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

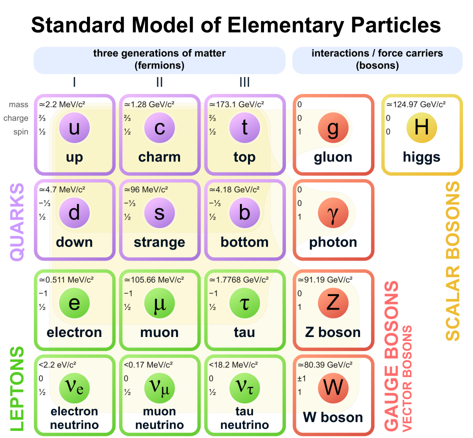
Christian Kirfel

4th April 2019

Outline

- The Standard Model of particle physics
- Introducing the tW channel
- The LHC and the ATLAS detector
- Neural networks in particle physics
- Results for different approaches of adversarial neural networks

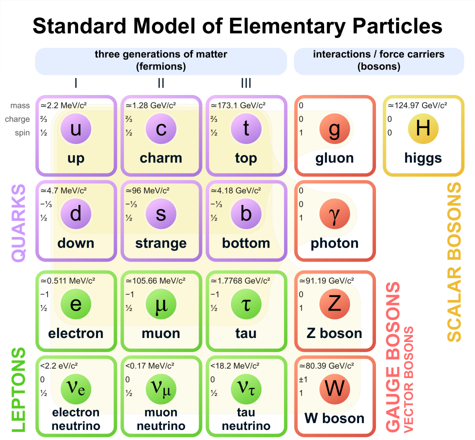
The standard model of particle physics



The top-quark

- $m \sim 173\,\mathrm{GeV}/c^2$
- $\tau \sim 5 \times 10^{-25}\,\mathrm{s}$
- Lifetime < typical hadronisation time
- Decay into a b-quark and a W-boson

The standard model of particle physics

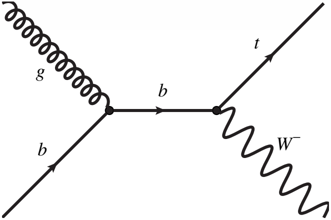


The top-quark

- $m \sim 173\,\mathrm{GeV}/c^2$
- $\tau \sim 5 \times 10^{-25}\,\mathrm{s}$
- Lifetime $<$ typical hadronisation time
- Decay into a b-quark and a W-boson

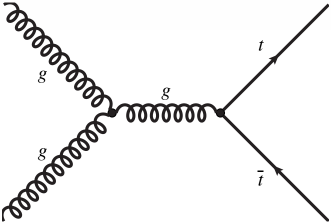
Top production

tW single top production



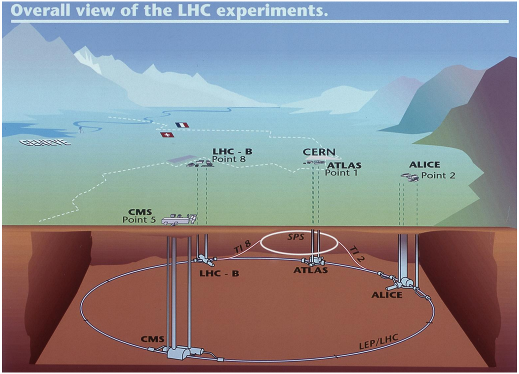
- Via weak interaction
- Relatively small cross section

$t\bar{t}$ pair production



- Via strong interaction
- Over 10 times larger cross-section

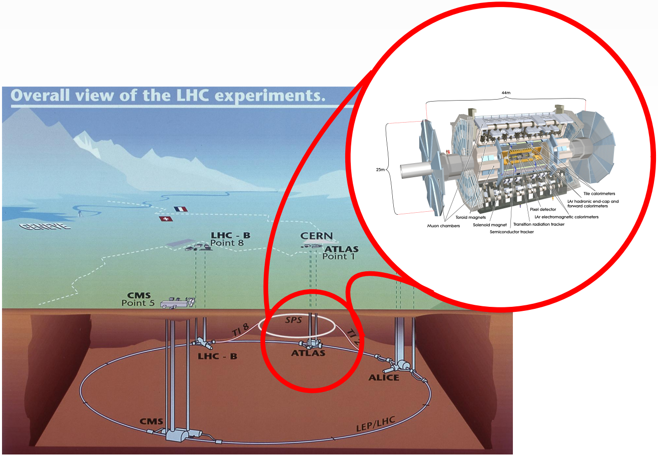
The Large Hadron Collider - LHC



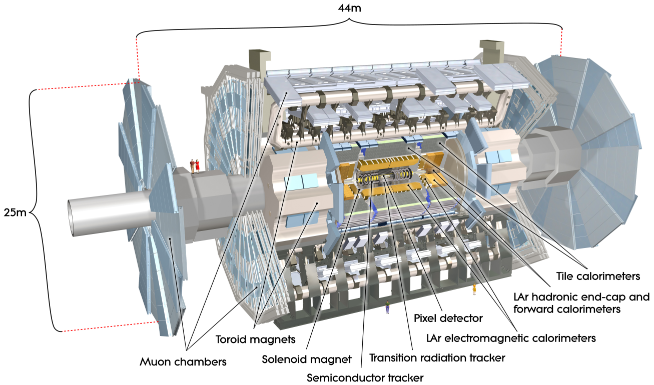
- Proton-proton collider
- $\sim 27\,\mathrm{km}$ circumfence

$$E_{CM} = 13\,\mathrm{TeV}$$
$$\mathcal{L} = 1.5 \times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$$

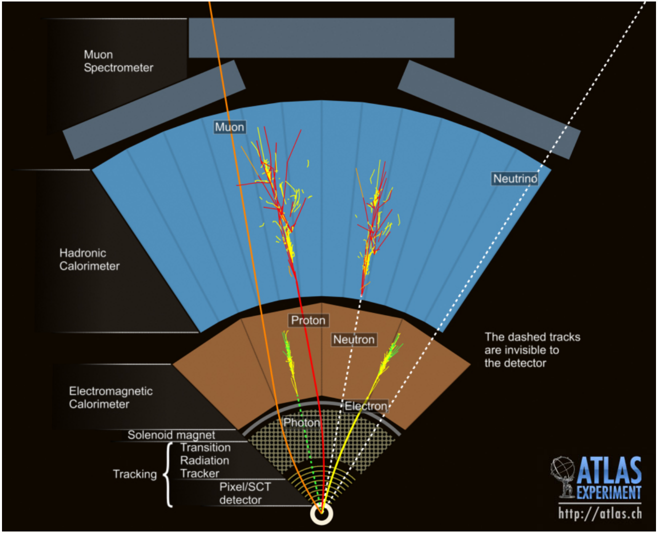
The ATLAS experiment



ATLAS - a genereal purpose detector

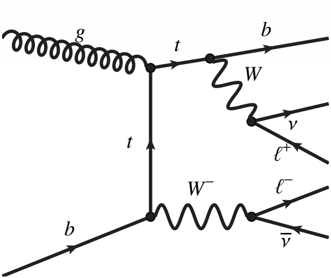


ATLAS - object identification



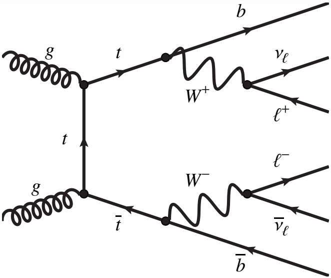
tW to $t\bar{t}$ separation at LO

tW decay



- $\sigma_{\mathrm{tW}} \sim 71.7\,\mathrm{pb}$
- 2 W, 1 b

$t\bar{t}$ decay



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

Challenge 1 - Signal to background separation

Problem 1

- Separation of signal to background
- Signal: tW
- Background: $t\bar{t}$

Classic approach

- Applying a cut selection

Alternative

- Machine Learning
- In particular: Classifying neural network

Artificial neural network

Neural Networks - Processing information

Humam senses

- Extraction of relevant info
- Impossible for machines

Human brain

- Web of neuron cells
- Input from surrounding cells
- Single combination $\rightarrow$ action

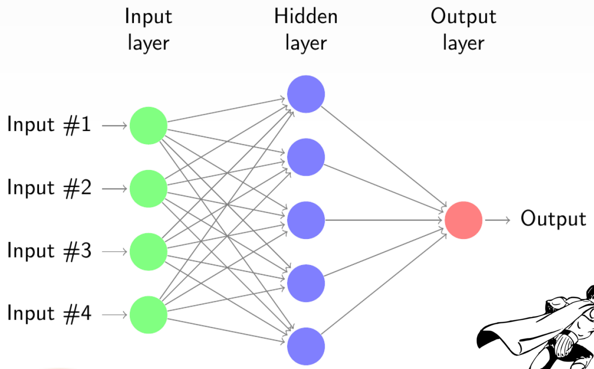
Input variables

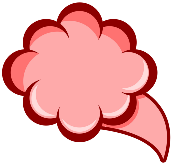
- Preprocessed by user
- e.g. kinematic variables

Net of nodes

- Nodes = simple processors
- Connected by linear function
- Combination forms non-linear model

Neural network structure





Neural Networks - Processing information

Evaluation of an action

- Simple perceptions: pain, satisfaction
- Expectation

Decision for a next step

- Trial and error
- Learning from experience

Loss function

- Supervised learning: compare to the desired outcome
- Loss = estimator for quality

Optimisation

- Back-propagation impact of parameters' on the loss
- Adjust parameters to minimise plot

Classifier training

Parameters and hyper-parameters

Parameters

- Affected during training
- Mainly: weight and bias

Hyper-parameters

- Set before the training
- Optimisation by testruns
- Nodes, Layers, optimiser, momentum, activation.....

Learning rate

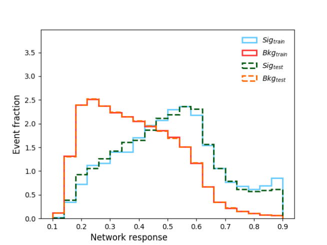
- Step-parameter of the optimiser
- Large: Oscillations
- Small: Inefficient training

Setup of the classifier

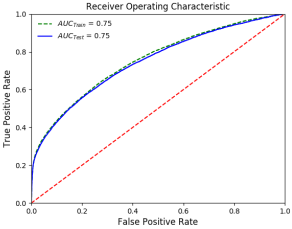
Hyper-parameter scan results

- 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

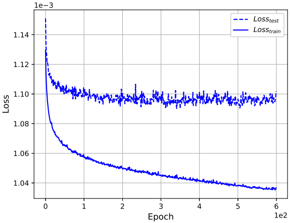
Simple network results



Separation



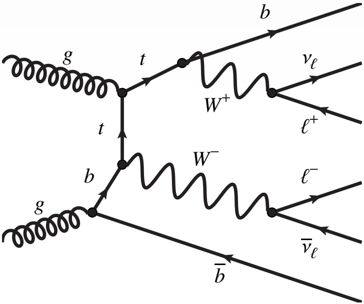
ROC curve



Losses

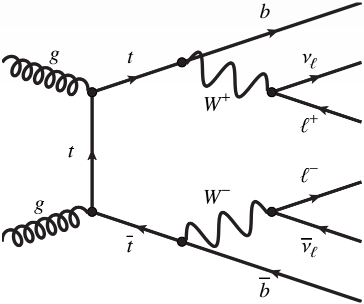
tW to $t\bar{t}$ separation at NLO

tW decay at NLO



- Identical final state
- Especially problematic in 2j2b region

$t\bar{t}$ decay



- Interference at NLO
- Different Monte Carlo generators

Interference in Monte Carlo

Amplitude

$$\begin{aligned}|\mathcal{A}|^2 &= |\mathcal{A}_{\mathrm{tW}}|^2 + 2\mathcal{R}\{\mathcal{A}_{\mathrm{tW}}\mathcal{A}^*_{\mathrm{t\bar{t}}}\} + |\mathcal{A}_{\mathrm{t\bar{t}}}|^2, \\ &\equiv \mathcal{S} + \mathcal{I} + \mathcal{D}.\end{aligned}$$

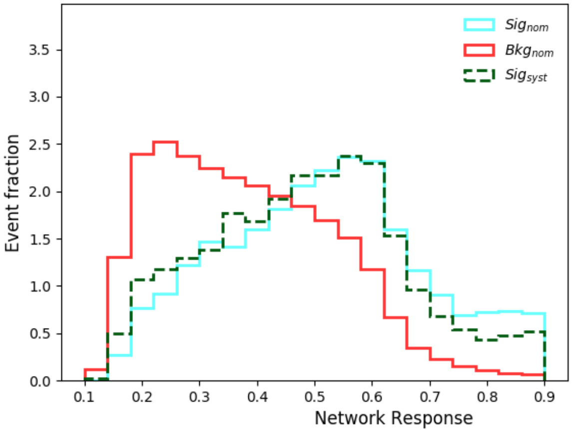
Diagram Removal - DR

$$|\mathcal{A}_{DR}|^2 = \mathcal{S}.$$

Diagram Subtraction - DS

$$\begin{aligned}|\mathcal{A}_{DS}|^2 &= \mathcal{S} + \mathcal{I} + \mathcal{D} - \widetilde{\mathcal{D}}, \\ &\approx \mathcal{S} + \mathcal{I}.\end{aligned}$$

Sensitivity to systemativ uncertainty



Adversarial Neural Network

Challenge 2 - Sensitivity to systematic uncertainties

Problem 2

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

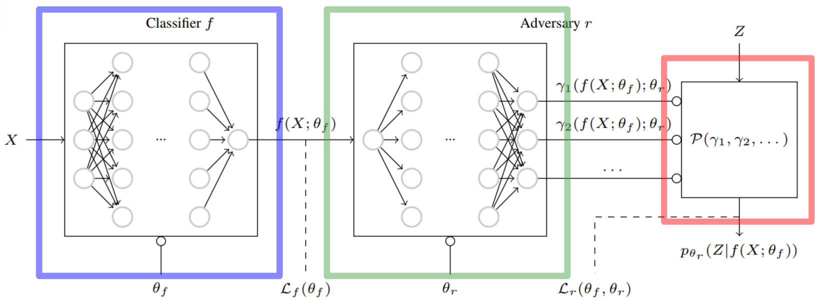
Presented solution

- 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

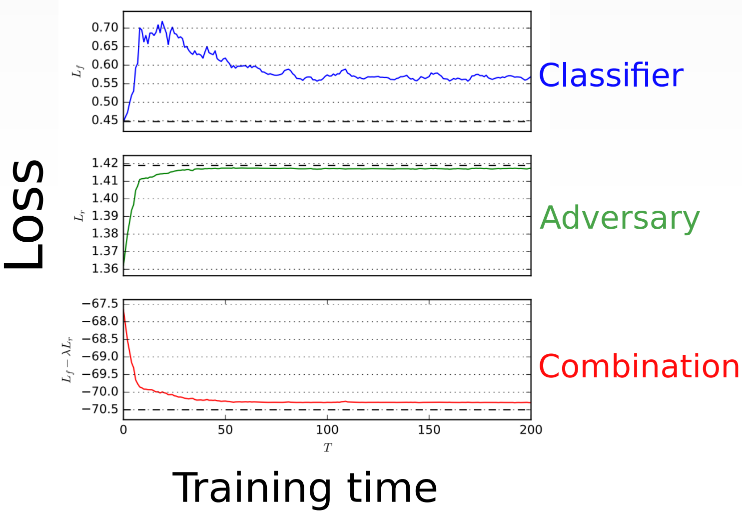
Adversarial neural network



(arXiv:1611.01046)

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

Expected ANN losses



(arXiv:1611.01046)

Adversarial training

Approach 1

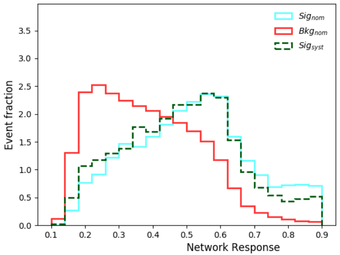
Run 1

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

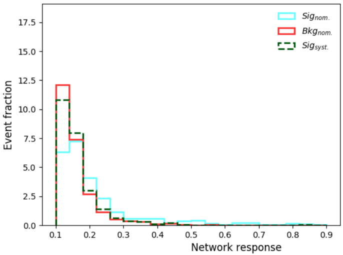
Setup

- Input: Classifier output
- Hidden layers: 3(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: 400 iterations
- $\lambda = 10$

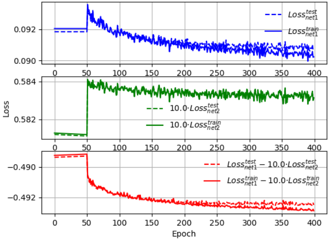
Approach 1 - Run 1 - Results



Simple



Adversarial



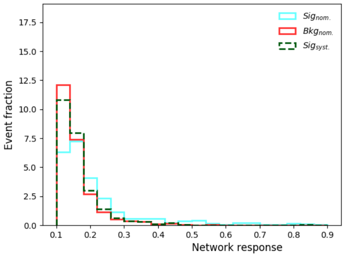
- Inferior separation
- Only combined losses depend on λ
- Systematics drawn to background

Approach 1 - Optimisation

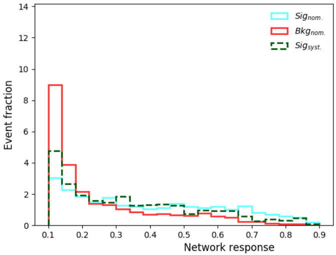
Setup of the adversary

- Input: Classifier output
- 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 iterations
- $\lambda = 0.1$

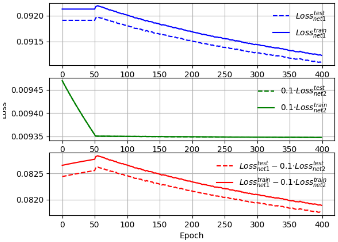
Approach 1 - Run 2 - Results



Run 1

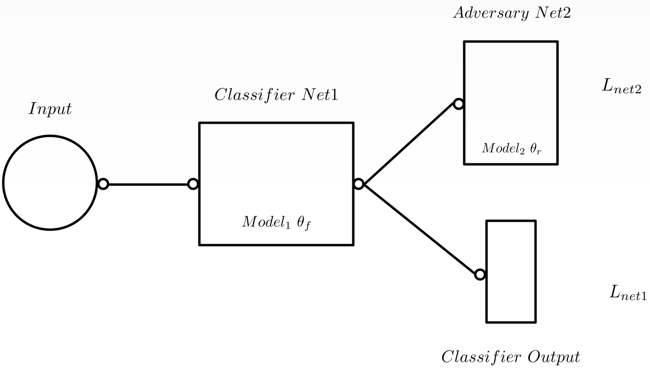


Run 2



- Improved separation
- Slightly improved sensitivity
- Only combined losses depend on λ

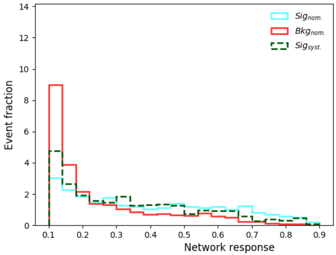
Approach 2



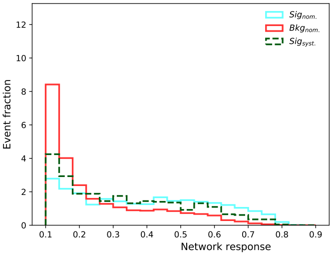
Input the last hidden layer to the adversary

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

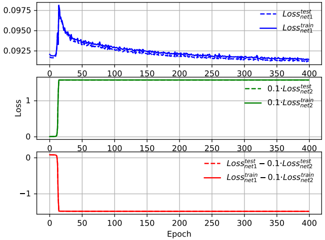
Approach 2 Results



Approach 1



Approach 2



- Even better separation
- Still visible disagreement
- Losses behave as expected

Approach 3

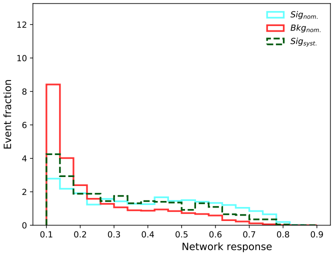
Assumption

Dimension of the hidden layer is too high

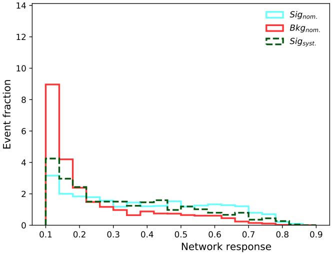
Easy workaround

- Add an intermediate layer
- Reduced to 16 nodes

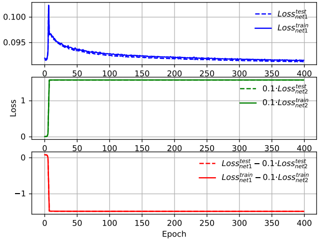
Approach 3 Results



Approach 2



Approach 3



- No clear further improvement
- Higher stability

Conclusions

Network performance

- Classic approach does not deliver expected results
- Changing the input results in slight improvements on the sensitivity

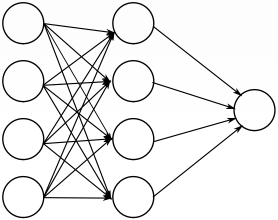
Insights

- Classifier and adversary need to be trained as a combination
- The complex architecture of the adversary is not justified by just inputting the classifier's output
- Learning rate is a the central hyper-parameter for the loss behaviour

https://xkcd.com/1838/

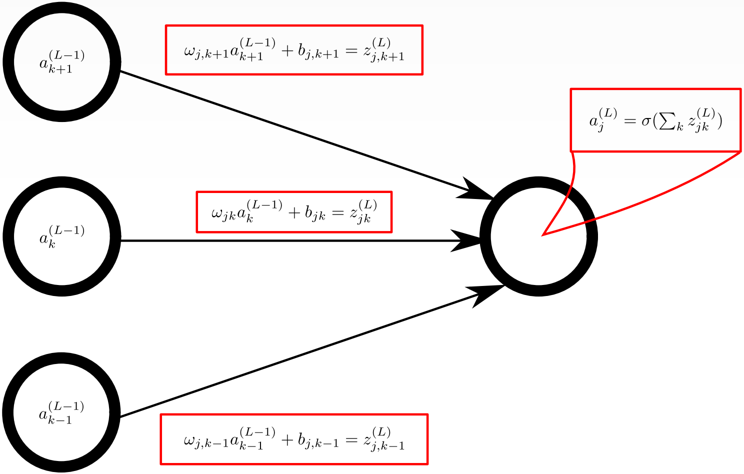
Neural Networks

- 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

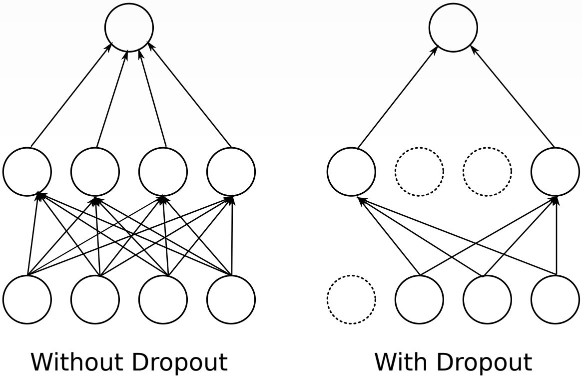
$$\nu' = \alpha\nu - \epsilon\frac{1}{m}\nabla_\theta \sum_j L(f(\hat{y}^j;\theta), y^j) \tag{3}$$

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

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

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

Dropout



Caption

Technical details

- The neural networks were built using Keras [1] with tensorflow [3] as backend

Variables

Variable
$p_{\mathrm{T}}^{\mathrm{miss}}$
$\lvert\eta_{jet1}\rvert$
$\lvert\eta_{jet1}\rvert$
$\lvert\eta_{lep1}\rvert$
$\lvert\eta_{lep2}\rvert$
$p_{\mathrm{T}\,jet1}$
$p_{\mathrm{T}\,jet2}$
$p_{\mathrm{T}\,lep1}$
$p_{\mathrm{T}\,lep2}$
ϕ_{jet1}
ϕ_{jet2}
ϕ_{lep1}
ϕ_{lep2}

Simple kinematic variables

Variable
$m_{lep1} + m_{jet2}$
$p_{\mathrm{T}\,lep1} + p_{\mathrm{T}\,lep2} + p_{\mathrm{T}}^{\mathrm{miss}}$
$p_{\mathrm{T}\,jet1} + p_{\mathrm{T}\,jet2}$
$m_{lep1} + m_{jet1}$
$p_{\mathrm{T}\,lep1} - p_{\mathrm{T}\,jet1}$
$R_{lep1} - R_{jet2}$
$R_{lep1,lep2} - R_{jet2}$
$m_{lep2} + m_{jet1}$
$p_{\mathrm{T}\,jet2}$
$R_{lep1} - R_{jet1}$
$R_{lep1} - R_{jet1}$
$R_{lep2} - R_{jet2}$
$\mathrm{Centrality}_{lep2} + \mathrm{Centrality}_{jet2}$
$R_{lep2} - R_{jet1}$

Complex variables