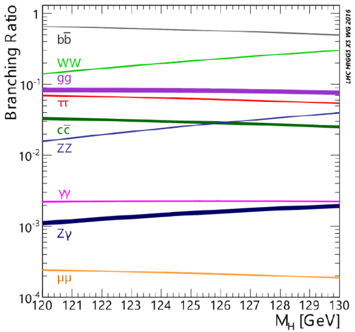
Neural network development in the analysis of single top-quark production in association with a Higgs boson and light-quark at ATLAS

Christian Kirfel

15th of March 2021

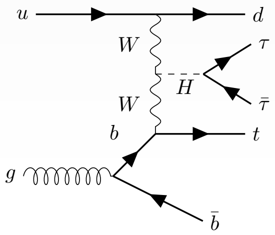
The tHq lepditau channel



[1]

- Relatively high branching ratio
- Hadronically decaying taus are more difficult to select than leptonic ones

tHq lepditau channel selection



- Number of jets: 2
- Number of b-jets: 1
- number of leptons: $1e/\mu$
- number of taus: 2 hadronic taus
- $E_{\mathrm{T,miss}}$: no cut (to 800 GeV)

- Jets:
  - $p_T > 35$ GeV
  - $|\eta| < 4.5$
- Electrons:
  - $p_T > 20$ GeV leading 27 GeV
  - $|\eta| < 2.5$ not in 1.37 - 1.52
- Muons:
  - $p_T > 20$ GeV leading 27 GeV
  - $0.01 < |\eta| < 2.5$
- Taus:
  - $p_T > 20$ GeV leading 27 GeV
  - $|\eta| < 2.5$ not in 1.37 - 1.52

Challenges

Different sizes of background processes

- Dominating background can diminish the significance of secondary backgrounds in training
- Especially very signal-like backgrounds are likely to be mislabeled

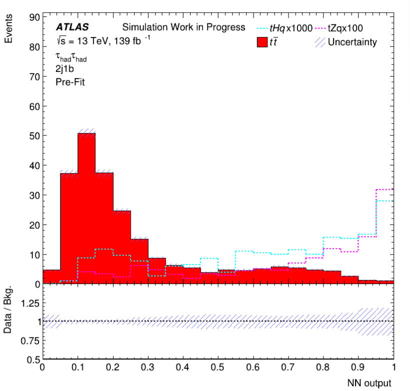
Negative weights in Monte Carlo

- Negative weights are needed in Monte Carlo generation to avoid double counting.
- In neural network training negative weights lead to an unwanted behaviour.

Accelerating network optimisation

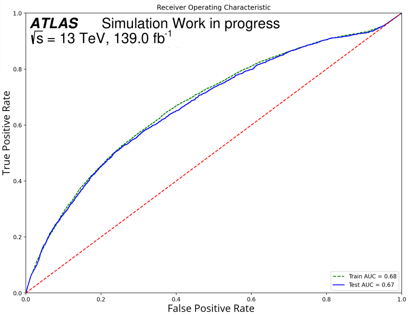
- Exploration of new features is only possible in an optimised network
- An evolutionary optimisation approach minimises the work effort

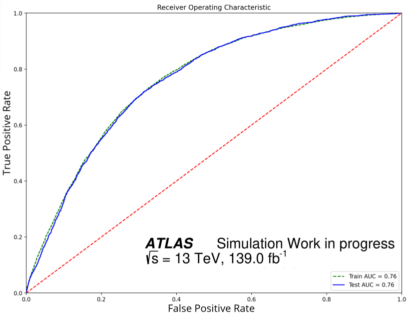
Background processes in neural network training



- tHq against other processes
- $t\bar{t}$ dominates the training
- tZq misclassified as signal
- Possible approaches:
  - multiple networks
  - multiple targets
  - reweighting samples

Impact of negative weights on the training





- About 35% negative weights
- Breaks the network training
- Possible approaches: absolute or just positive weights for training

Problems in network optimisation

Obstacles of optimisation processes

- Grid searches are both tedious and resource intensive
- Knowledge gained in one problem is rarely universal
- Experienced users can develop a bias towards certain hyperparameters

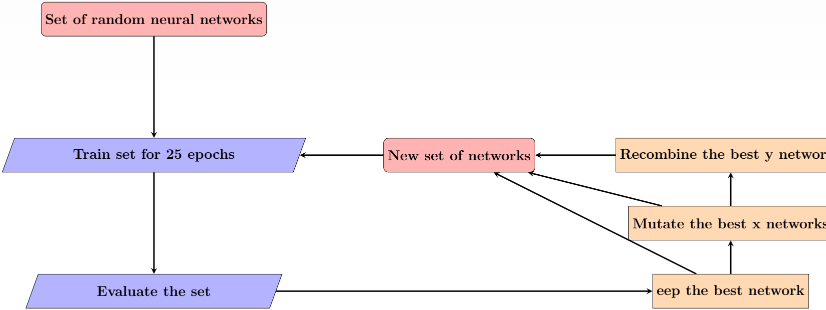
Applications of evolutionary optimisation

- A neural network should be developed in parallel to an ongoing analysis.
- New additions need a new optimisation.

Evolutionary optimisation of neural networks

- Combination of a grid searches with a survival of the fittest setup
- Start with a number of random hyperparamters
- Evaluate the set of hyperparameters
- Create a a new set of networks based on the previous best
- Add recombination and variation to avoid local minima and bias

test


Set of random neural networks
Train set for 25 epochs
Evaluate the set
New set of networks
Recombine the best y networ
Mutate the best x networks
eep the best network

Example of an evolutionary training

- Signal: $\mathrm{tH}q$
- Background: $\mathrm{t\bar{t}}$
- Testing: nodes, layers, dropout
- Fixed hyperparameters:
  - Optimizer: Adam
  - Activation: relu, sigmoid
  - Batchsize: 1000
  - Epochs per generation: 25

Initial parameters

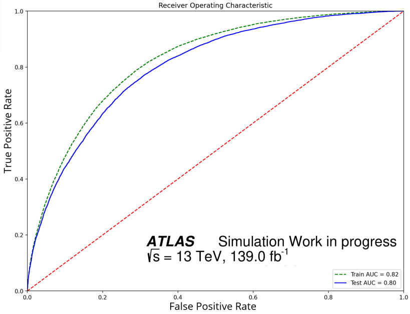
- Layers: $1 - 10$
- Nodes: $1 - 100$
- Dropout: $0 - 1$

Final parameters

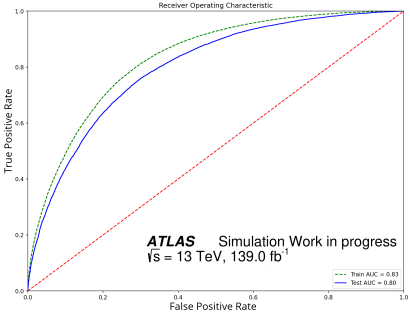
- Layers: $[2:6]$
- Nodes: $[30:90]$
- Dropout: $[0.1:0.7]$

Comparing ROC to a grid search

Computational ressources are comparable for both approaches. For the evolutionary optimisation a larger parameter space has been scanned.

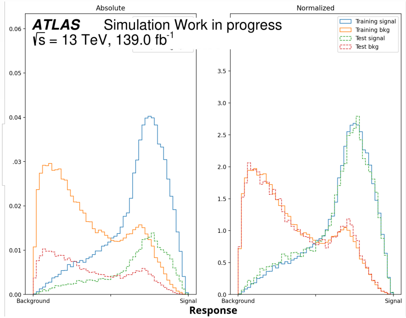


Grid search

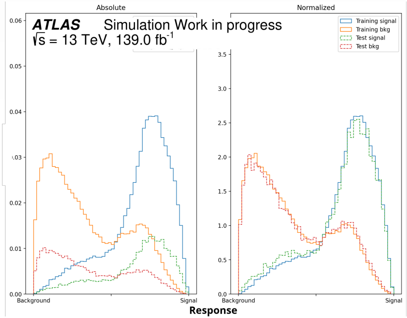


Evolutionary search

Comparing response to a grid search



Grid search



Evolutionary search

Conclusions

- Neural networks in the $\mathrm{tH}q$ channel is a challenging analysis.
- Negative weights resulting from Monte Carlo generators cannot easily be handled by machine learning algorithms.
- Different sizes of background samples result in smaller backgrounds getting ignored or even misclassified as signal.
- Evolutionary optimisation of neural networks is a nice approach to reoptimise a network automatically.