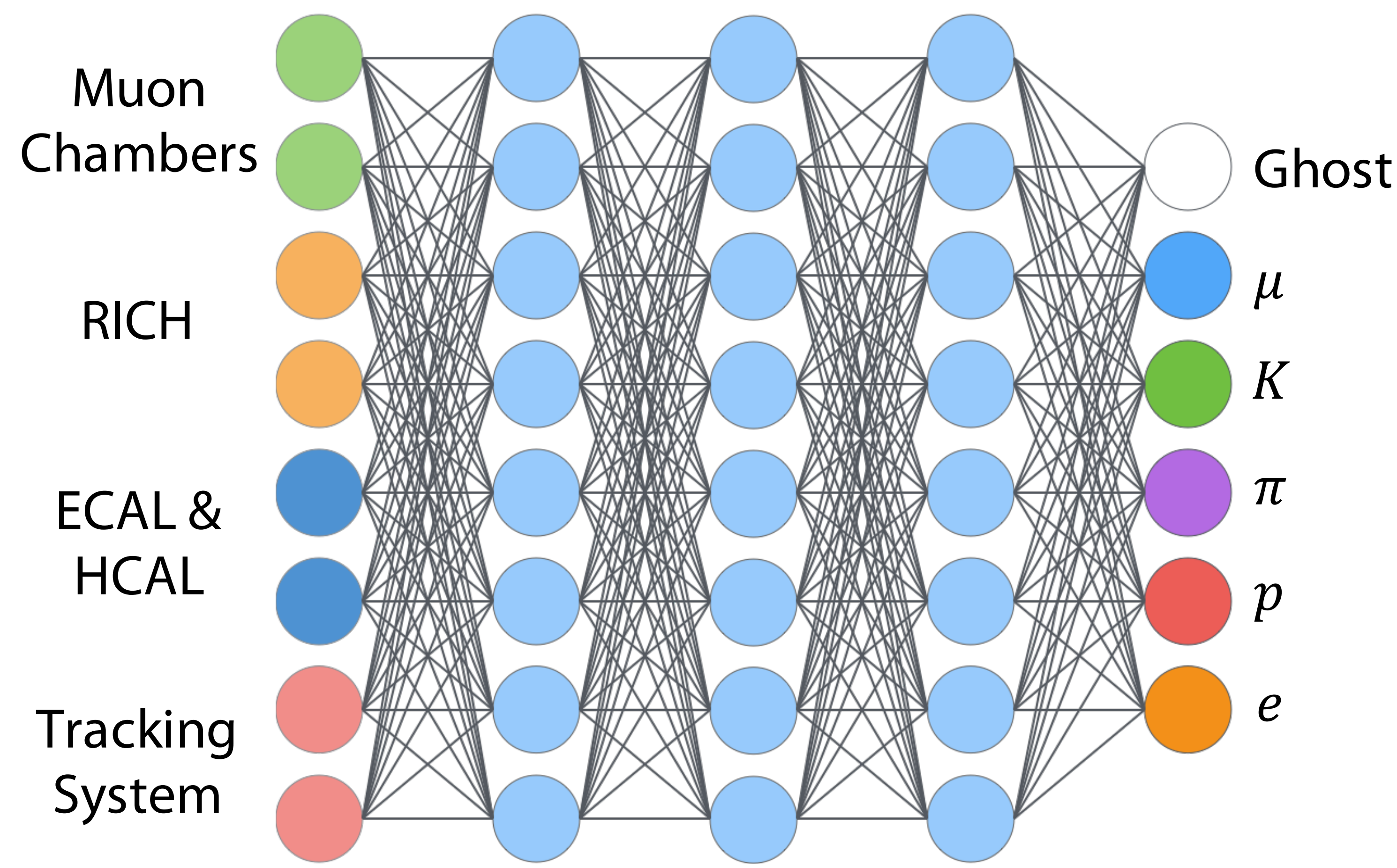


Global Particle Identification

Problem: identify particle type associated with a track.

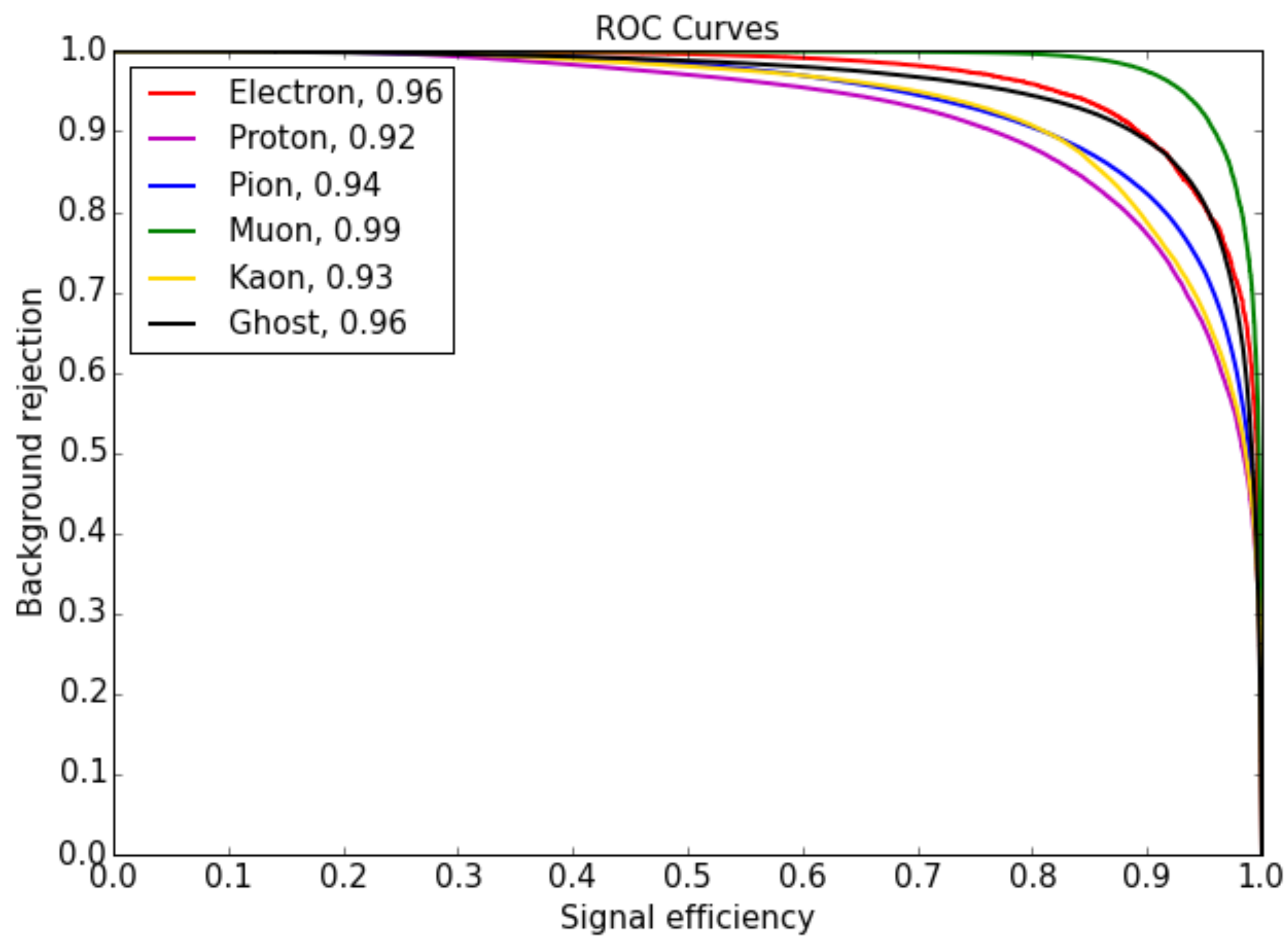
Particle types: Electron, Muon, Pion, Kaon, Proton and Ghost

Input observables: particle responses in RICH, ECAL, HCAL subdetectors, Muon Chambers and Track observables.



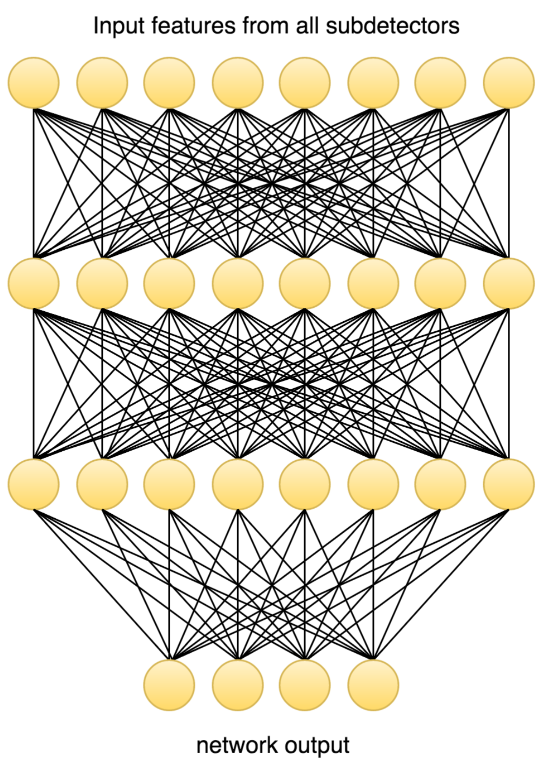
Quality Metrics

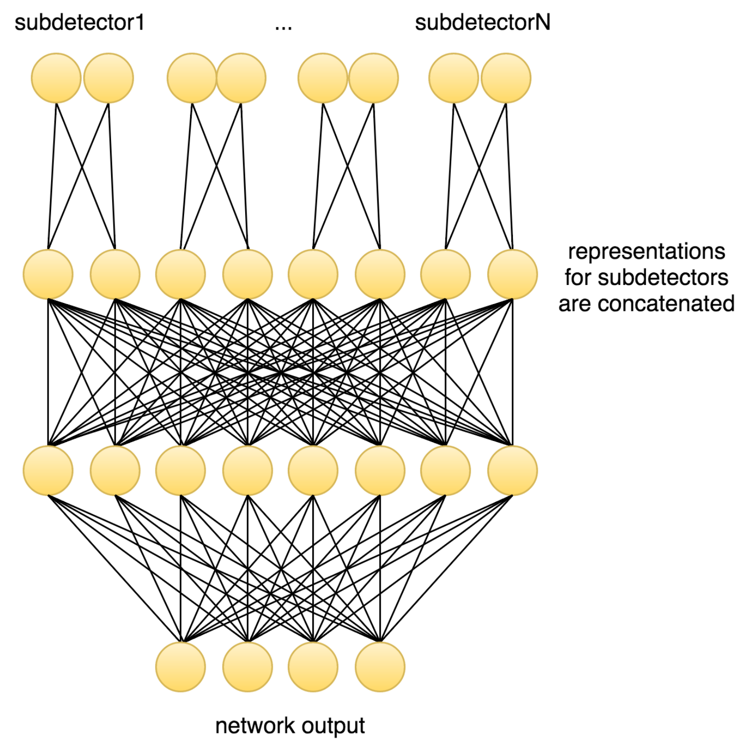
- One-vs-rest ROC curves used to measure models quality.
- Area under them (ROC AUC) are used as target metrics to select the best models.

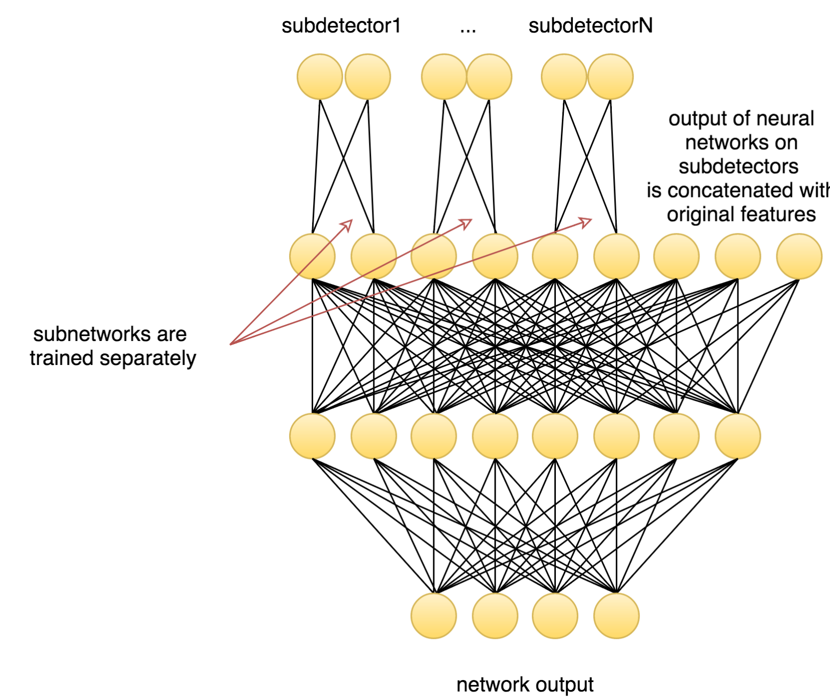


Technologies

- Several possibilities were tested, all of them were inspired by the knowledge of detector responses.



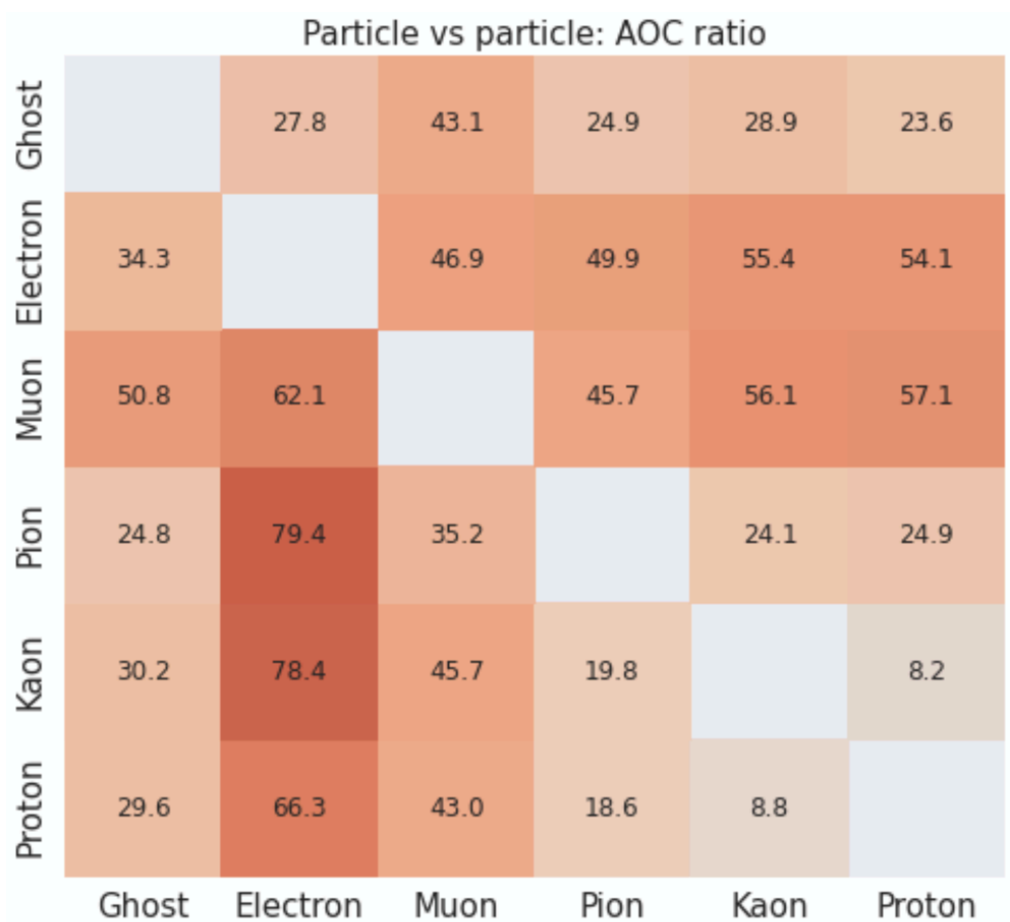


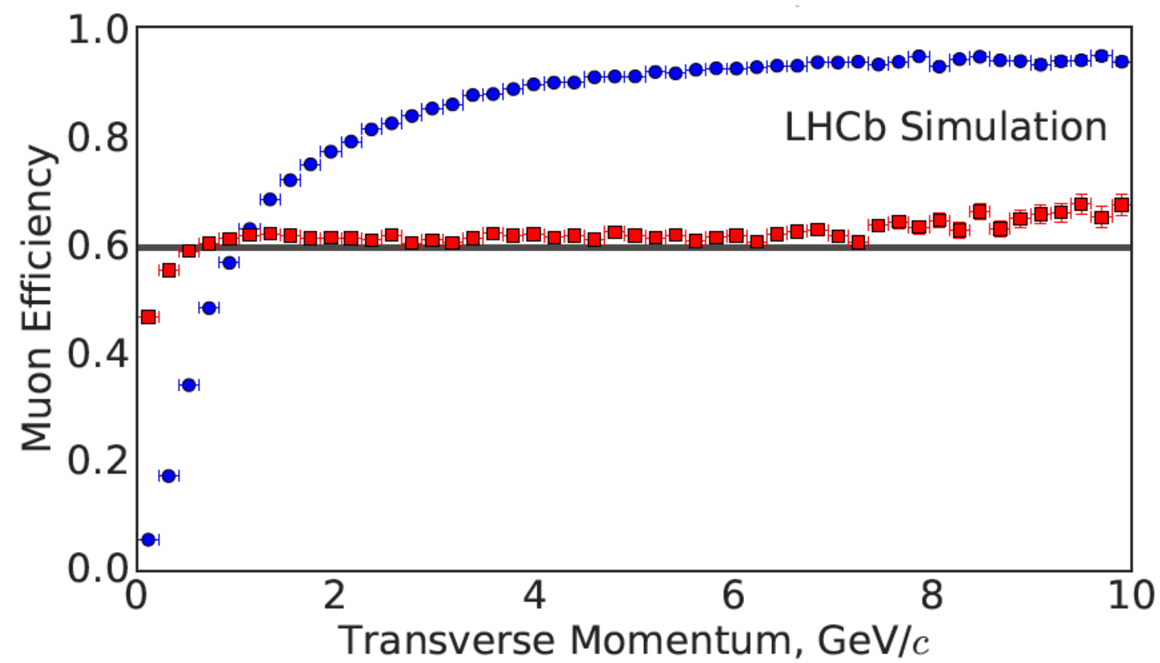


- Other approaches using Decision trees were also tested and brought competitive results.

Results

〉 Using the above mentioned approaches we were able to decrease the error rate by up to 80%.





〉 In addition to this, we were able to correct the detector acceptance function, which lead to a lower systematics.

Journal of Physics: Conference Series. 2018. Vol. 1085. No. 4. P. 1-5.

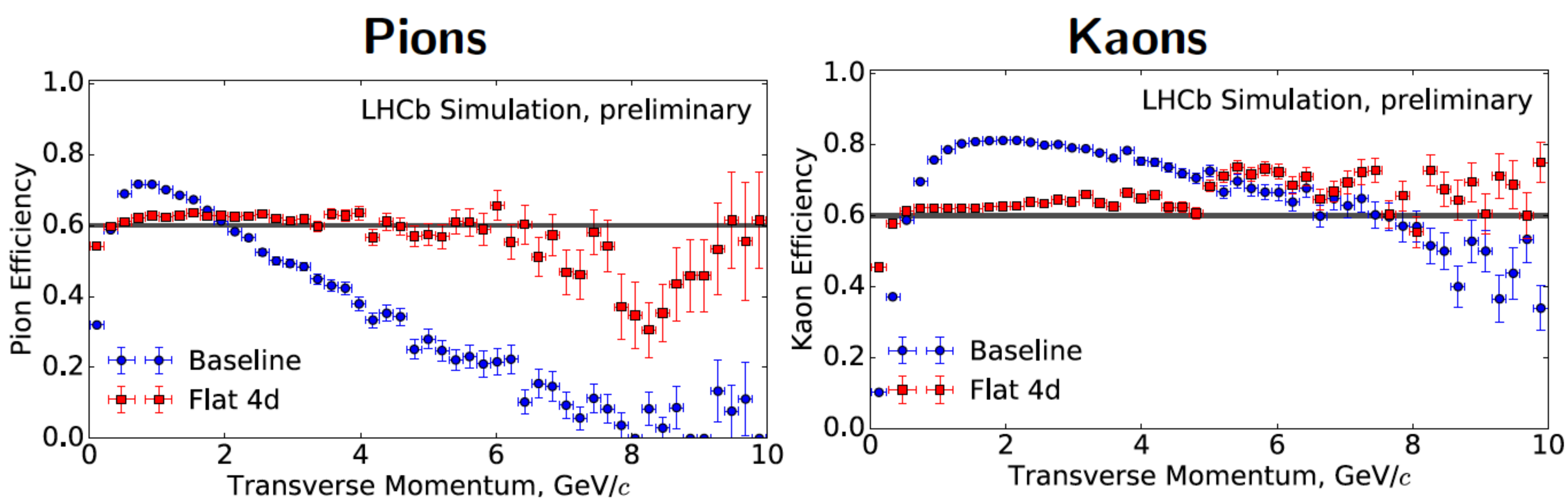
http://iopscience.iop.org/article/10.1088/1742-6596/1085/4/042038/meta

Flat efficiency approach

- PID performace depends on **particle kinematics** (p, p_T, η) and $\mathbf{N_{tracks}}$
- Flat PID efficiencies:
  - ★ Good discrimination for different analyses
  - ★ Unbiased background discrimination
  - ★ Reduced systematic uncertainties

Introduce flatness term in loss function: $\mathcal{L} = \mathcal{L}_{AdaLoss} + \alpha \mathcal{L}_{Flat}$

- **Flat4d:** $\mathcal{L}_{Flat_{4d}} = \mathcal{L}_{Flat\_P} + \mathcal{L}_{Flat\_PT} + \mathcal{L}_{Flat\_nTracks} + \mathcal{L}_{Flat\_\eta}$

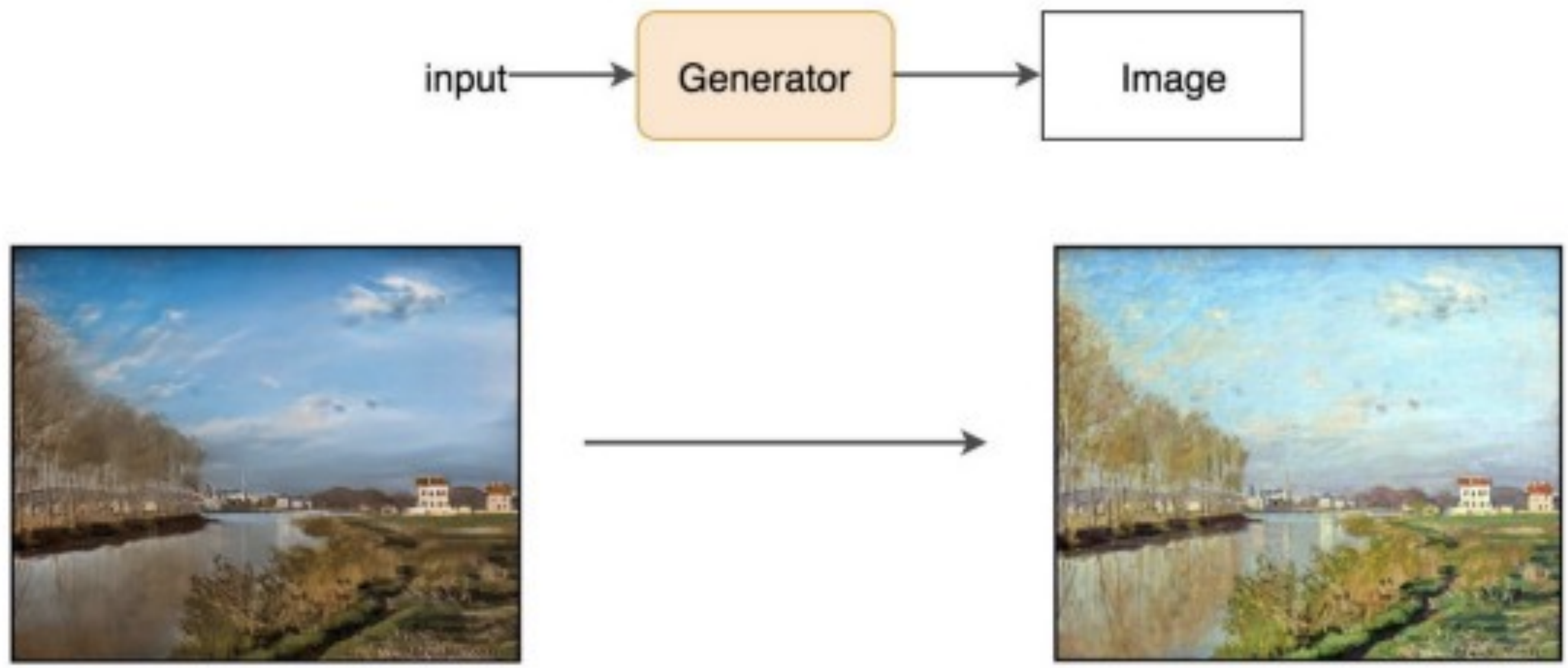


Flat4d, ProbNN

→ Better PID efficiency flatness in p, p_T, η, N_{tracks} than baseline

Generative adversarial Networks

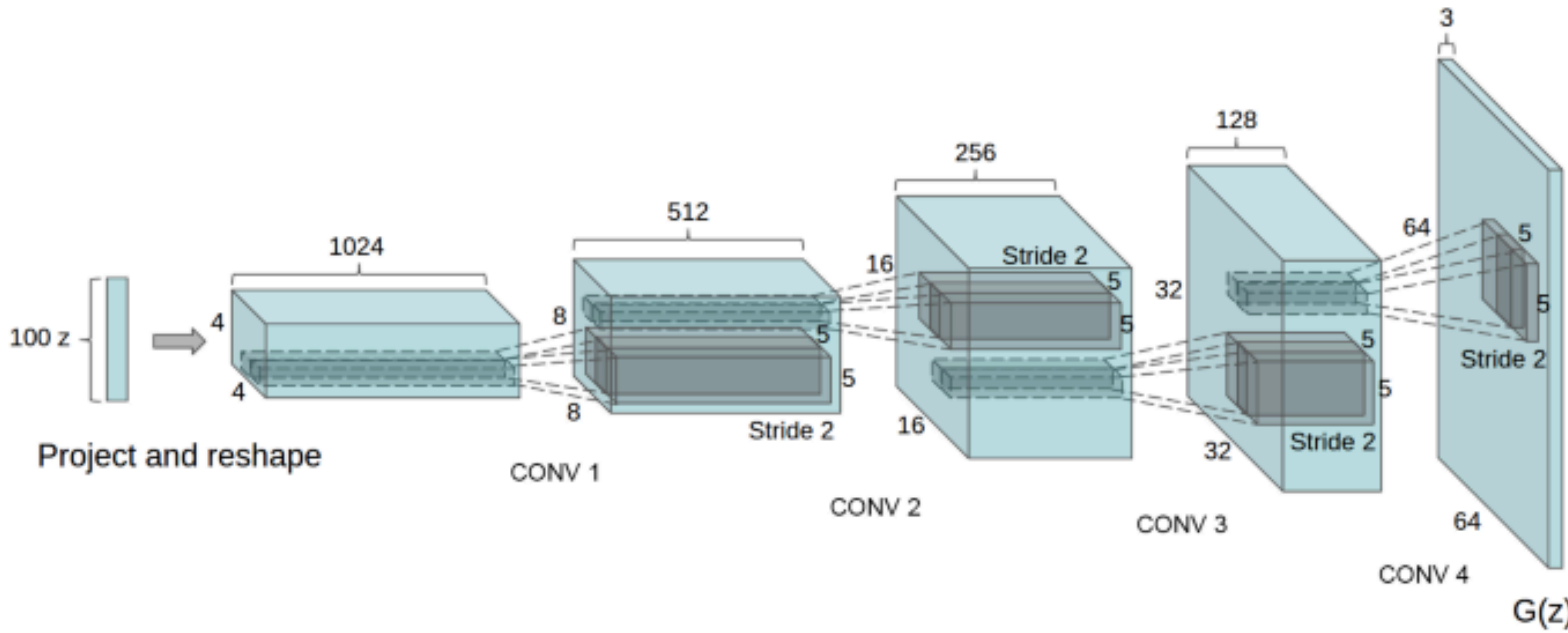
Generative adversarial networks



We want a realistic generation of the images with good randomisation.

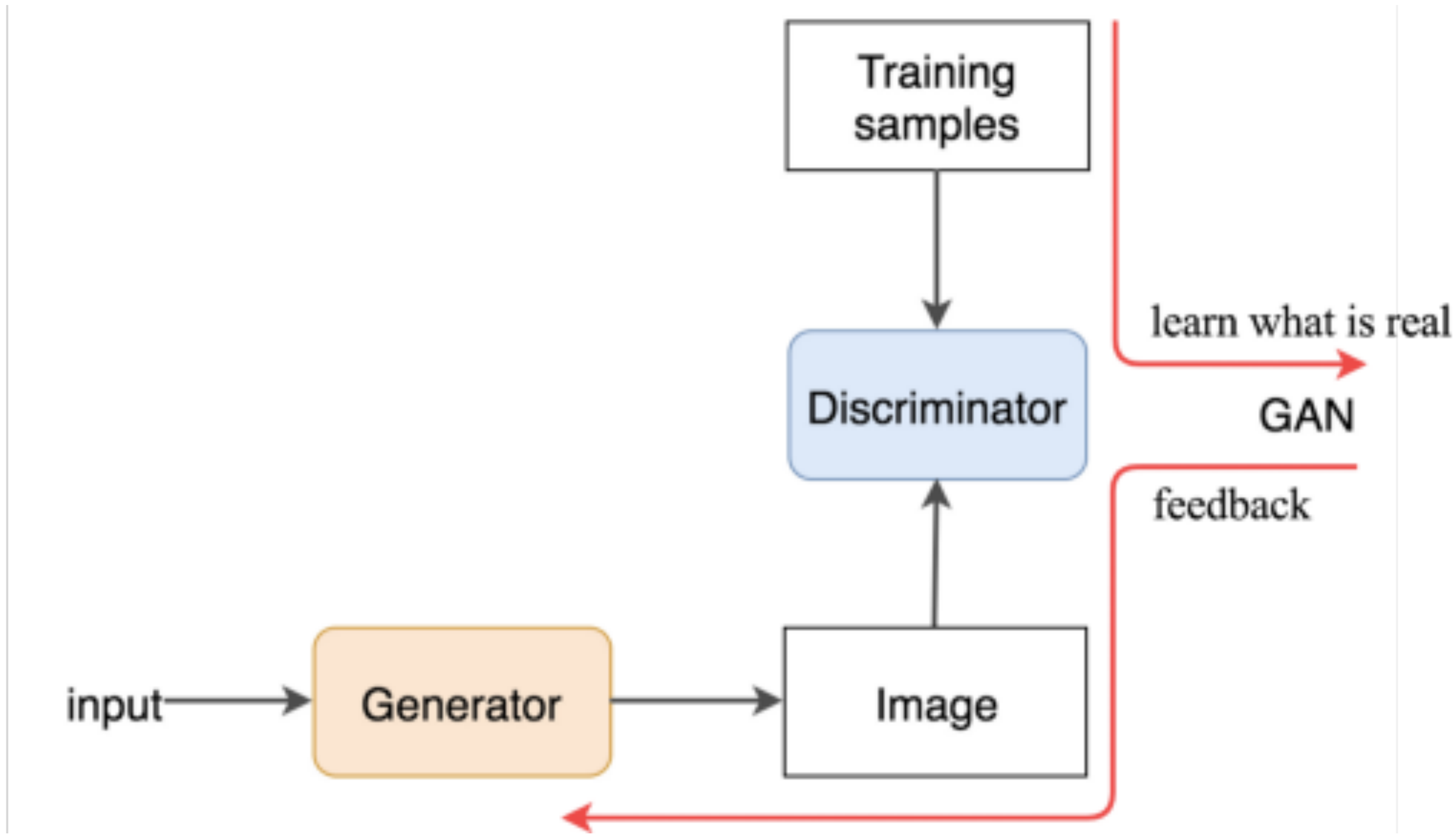
https://medium.com/@jonathan_hui

Generative adversarial networks



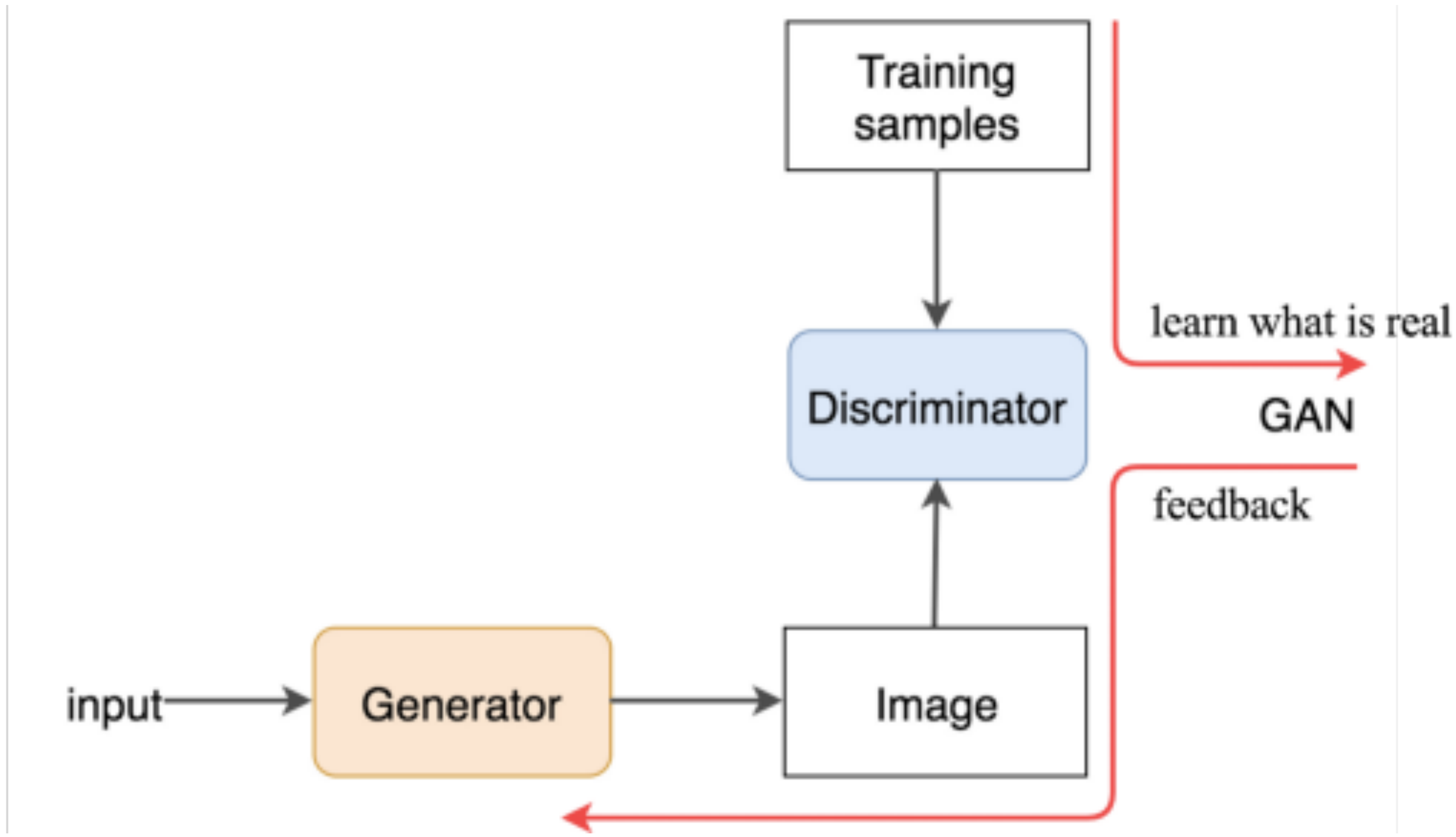
We can construct a network that has ever increasing number of elements in layers. Thus, we will be able to generate something out of random noise. How we can make it more realistic?

Generative adversarial networks



We introduce discriminator! Another neural network that can can check that the image we produce looks real.

Generative adversarial networks



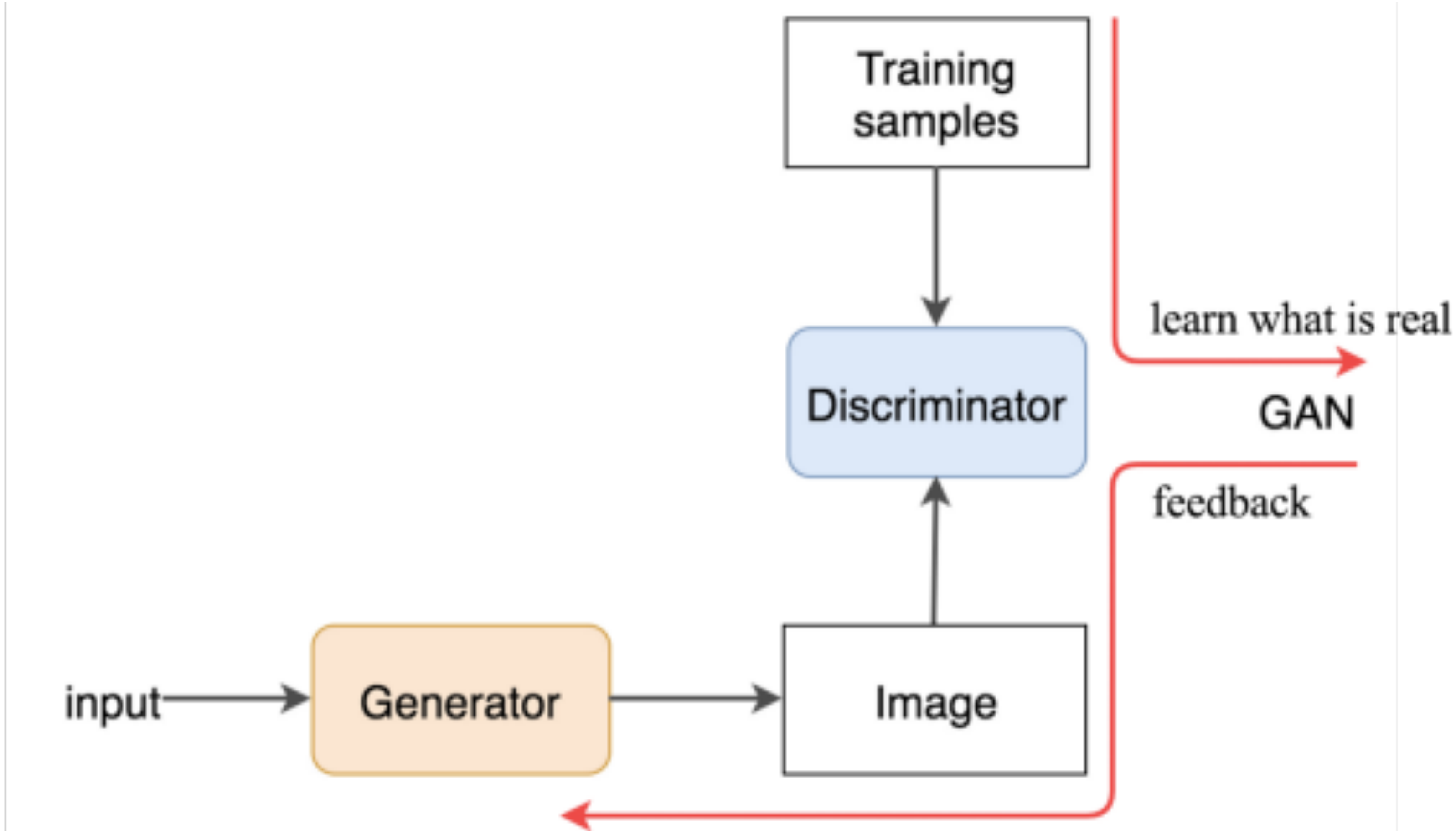
For discriminator, we use a typical objective to discriminate between figures

$$\max_D V(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

recognize real images better

recognize generated images better

Generative adversarial networks

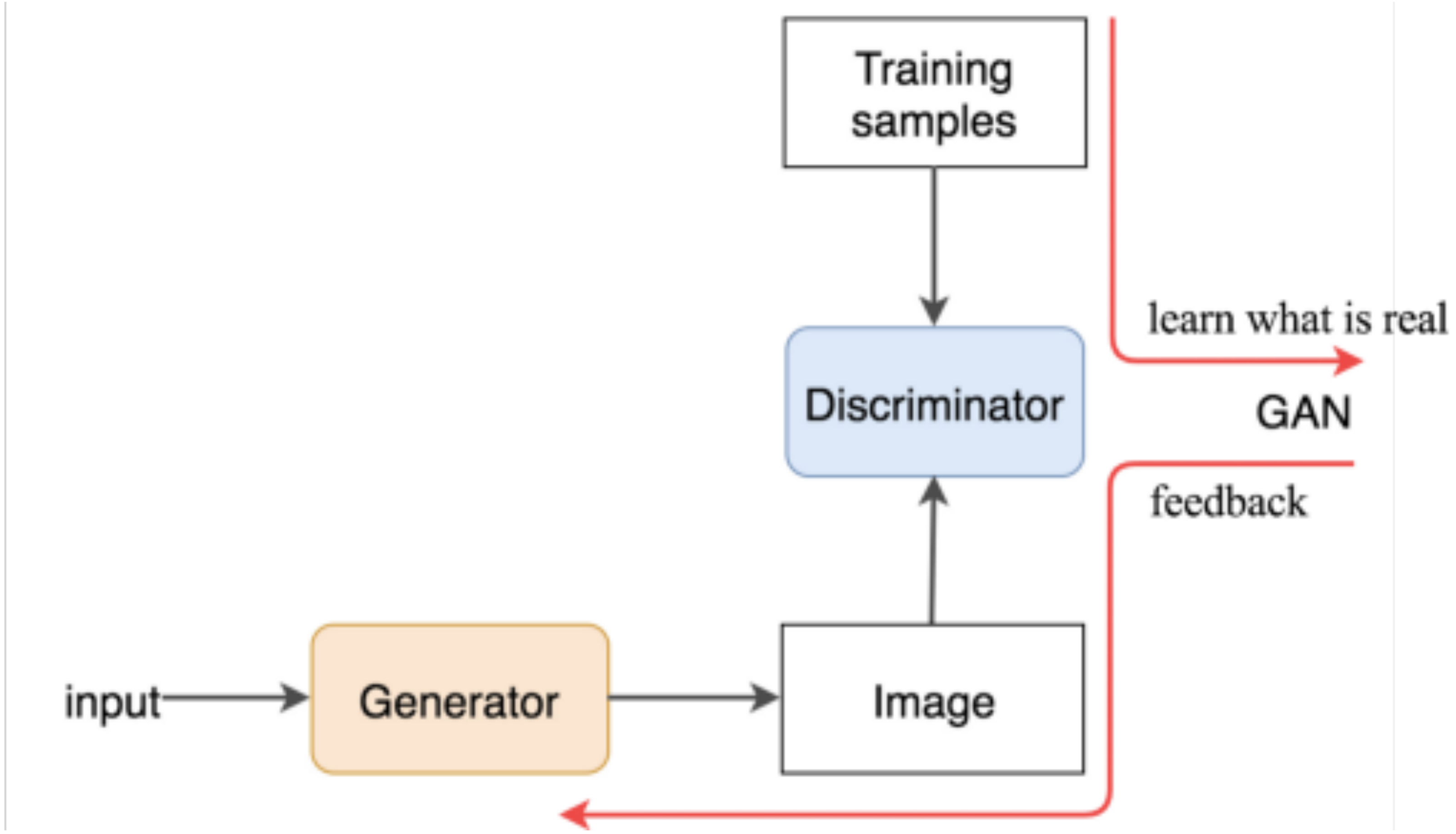


For generator, we ask to make generation as real as possible:

$$\min_{G} V(G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Optimize G that can fool the discriminator the most.

Generative adversarial networks



And we can rewrite the objective into a single line:

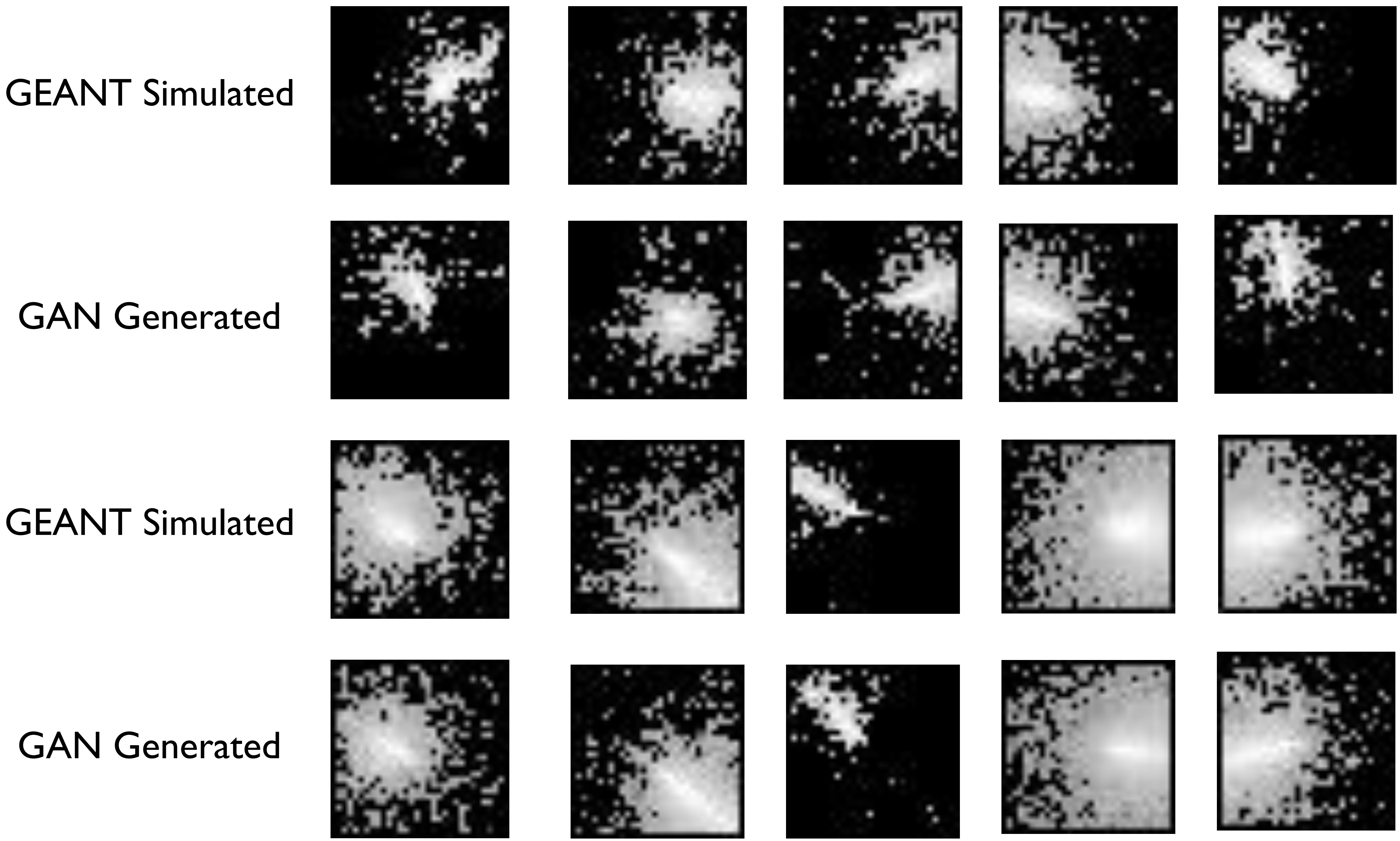
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Generative adversarial networks

Using this technique, we can generate «realistic» cats. What else?


Collision Event at 7 TeV
ATLAS EXPERIMENT
2010-03-30, 12:58 CEST
Run 152166, Event 316199
http://atlas.web.cern.ch/Atlas/public/EVTDISPLAY/events.html

Realistic responses of detector?



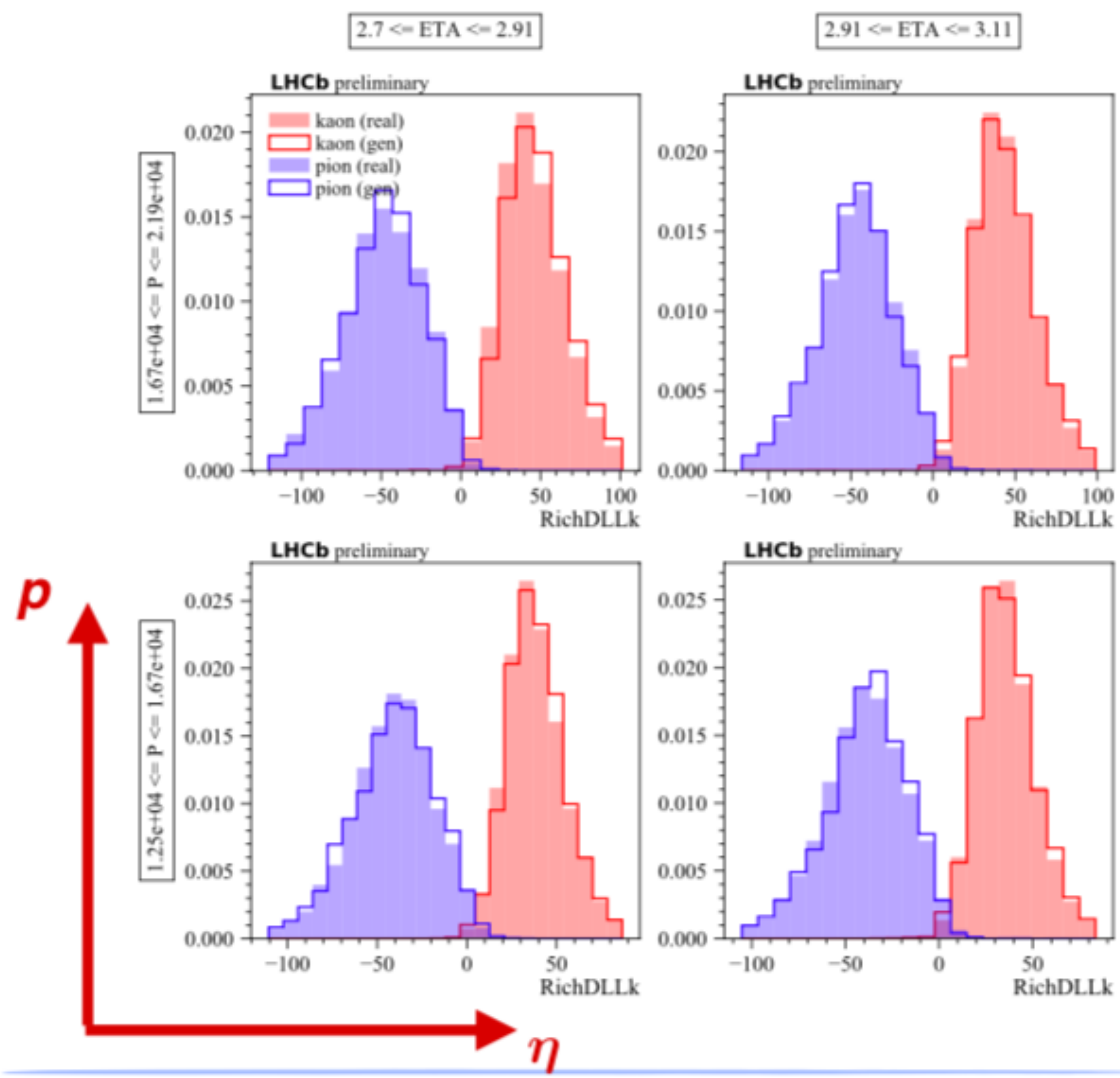
http://arxiv.org/abs/arXiv:1812.01319

Realistic responses of detector?

In fact, we do not care about the image - we need better description of the reconstructed observables.

For example, prediction how particle identification of Cherenkov detectors behave.

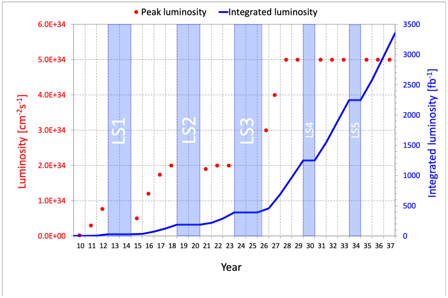
Currently, the fast simulation of this kind takes 10s of milliseconds, while full simulation is around 10s of seconds.



https://doi.org/10.1016/j.nima.2019.01.031

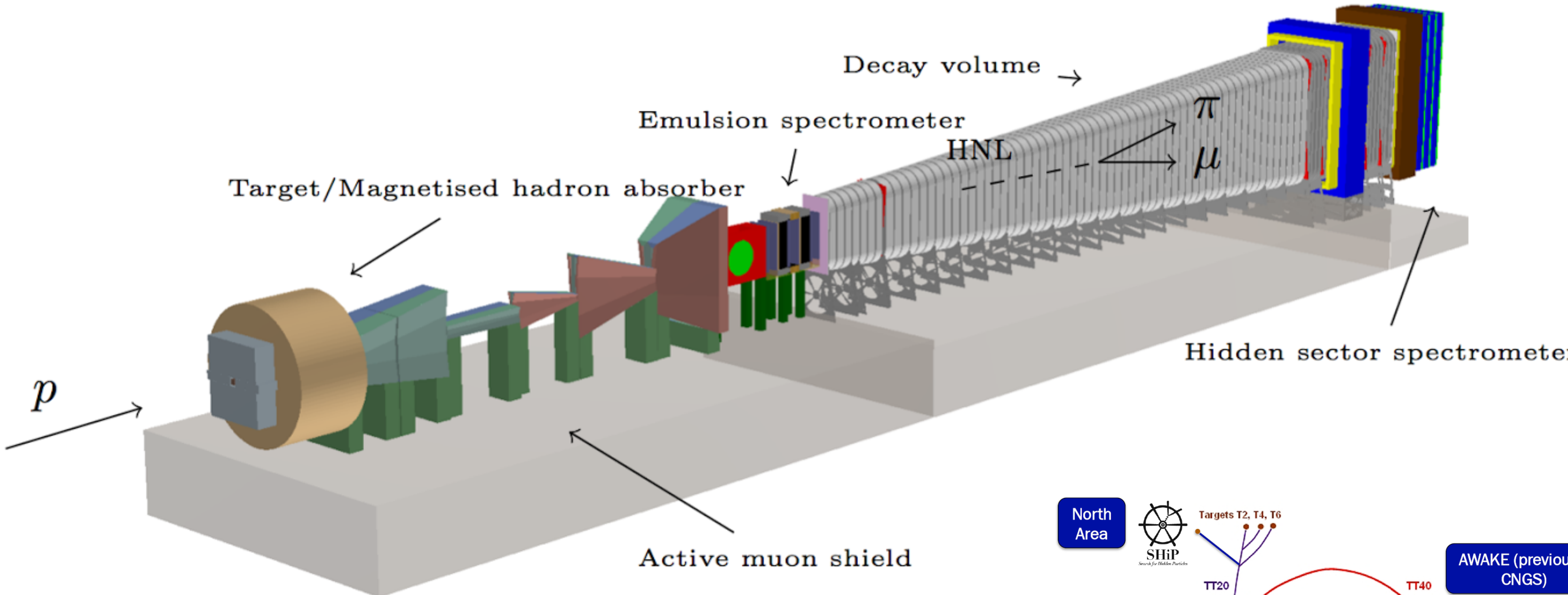
Need for statistics

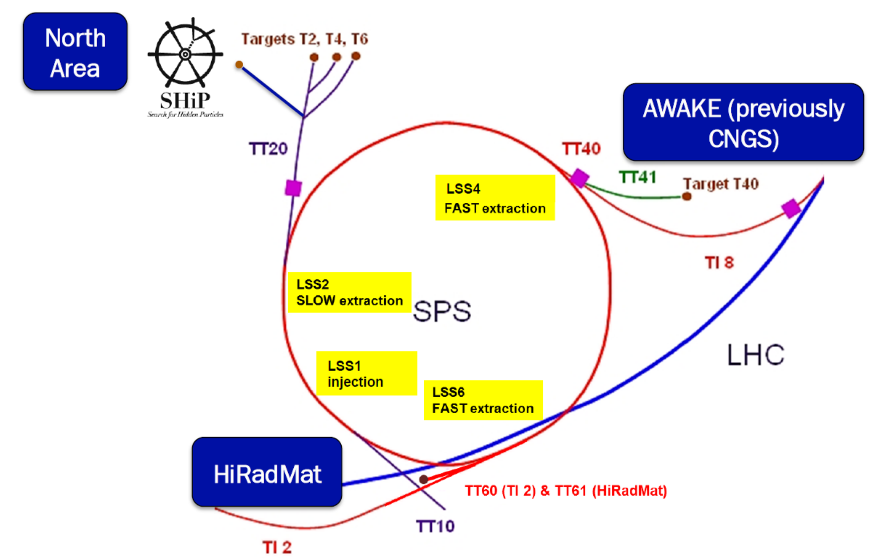
- New experiments and upgrades require a lot of simulation.
- Full simulation of LHC event can take up to several minutes.
- We need to simulate billions of events.



Bayesian optimization

SHiP Experiment

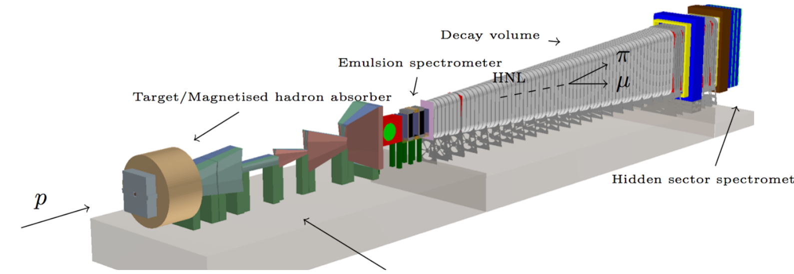


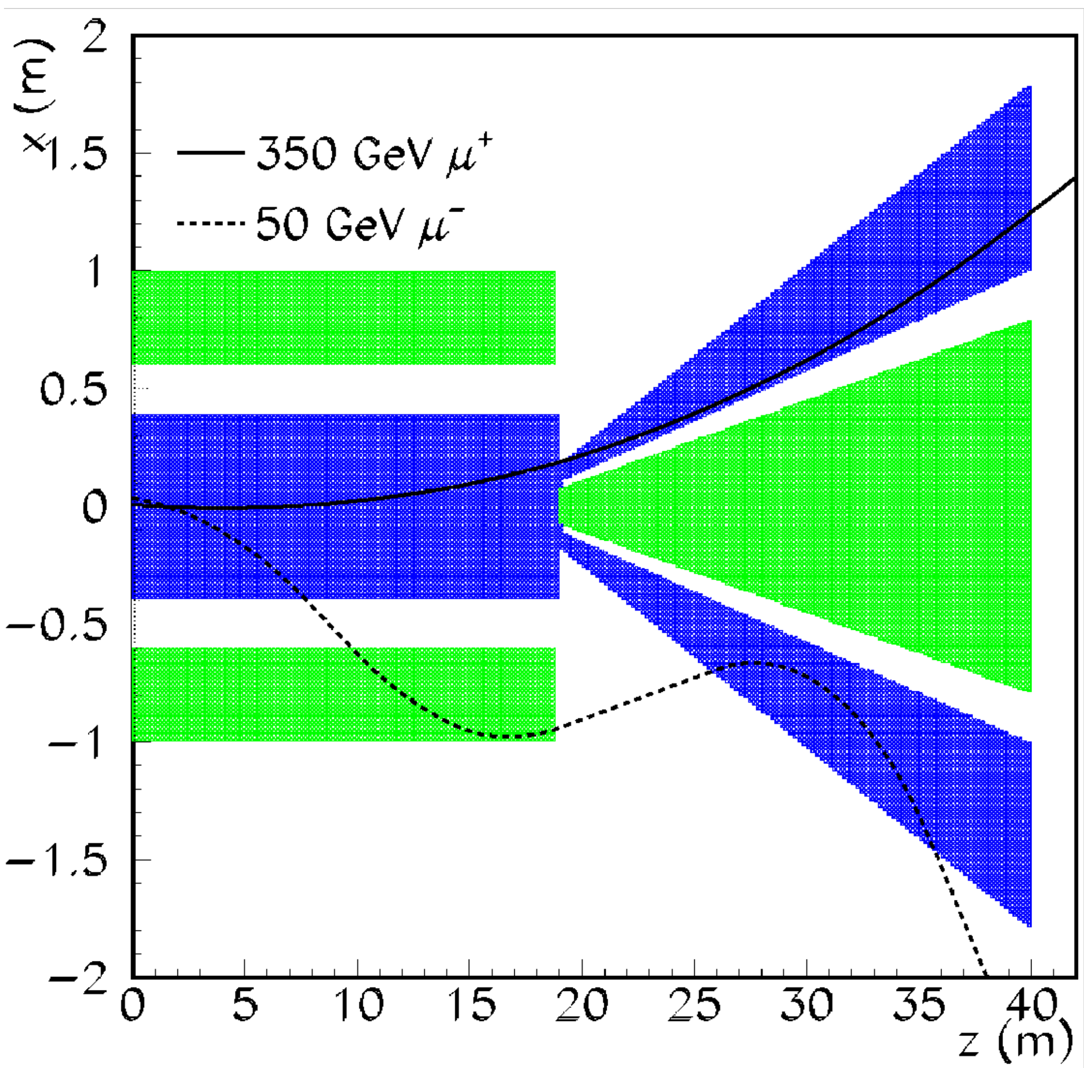


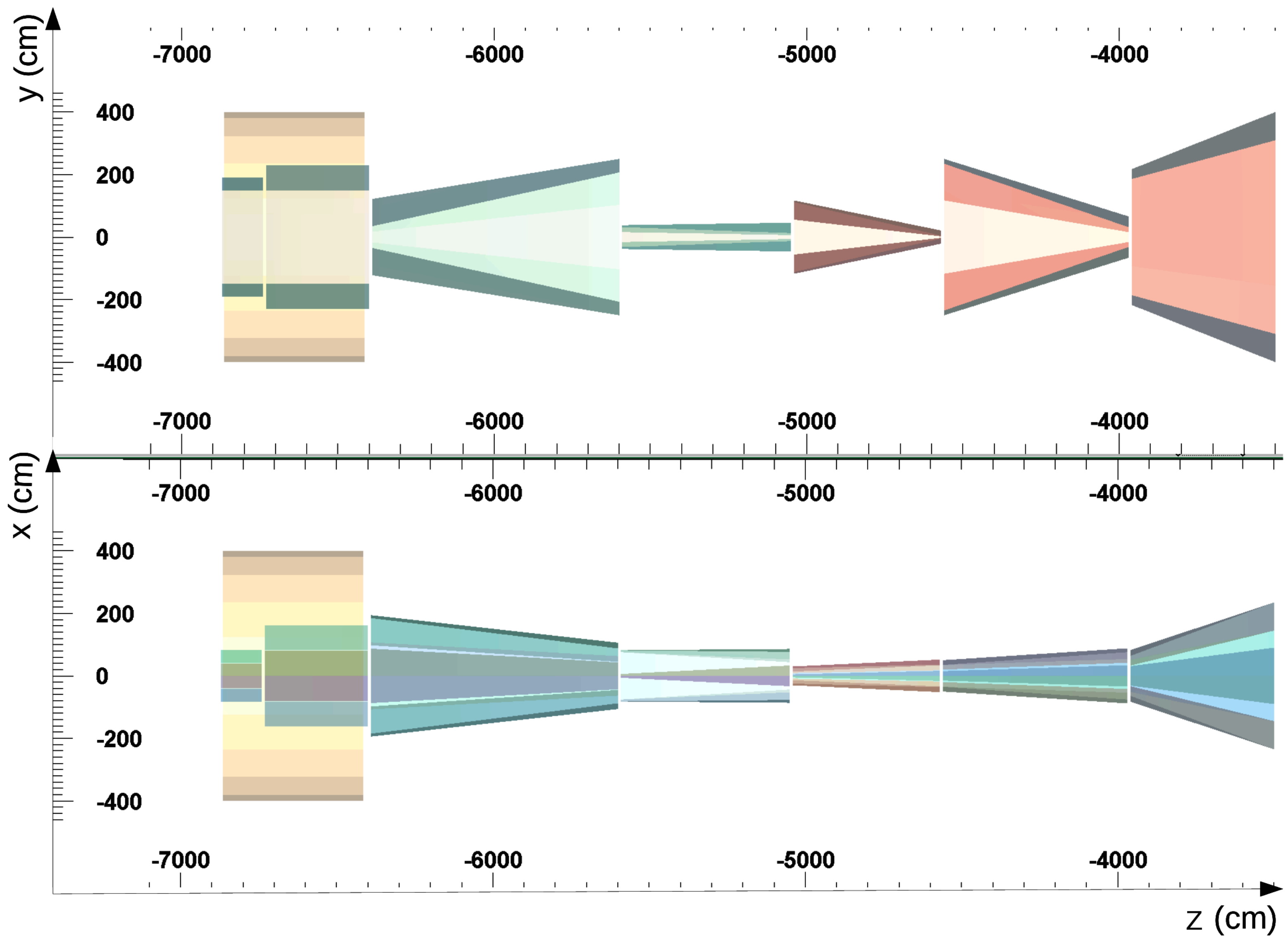
- ◇ Search for Hidden Particles
  - ◇ Post-LHC era experiment for direct search of very weakly interacting light particles

J.Phys.Conf.Ser. 934 (2017) no.1

Active Magnetic Shield



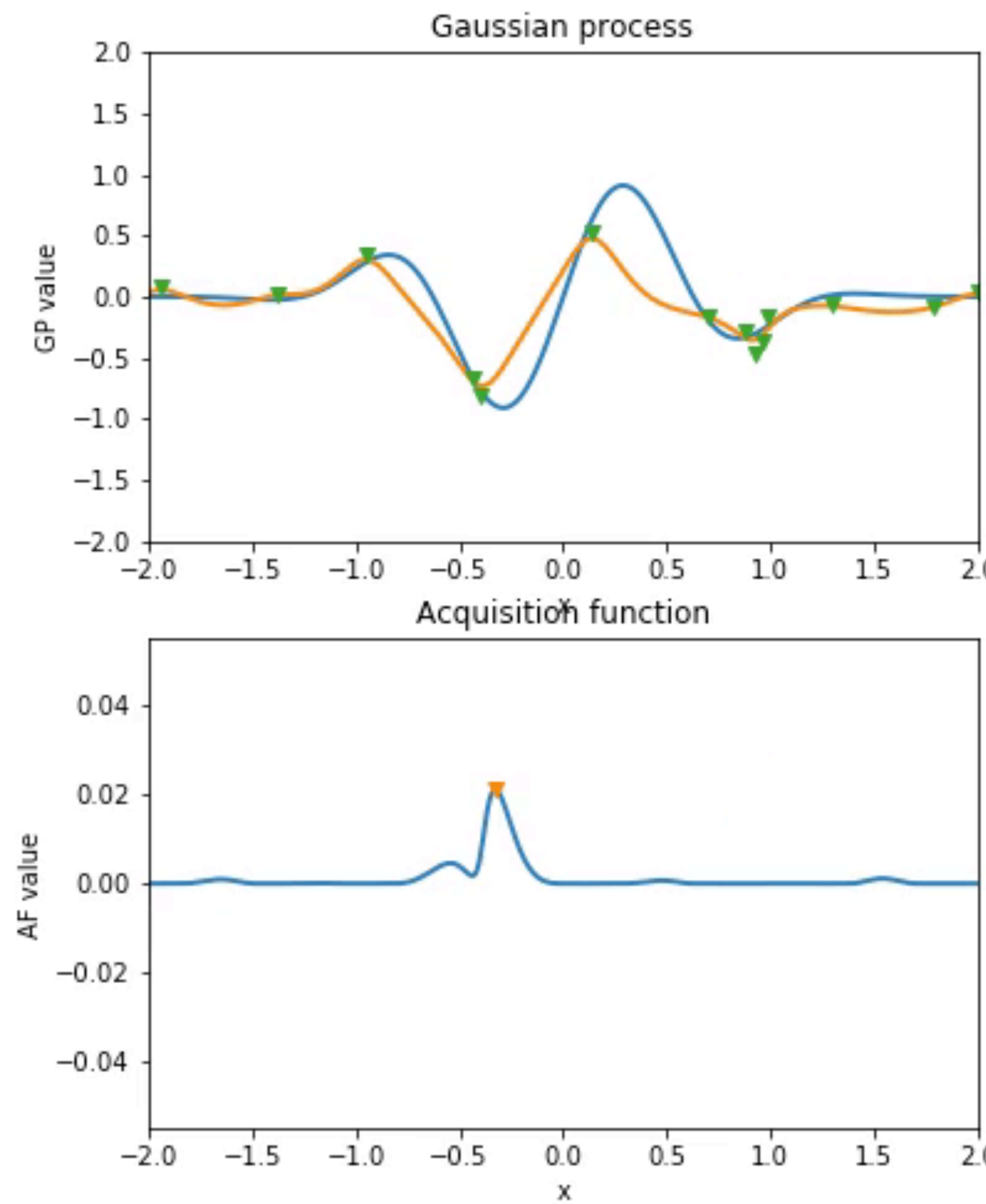




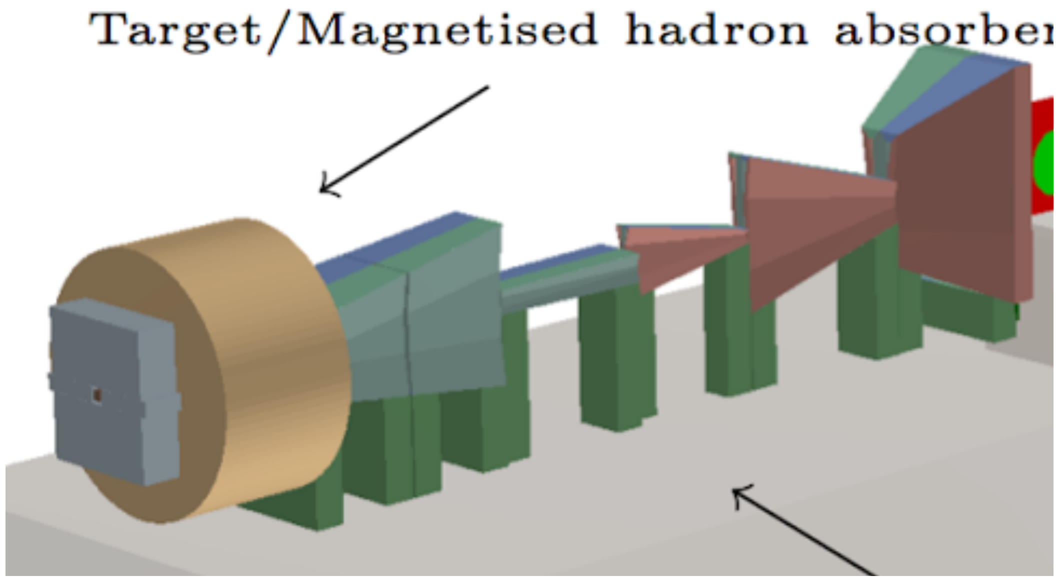
- Absorber shape optimization: background suppression at reasonable cost

Gaussian Process Optimization

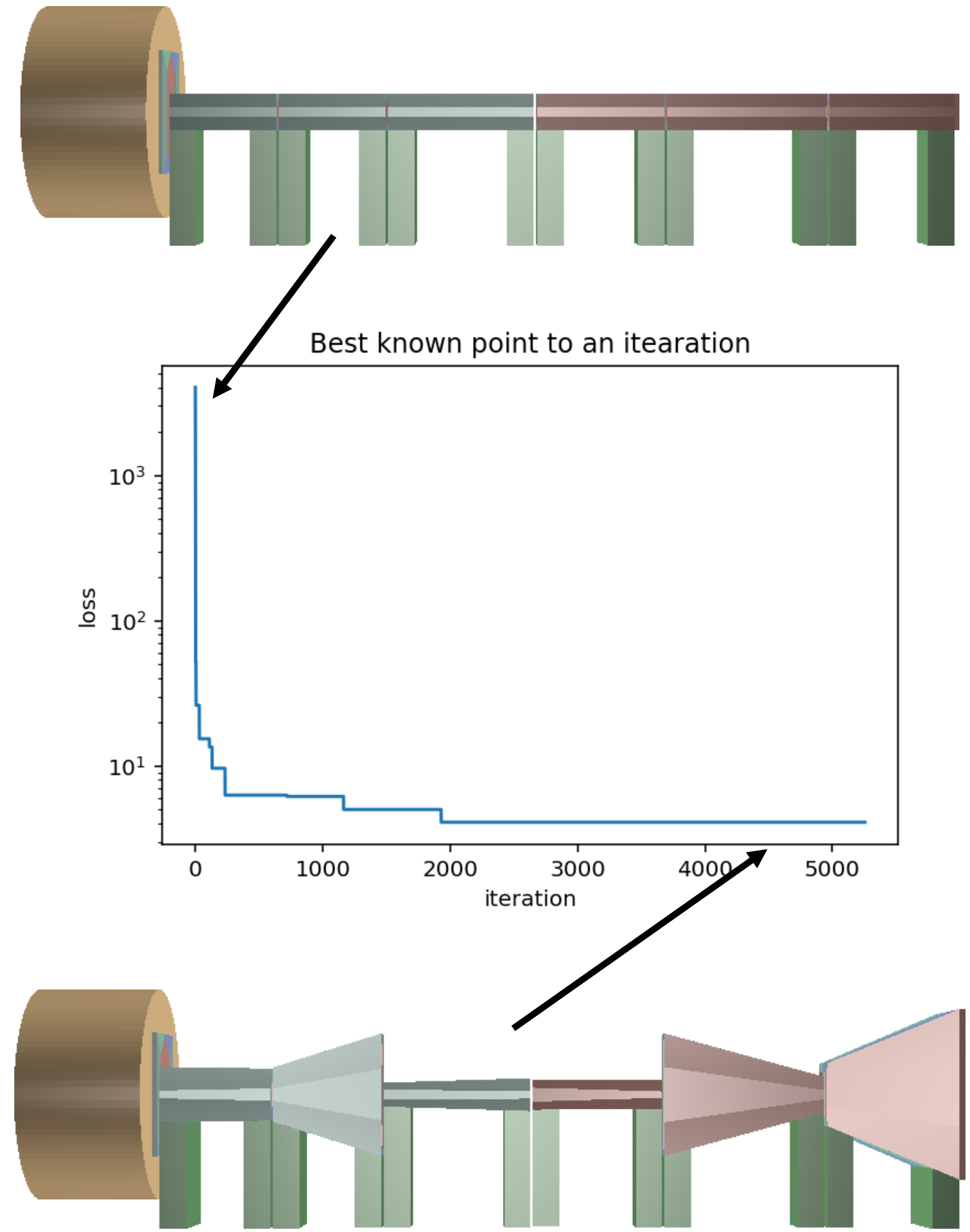
- Loss function includes both background level and cost
- 50+ configuration parameters
  - estimation in every point takes significant time
    - full GEANT simulation of 10+M muons passing through iron
  - loss function is very irregular in the multidimensional parameter space
- Use Gaussian Processes



Shield Optimization



- The same background suppression
- Twice lighter
  - save $$



Advanced optimization methods rule in multidimensional space

Journal of Physics: Conference Series, 2017, Vol. 934, P. 1-5
http://iopscience.iop.org/article/10.1088/1742-6596/934/1/012050/meta

Emerging Challenges: Reliable and Fast Simulation

- Computationally heavy tasks
  - e.g. simulating shower development in the calorimeter
- May be substituted by generative models trained on the original task
  - save orders of magnitude in computing performance
  - challenge is to keep physics performance high

Conclusions

- the first steps in machine learning are extremely easy: we started from a simple linear regression;
- modern machine learning algorithms help processing a lot of information in high-energy physics;
- more interesting applications are coming.