Hyperparameter Optimisation of an Adversarial Neural Network in the tW channel at 13 TeV with ATLAS

Christian Kirfel

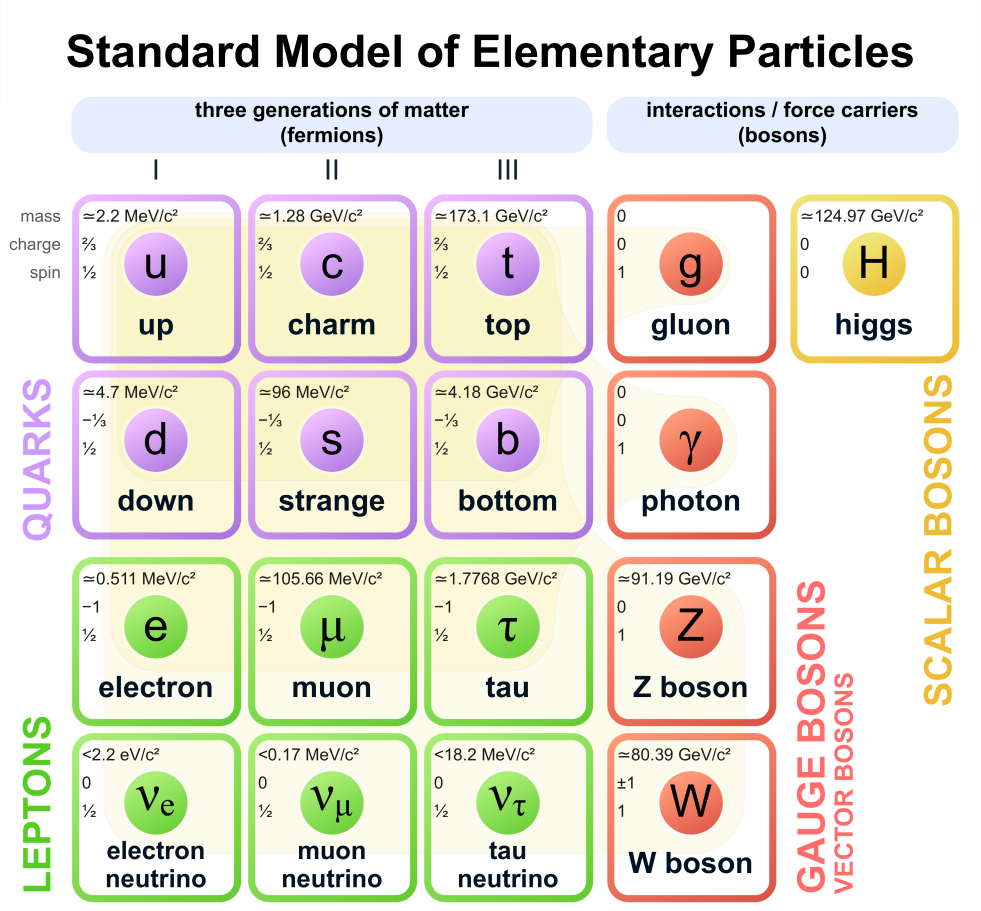
31. März 2019

Universtität Bonn

Outline

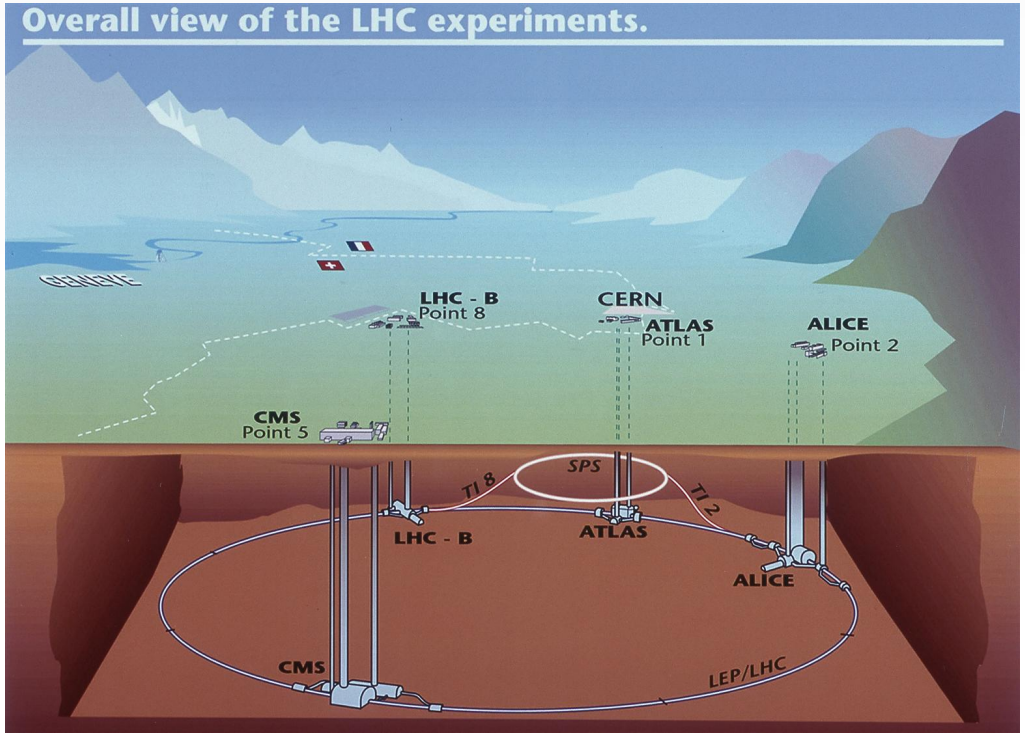
- The Standard Model of particle physics
- The LHC and the ATLAS detector
- Motivating the tW channel
- Neural networks as a possible solution
- Results for different approaches of adversarial neural networks

The standard model of particle physics



[5]

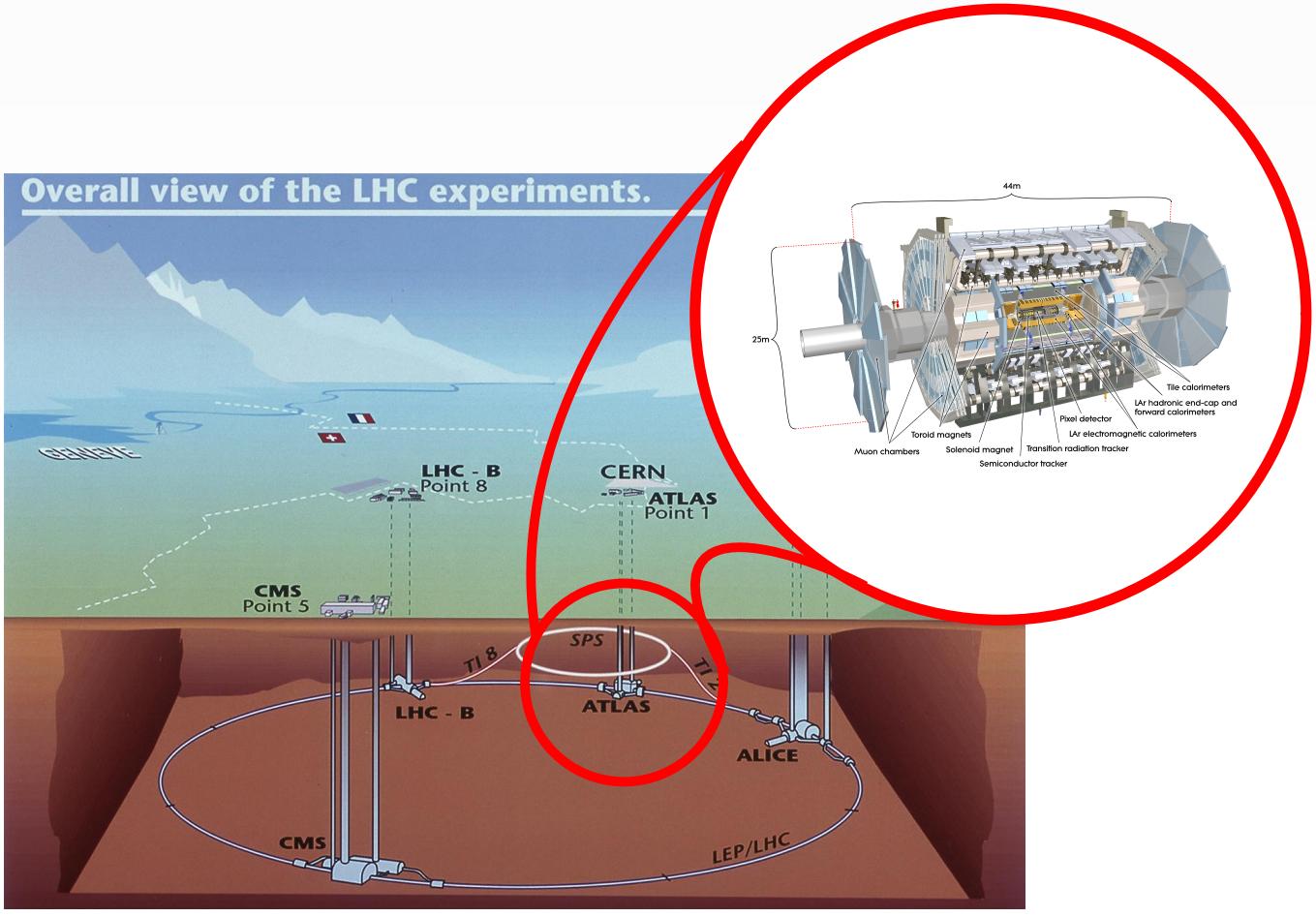
The Large Hadron Collider - LHC



[6]

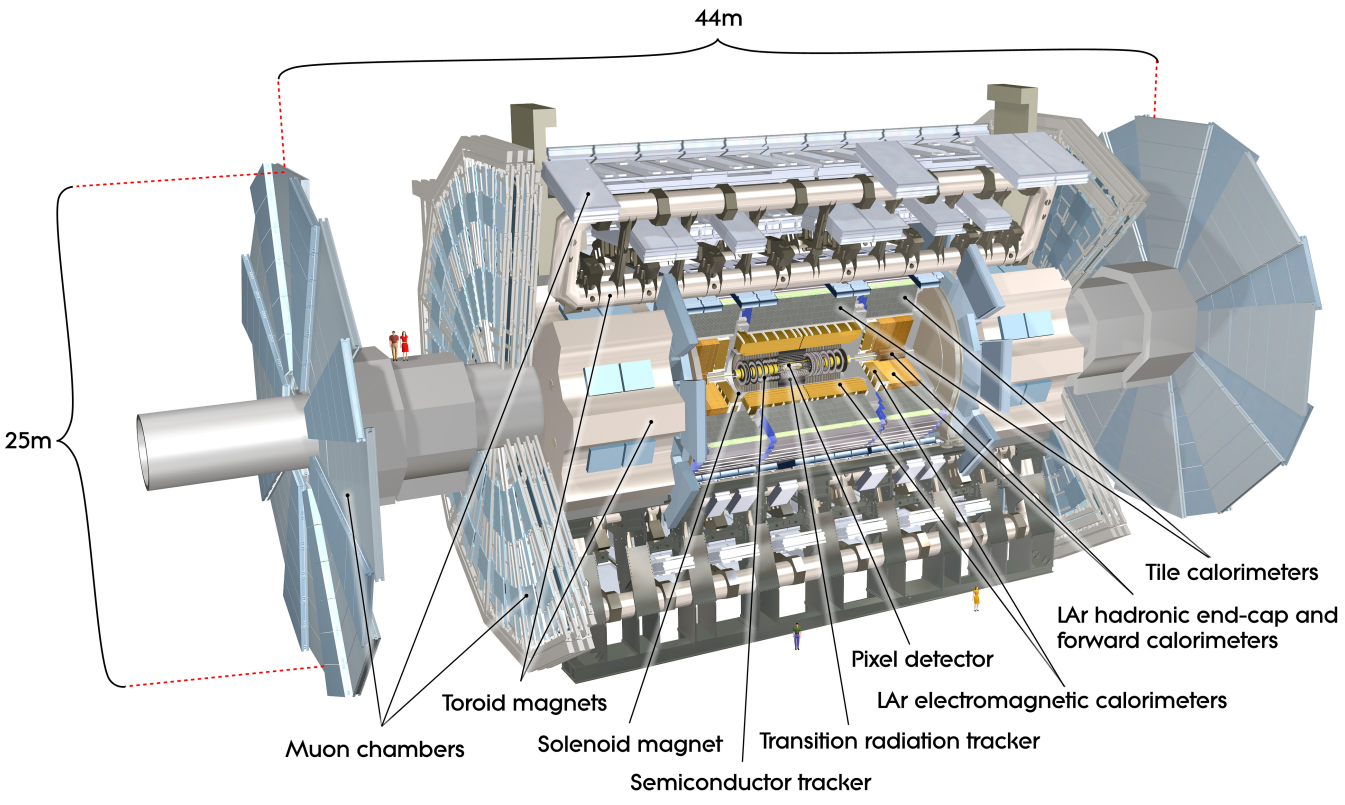
$$E_{CMS} = 13\,\text{TeV}$$
$$\mathcal{L} = 1{,}5 \cdot 10^{34}\,\text{cm}^{-1}\,\text{s}^{-2}$$

The ATLAS experiment



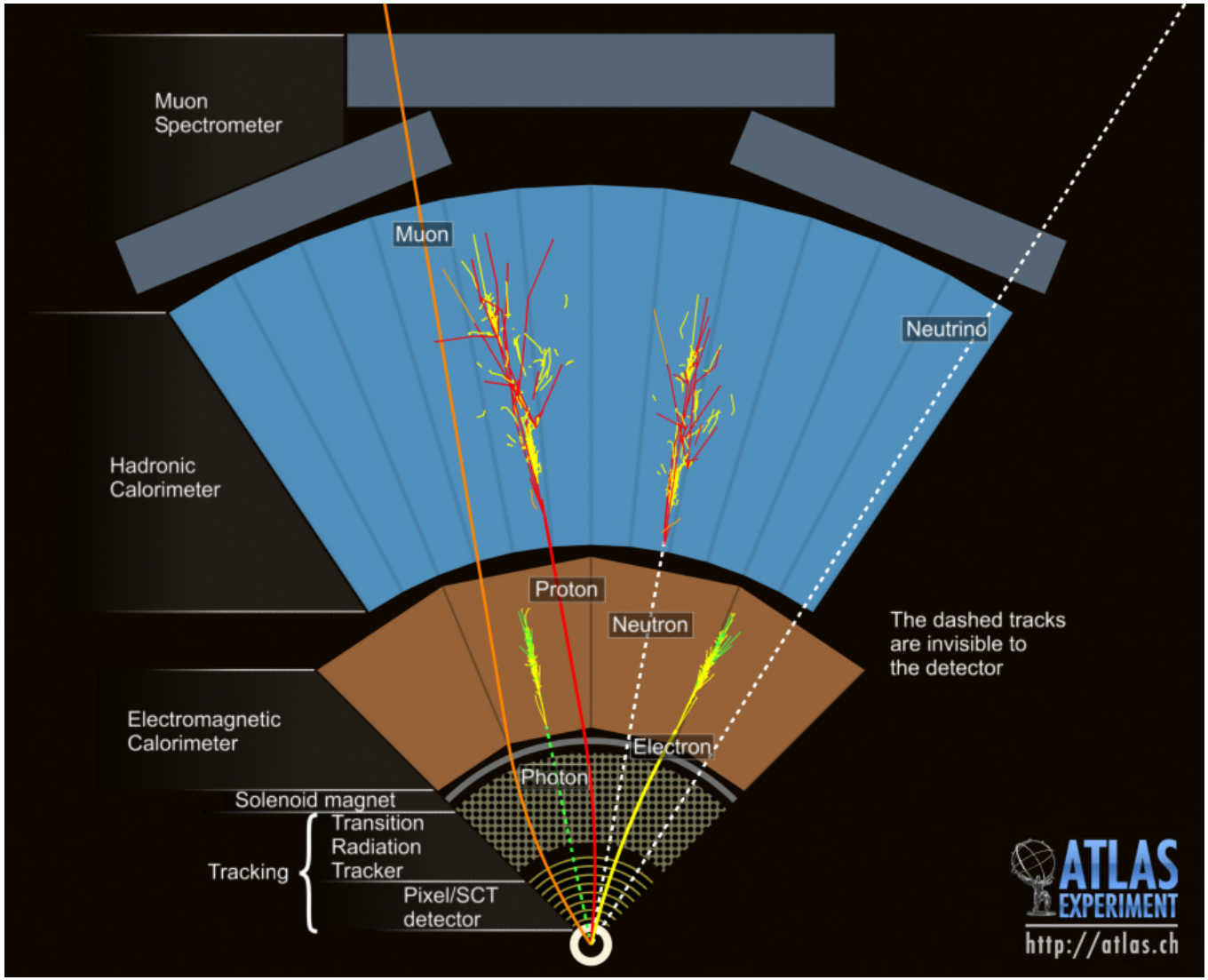
[6]

ATLAS - a genereal purpose detector



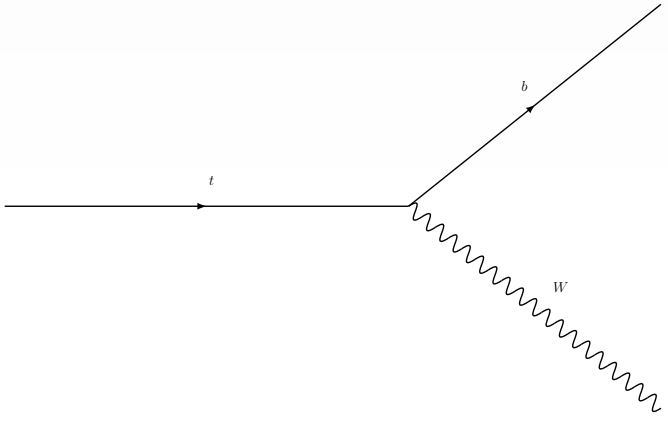
[6]

ATLAS - object identification



[6]

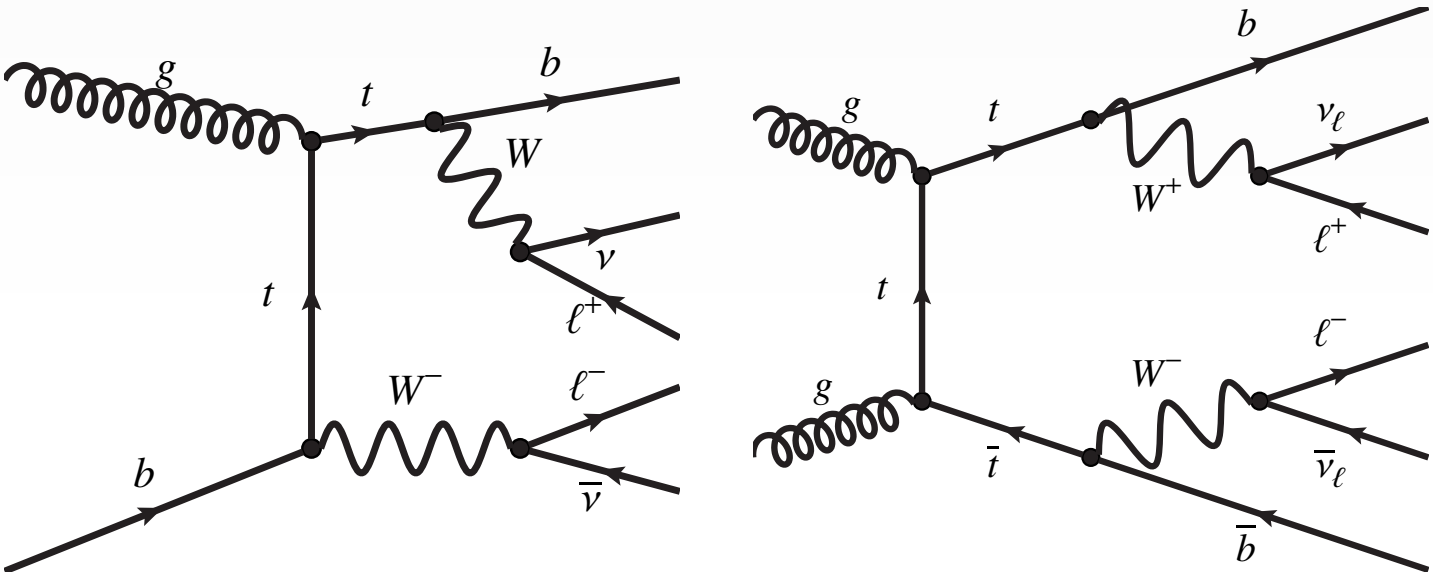
Top physics



- $m \sim 173\,\mathrm{GeV}/{c_0}^2$
- $\tau \sim 5 \cdot 10^{-25}\,\mathrm{s}$

- Decays almost always into a top-quark and a W-boson
- Lifetime shorter than typical time of hadronisation
- No bound states

tW to $t\bar{t}$ separation at LO



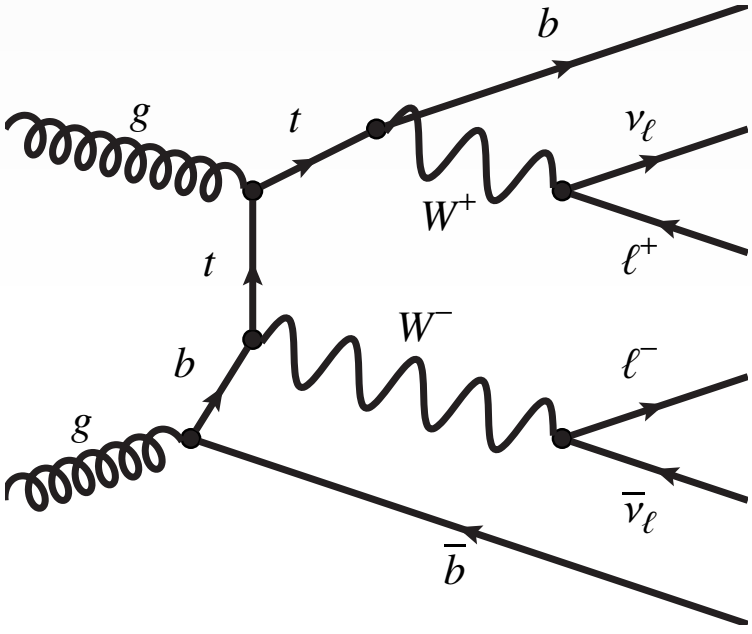
tW decay

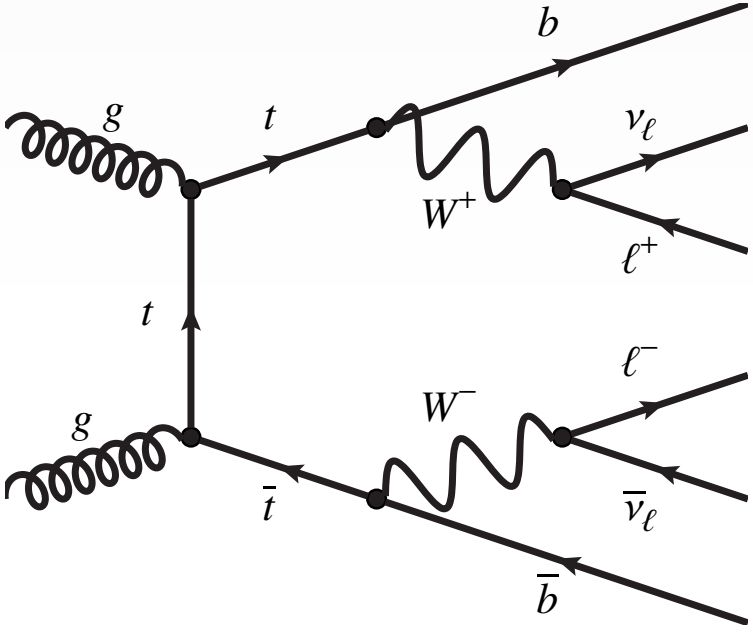
$\sigma_{tW} \sim 71{,}7\,\text{pb}$
Final state: 2 W, 1 b

$t\bar{t}$ decay

$\sigma_{t\bar{t}} \sim 832\,\text{pb}$
Final state: 2 W, 2 b

tW to $t\bar{t}$ separation at NLO





tW decay at NLO

Final state: 2 W, 2 b

$t\bar{t}$ decay

Final state: 2 W, 2 b

Challenge 1 - Signal to background separation

Problem 1

- Separation of signal to background
- Signal: tW_*DR*, tW_*DS*
- Background: $t\bar{t}$

Classic approach

Applying cuts

Alternative

- Machine Learning
- In particular: Classifying neural network

Artificial neural network

Neural Networks - Processing information

Human senses

- Extract relevant information
- Not possible for machines

Human brain

- Web of neuron cells
- A neuron cell takes input from other cells
- Combination of signals creates a result

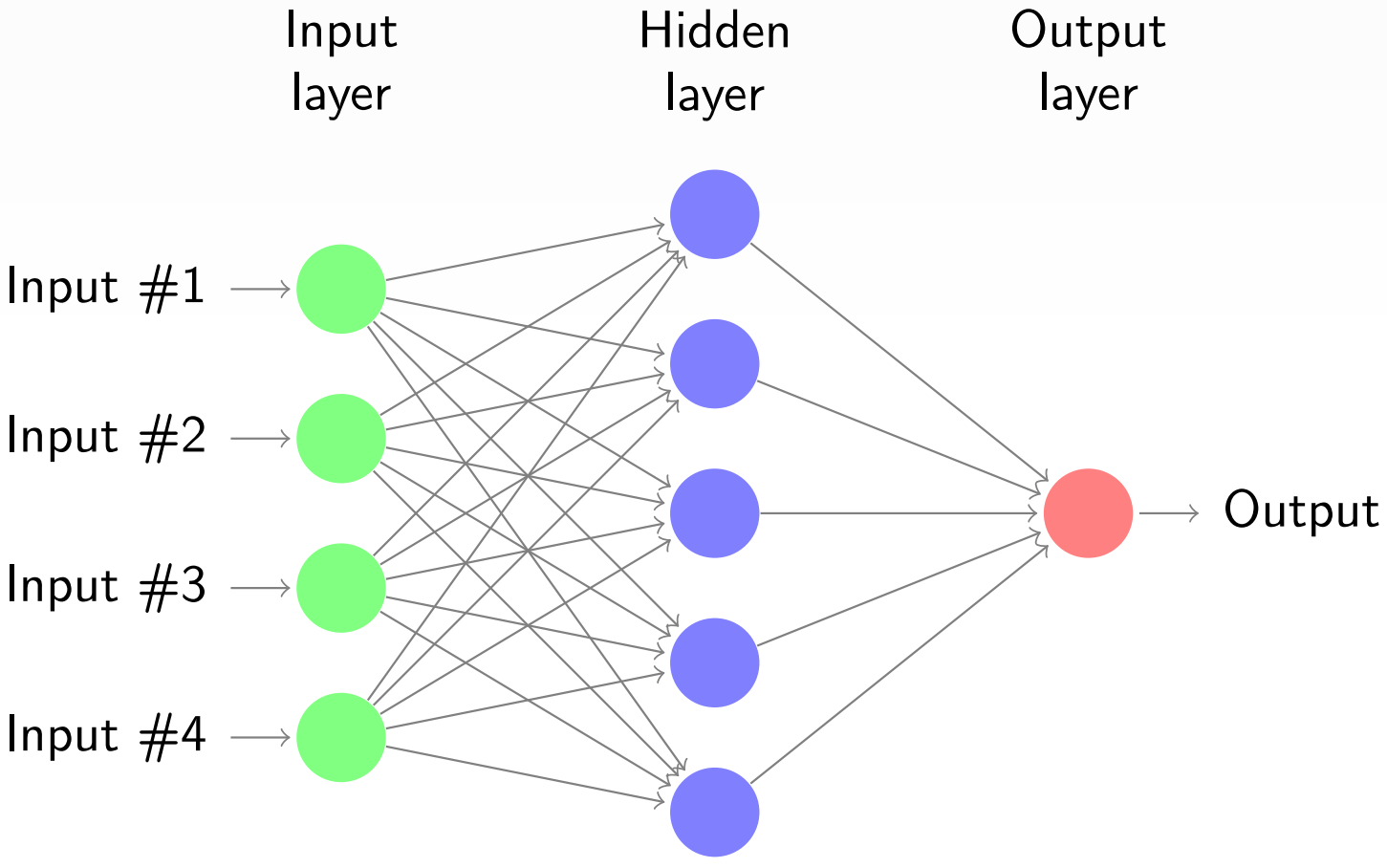
Input variables

- Preprocessed by human user
- e.g. kinematic variables of an event

Net of nodes

- Nodes are simple processors
- Connected by linear function (weight and bias)
- Combines input using an activation function

Neural network structure



[NN_sketch]

Neural Networks - Choosing the next step

Evaluation of an action

- Simple perceptions: pain, satisfaction
- Expected or desired outcome

Deciding for a better action

- Trial and error
- Learning from experience

Loss function

- Supervised learning: compare to the desired outcome
- Loss is an estimator for a model's quality

Optimisation

- Propagate the impact of the parameters on the loss backwards
- Adjust parameters to achieve a model representing a smaller loss

Challenge 2 - Sensitivity to systematic uncertainties

Problem 2

- Minimal sensitivity to the systematic uncertainty
- Nominal: tW_*DR*, ($t\bar{t}$)
- Systematic: tW_*DS*

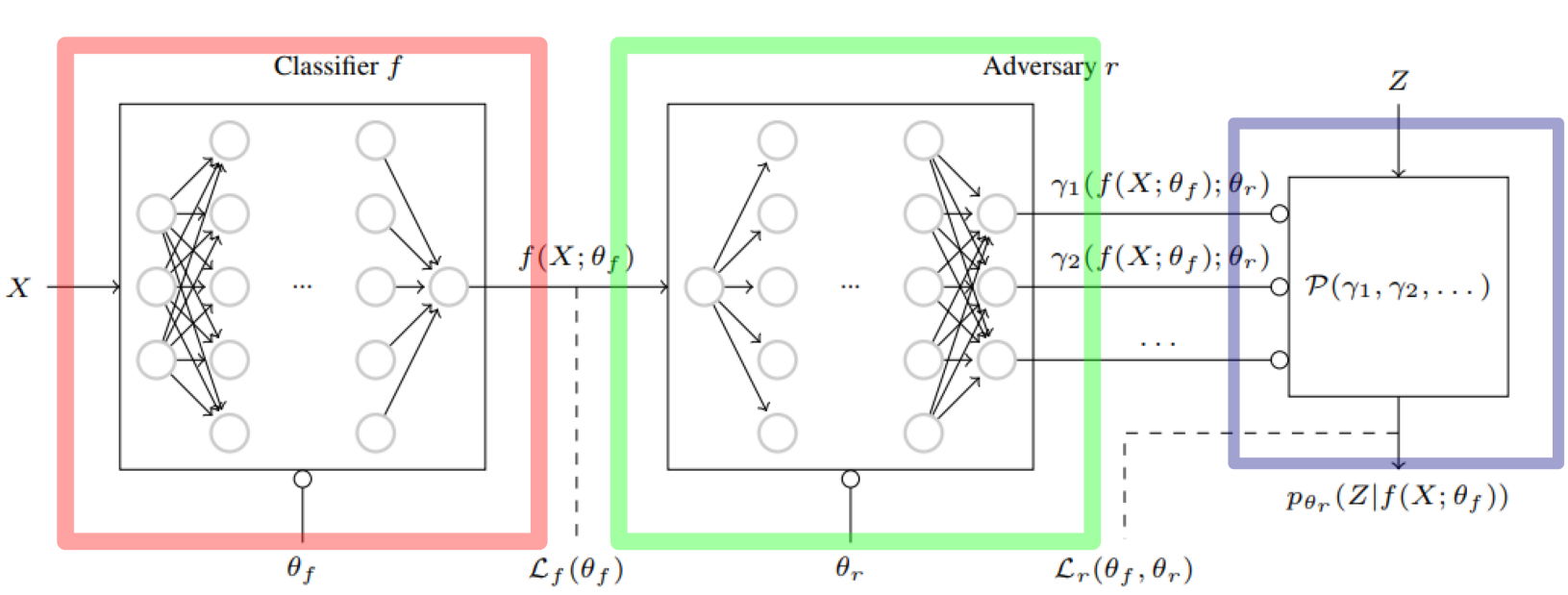
Presented solution

- Addition of a second classifier for nominal to systematics separation
- Bad performance $\longrightarrow$ low sensitivity

Implementation

- Feed output into a second net
- Iterative training
- Combined loss function $\longrightarrow$ Minimax problem

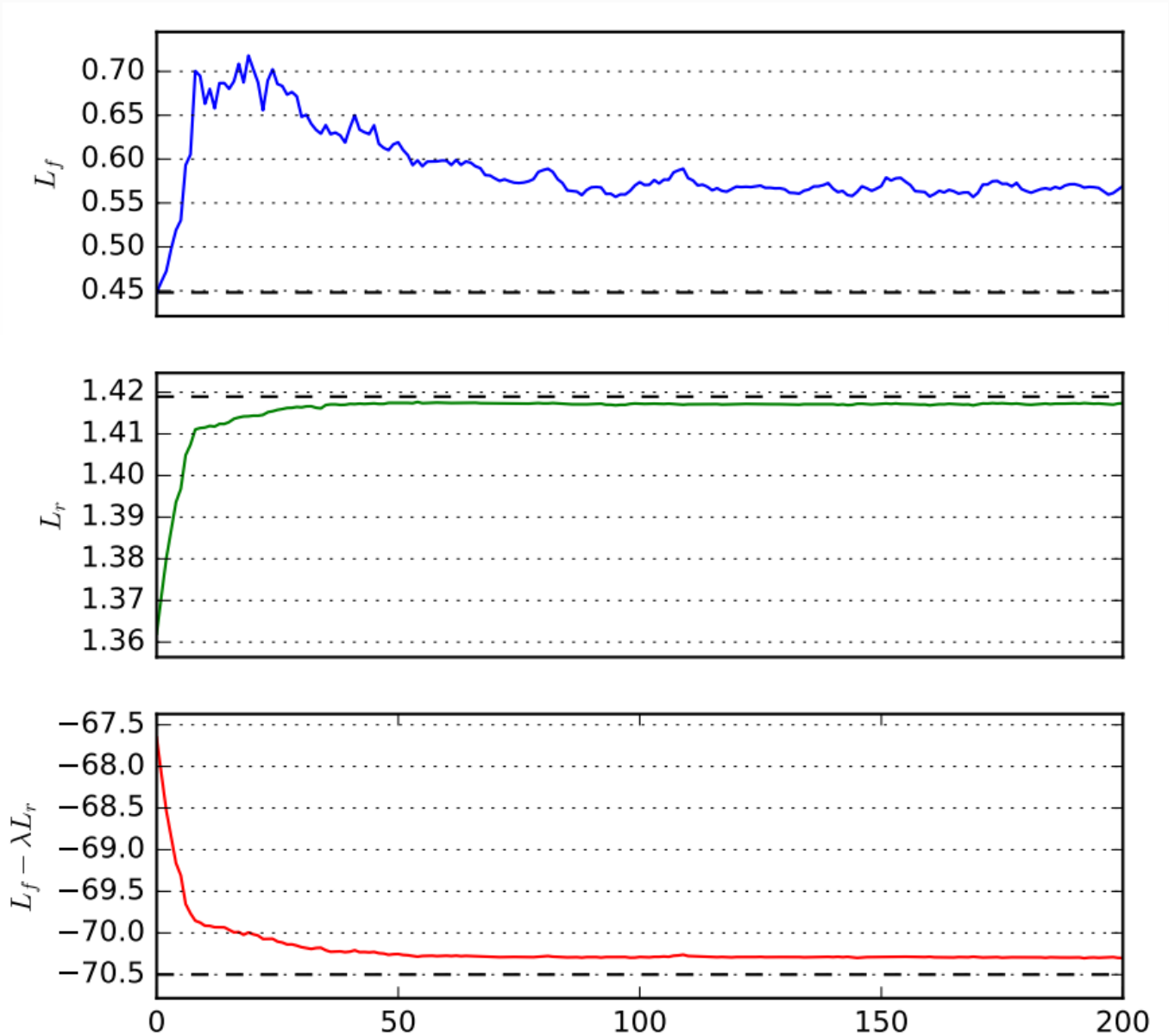
Adversarial neural network



[3]

$$\mathbb{E}(\theta_f, \theta_r) = \mathcal{L}_f(\theta_f) - \lambda \mathcal{L}_r(\theta_f, \theta_r)$$

Expected ANN losses



Classifier training

Setup of the classifier

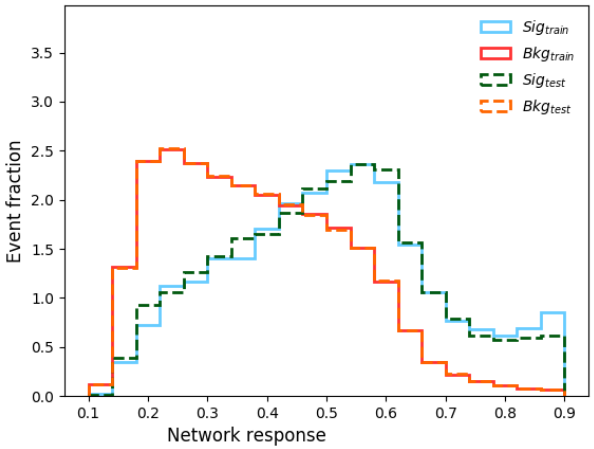
Hyper-parameter scan results

- Input: 14 variables motivated by a BDT variable scan.
- Hidden layers: 6 elu layers $\times$ 128 nodes each
- Output layer: 1 sigmoid node
- Optimisation: SGD, Learning rate $= 0.06$, momentum $= 0.3$, no nesterov, no decay
- Duration: 600 epochs

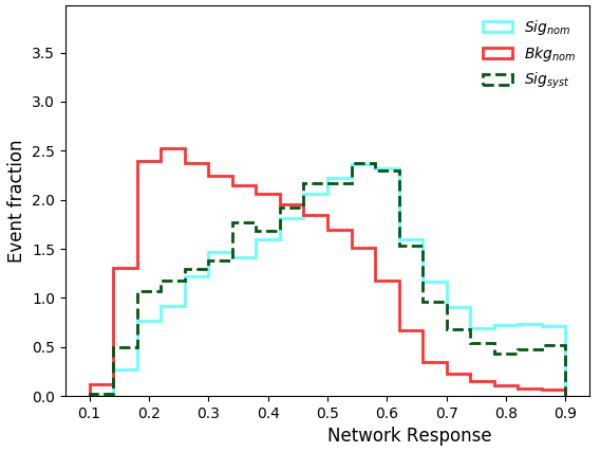
Variables

- Motivated by a BDT significance scan
- Tested against simple kinematic variables

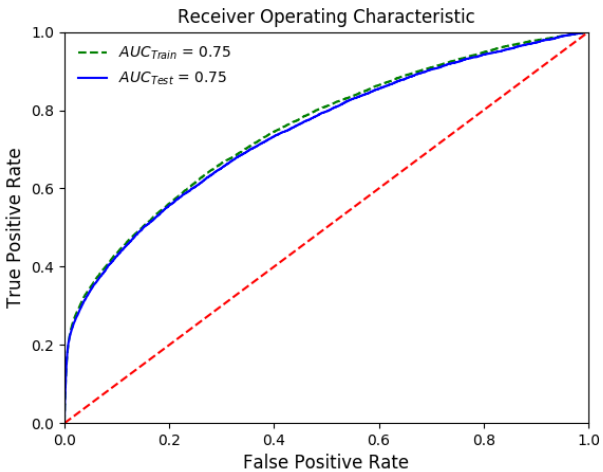
Simple network results



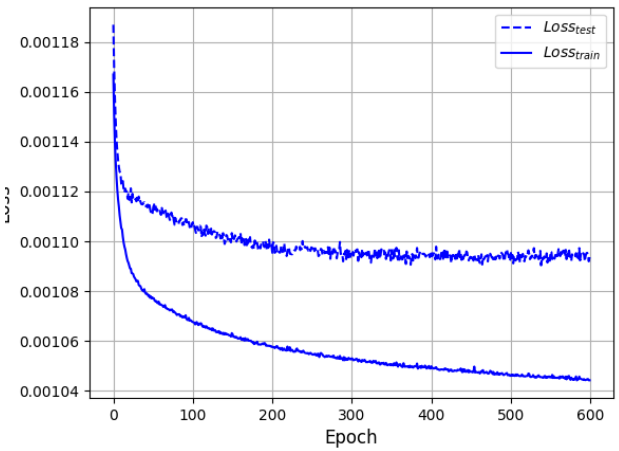
(a) Separation



(b) Systematics



(c) ROC curve



(d) Losses

Adversarial training

Approach 1

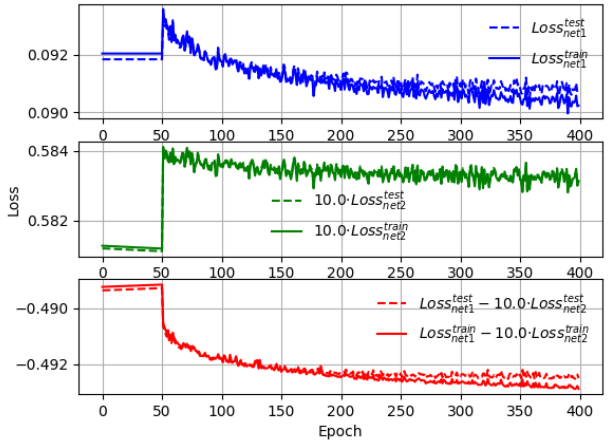
Run 1

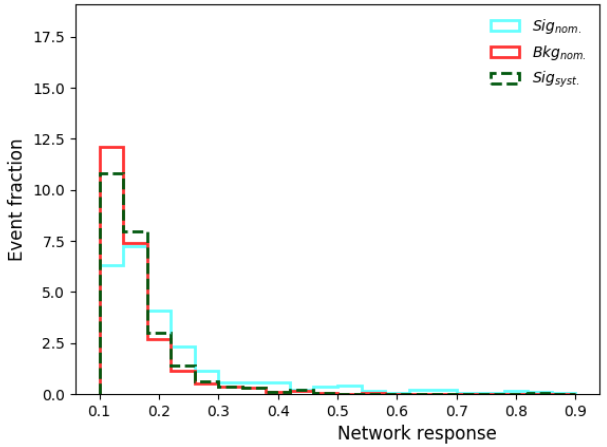
- Same hyper-parameters for both networks
- Adversary uses only 3 layers

Setup

- Input: 14 variables motivated by a BDT variable scan.
- Hidden layers: 6(3) elu layers $\times$ 128 nodes each
- Output layer: 1 sigmoid node
- Optimisation: SGD, Learning rate $= 0.06$, momentum $= 0.3$, no nesterov, no decay
- Duration: 400 iteration
- $\lambda = 10$

Approach 1 - Run 1 - Results





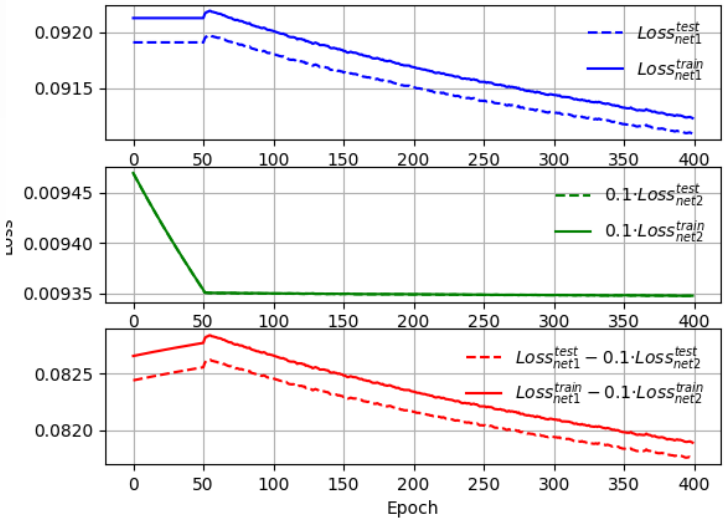
- Both losses decreasing
- Only combined losses depend on λ
- Bad separation
- High sensitivity to systematics

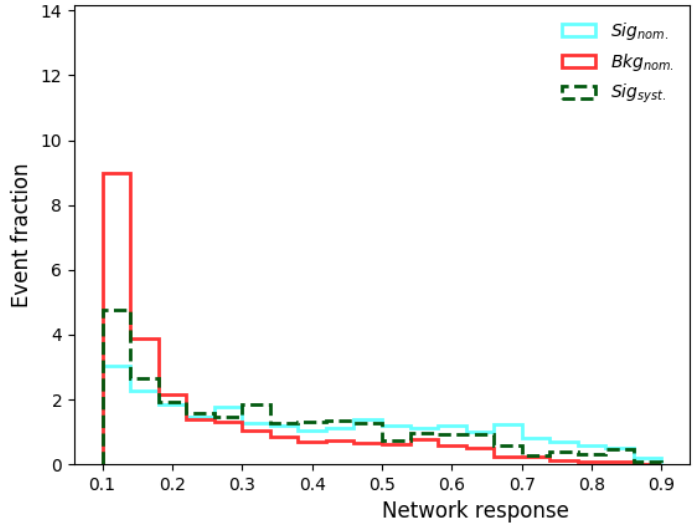
Approach 1

Setup of the adversary

- Input: 14 variables motivated by a BDT variable scan.
- Hidden layers: 3 elu layers $\times$ 32 nodes each
- Output layer: 1 sigmoid node
- Optimisation: SGD, Learning rate $= 0.001$, momentum $= 0.$, no nesterov, no decay
- Duration: 400 iteration
- $\lambda = 0.1$

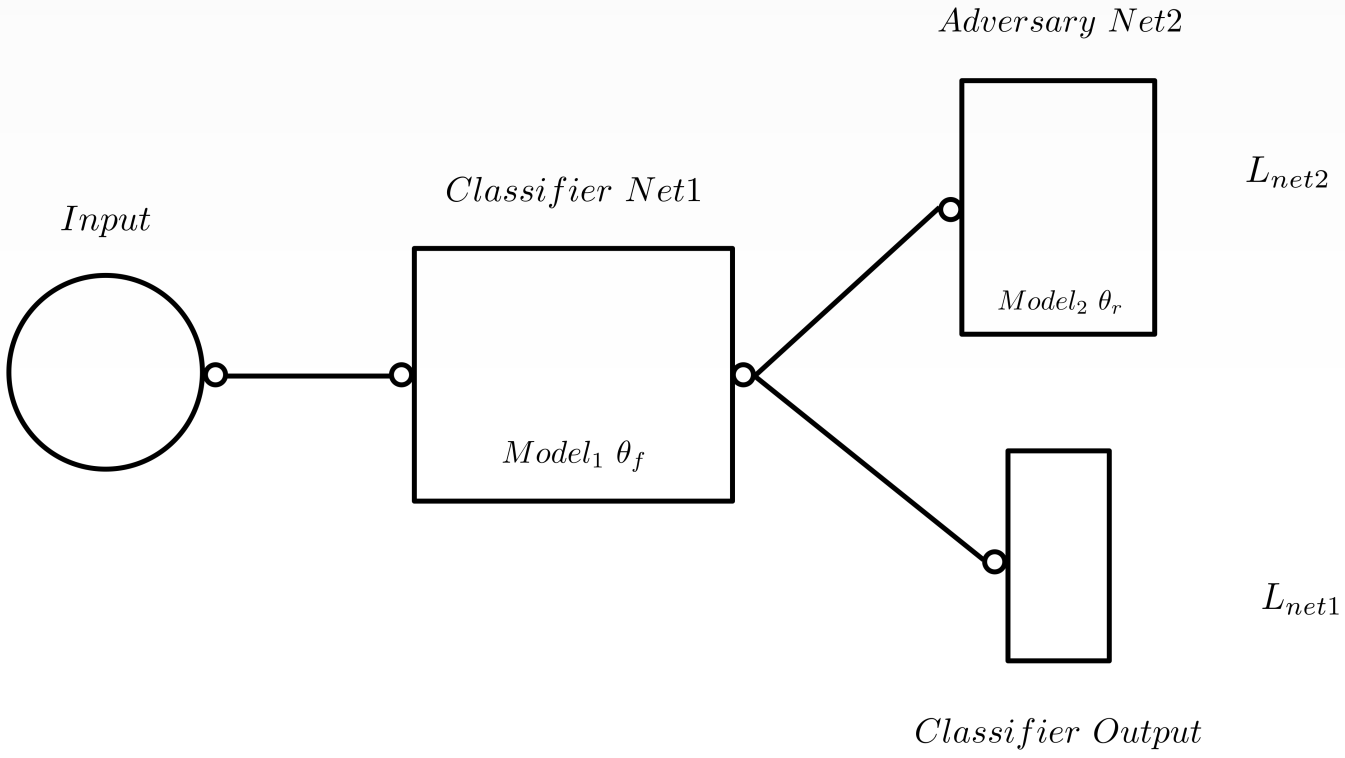
Approach 1 - Run 2 - Results





- Visible separation
- Adversary loss stops falling
- No improvement in the sensitivity visible

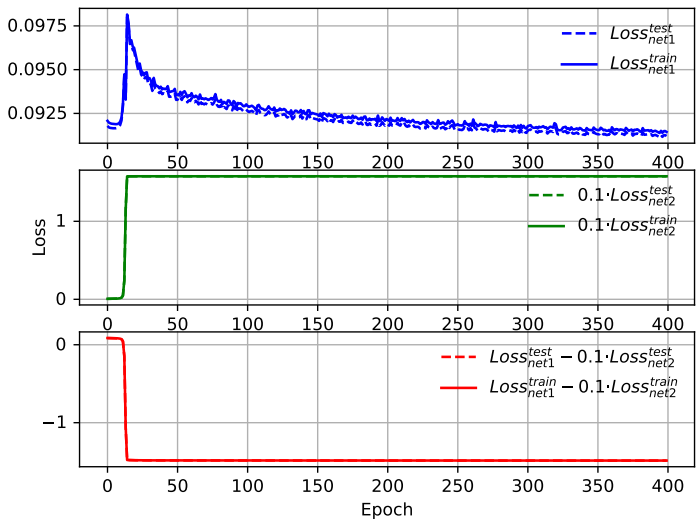
Approach 2

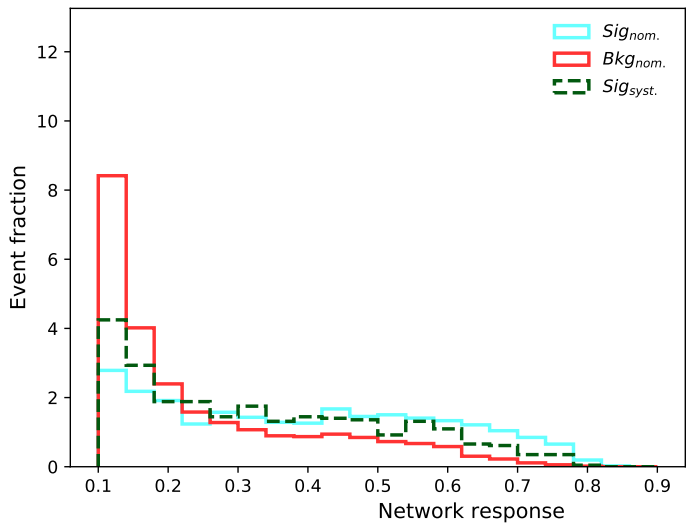


Input the last hidden layer to the adversary

- More information provided
- Possible danger: Missing last decision step

Approach 2 Results





- Losses behave as desired
- steep rise, although partially due to scale
- λ almost unusable due to difference in magnitude
- still high sensitivity

Approach 3

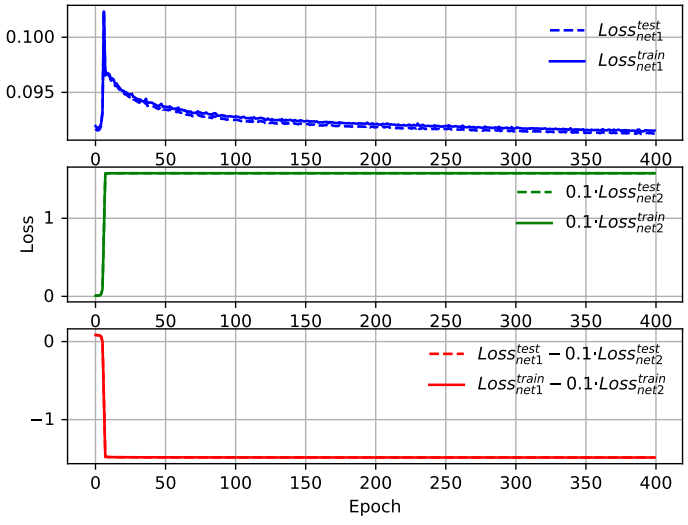
Assumption

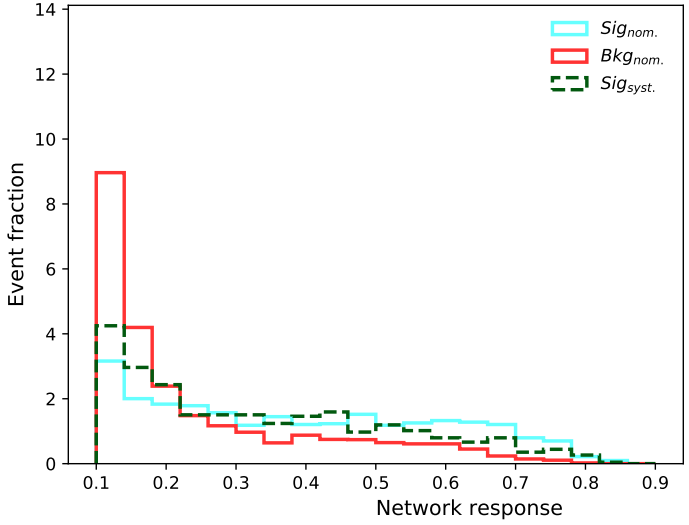
Dimension of the hidden layer is too high

Easy workaround

- Add an intermediate layer
- Reduced to 16 nodes

Approach 3 Results





- Almost no difference
- Slightly higher stability

Thoughts

Input

- Different variable set $\rightarrow$ different problem topology
- Concatenate layers as input $\rightarrow$ Combination of more information and final output
- Separate background from the adversary's input $\rightarrow$ remove possible bias

General

- Different analysis $\rightarrow$ Insight on the network's general inability

Conclusions

Optimisation of a combination of networks

- An adversary cannot just be added
- An ANN has to be set up as a combination

Input of the adversary

- Input of the adversary does not justify the architecture
- Space for improvement in the input

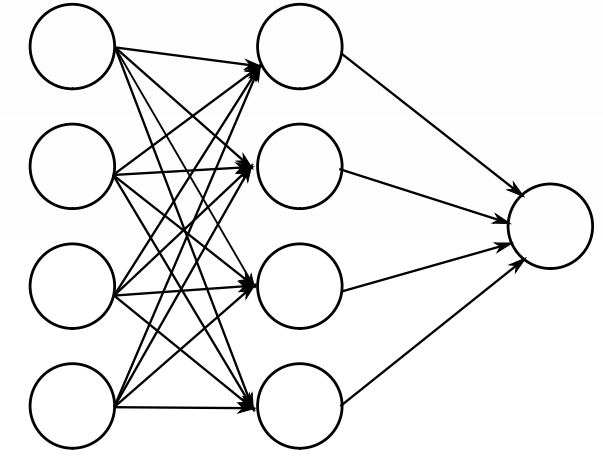
Strong dependency on the learning rate

- Learning rate is the defining factor of the performance of the losses
- Presumably due to the fine topology around the minimum

https://xkcd.com/1838/

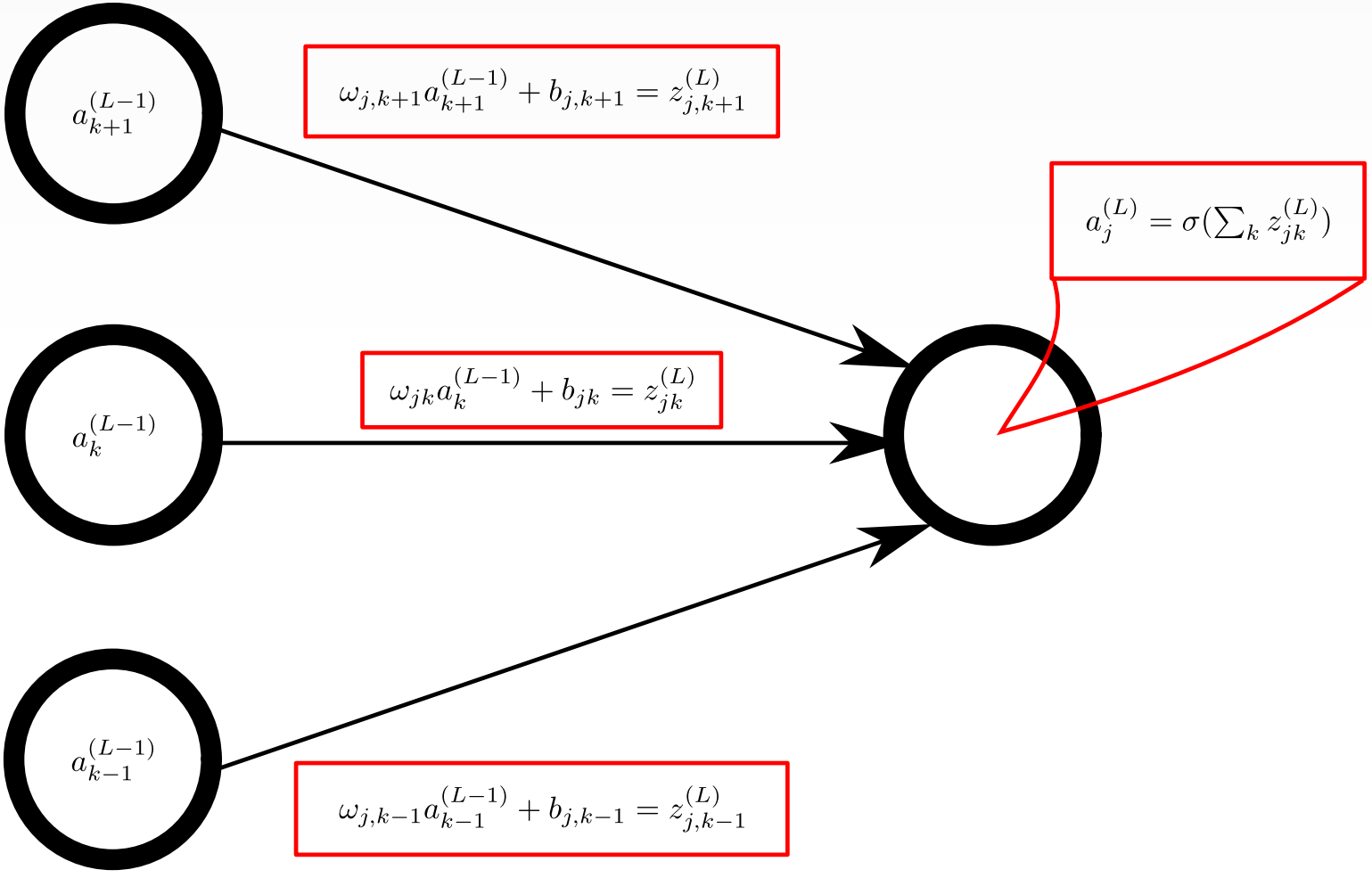
Neural Networks

- Built of layers of simple processors called nodes
- Each node is connected to each node in the surrounding layers
- The connection is built by a linear function with a weight ω and a bias b and a non linear activation function σ
- Learning is accomplished by backpropagation and a step optimizer



$$z_j^L = \omega_{jk}^L a_k^{L-1} + b_k$$
$$a_j^L = \sigma(z_j^L)$$

Neural Networks



Forward Propagation in a neural network

Choosing the next step

- In supervised learning truth tagged training data allows to calculate a cost function to estimate a model's quality. For example crossentropy:

$$C = -(y \log p + (1 - y) \log(1 - p)) \tag{1}$$

- Backpropagation estimates the parameters' impact on the cost function using the partial derivatives

$$\frac{\partial C}{\partial a_k^{L-1}} = \sum_{j=1}^{N} \frac{\partial z_j^L}{\partial a_k^{L-1}} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial C}{\partial a_j^L}$$

- Each parameter is then updated accoring to its impact on the cost function

Adversarial Neural Networks

- Originally introduced as Generative adversarial neural networks to overcome weaknesses of generative networks [2]
- Adds a second network controlling the dependency on features with large systematic uncertainties.
- The adversarial structure of the networks creates a minimax game
- Combined loss function

Optimisers

- Gradient Descent: Update parameters in the opposite direction of the error gradient
- Stochastic Gradient Descent: Gradient Descent but for each training example separately
- Adaptive Moment Estimation, Adam: adaptive learning rate based on exponentially decaying mean of past gradients and past square gradients.
- Others
- https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f

Learning rate, momentum, decay

- Learning rate: Step-length in gradient direction

$$\theta' = \theta - lr \cdot g \tag{2}$$

- Momentum: create an adaptive and weight dependent learning rate

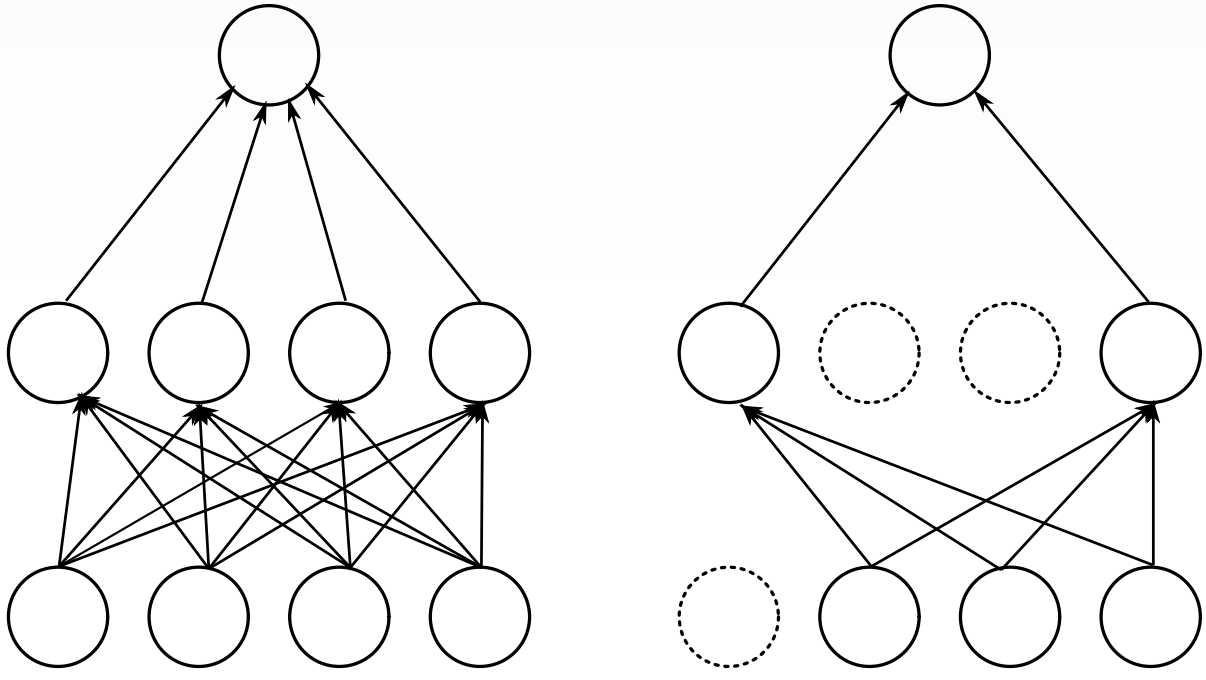
$$\nu' = \alpha\nu - \epsilon\frac{1}{m}\nabla_\theta \sum_j L(f(\hat{y}^j;\theta), y^j) \tag{3}$$

$$\theta' = \theta + \nu \tag{4}$$

- Decay: Decrease the learning rate over the course of the training

$$\epsilon' = \frac{\epsilon}{1+\phi t} \tag{5}$$

Dropout



Without Dropout With Dropout

Caption

Technical details

- The neural networks were built using Keras [1] with tensorflow [4] as backend