Deep learning in ATLAS $t\bar{t}H(\rightarrow b\bar{b})$ analysis

Yann Coadou

mostly reporting work from Ziyu GUO's thesis
in co-supervision with Thierry Artières, LIS/Ecole Centrale Marseille

CPPM Marseille

IN2P3/IRFU Machine Learning workshop
CC-IN2P3, 23 January 2020

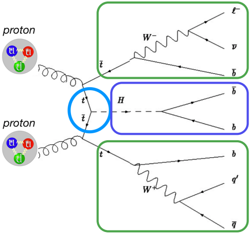
Depuis 80 ans, nos connaissances
bâtissent de nouveaux mondes

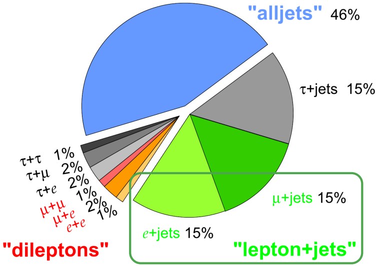
Source

- Ziyu GUO's PhD thesis: 2016–2019
  "Search for the Higgs boson in the $t\bar{t}H$ ($H \to b\bar{b}$) channel in the ATLAS experiment at the LHC using machine learning methods and synchronization of the ITk geometry description for simulation and radiation studies for the HL-LHC ATLAS upgrade"
- Inter-doctoral school grant at Aix-Marseille Université
- Collaboration between CPPM and Laboratoire Informatique et Systèmes (LIS) at AMU
- Co-supervision with Thierry Artières, LIS/Ecole Centrale Marseille
- Defended on 5 November 2019
- Manuscript and details: CERN-THESIS-2019-222

Search for the Higgs boson in $t\bar{t}H$ ($H \to b\bar{b}$)

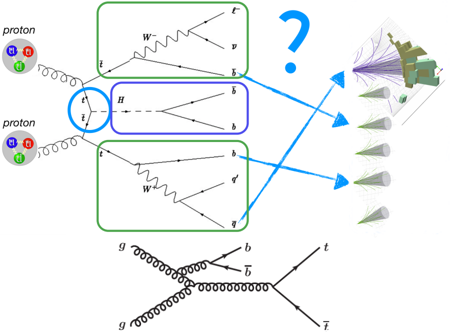
- $t\bar{t}H$ production: direct measurement of top Yukawa coupling
- Dominant decay mode: $H \to b\bar{b}$ with 58% branching ratio
- Single-lepton channel: large statistics and lepton signature

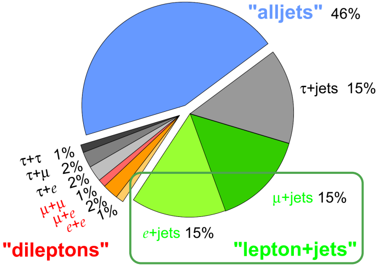




Search for the Higgs boson in $t\bar{t}H$ ($H \to b\bar{b}$)

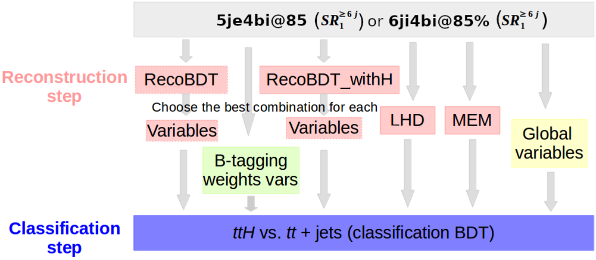
- Rare $t\bar{t}H$ signal production w.r.t. main $t\bar{t}+$ jets background
- Hard to reconstruct:
  - multiple jets/b-jets in final state
  - limited b-tagging efficiency
  - ambiguity to associate jets to initiating quarks or gluons
- Large theoretical uncertainties in $t\bar{t}$ + jets Monte Carlo modeling





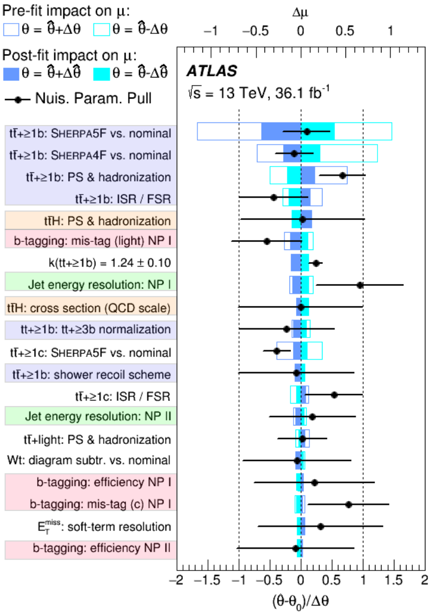
Using BDT for reconstruction and classification

- **Reconstruction step:** solve ambiguity between jets and partons
  - Reco BDT: pick jet combination with highest BDT score as correct matching (trained on correct/wrong combinations in $t\bar{t}H$ sample)
  - Likelihood discriminant (LHD): probability distribution function under $t\bar{t}H/t\bar{t}$ hypotheses using 1D variable distributions from all possible combinations
  - MEM: exploit full matrix element calculation



- **Classification step:** use information from all reconstruction MVAs + event level variables

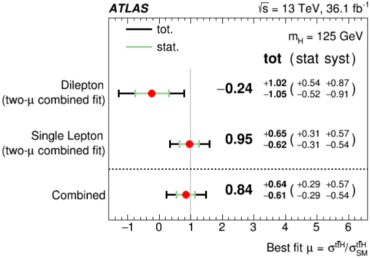
Systematic uncertainties

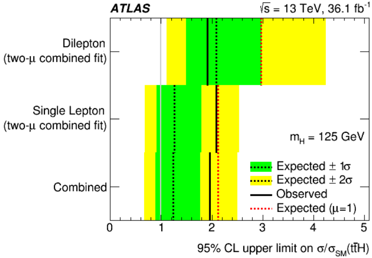


Sensitivity driven by systematic uncertainties

- Most dominant systematic sources: $t\bar{t} + \geq 1b$ modelling
  - Differences between generators
- Sub-leading source: low statistics of MC samples
- Other important uncertainties:
  - $t\bar{t}H$ modeling
  - b-tagging efficiency
  - Jet energy scale and resolution

- Combined fit across single- and di-lepton regions: $\mu = 0.84^{+0.64}_{-0.61}$
  - Dominated by single-lepton channel
- $t\bar{t}H$ excess significance: 1.4 σ observed (1.6 σ expected)





- Excluding $\mu > 2.0$ at 95% confidence level
- Results published in Phys.Rev.D **97** (2018) 072016

Rethinking $t\bar{t}H$ and $t\bar{t}$ classification

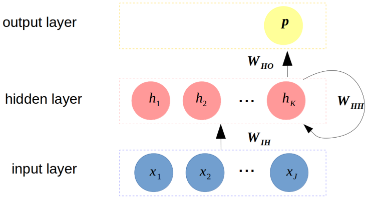
- Baseline MVA techniques: 2 steps, 3 algorithms
  - **Reconstruction step:**
    - Matrix Element Method
    - Likelihood: no variable correlations, using all combinations
    - Reconstruction BDT: exploiting variable correlations, only one combination
      - best combination only: limited truth matching fraction

best1	best2	best3	best4
30%	26%	14%	11%

  - **Classification BDT**: use info from reco MVAs and event-level variables to separate $t\bar{t}H$ and $t\bar{t}$
- Goal: end-to-end model to learn more information from inputs
  $\Rightarrow$ both variable correlations and more combinations

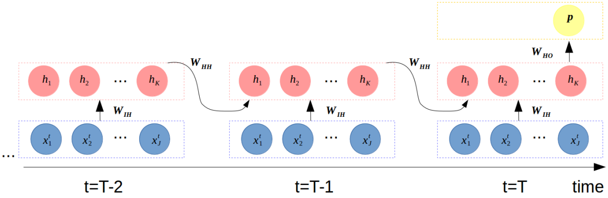
Using RNN for $t\bar{t}H$ and $t\bar{t}$ classification

- Recurrent neural networks (RNN) deal with variable-size sequence data
  - aggregate information: keeping information of earlier frames while seeing more of a sequence
  - e.g. popular in natural language processing



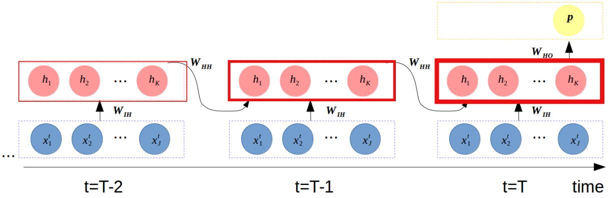
Using RNN for $t\bar{t}H$ and $t\bar{t}$ classification

- Recurrent neural networks (RNN) deal with variable-size sequence data
  - aggregate information: keeping information of earlier frames while seeing more of a sequence
  - e.g. popular in natural language processing

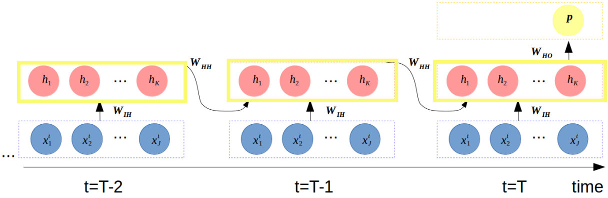


Using RNN for $t\bar{t}H$ and $t\bar{t}$ classification

- Recurrent neural networks (RNN) deal with variable-size sequence data
  - aggregate information: keeping information of earlier frames while seeing more of a sequence
  - e.g. popular in natural language processing
- Long-term dependence issue: early frames do not impact weight update very much



- Recurrent neural networks (RNN) deal with variable-size sequence data
  - aggregate information: keeping information of earlier frames while seeing more of a sequence
  - e.g. popular in natural language processing
- Long-term dependence issue: early frames do not impact weight update very much



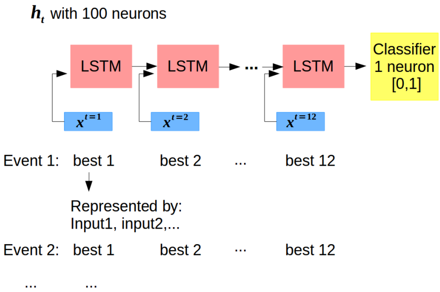
- Long short-term memory (LSTM), a variation of RNN
  - using gates to regulate information flow
  - can also use Gated Recurrent Unit (GRU), similar performance here

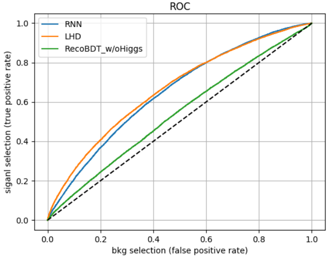
RNN sequence input

- Event = sequence, combinations = frames, sorted by recoBDT score

LHD: all combs, ✓ Higgs, ✓ b-tagging

RNN: 3 combs, ✗ Higgs, ✗ b-tagging

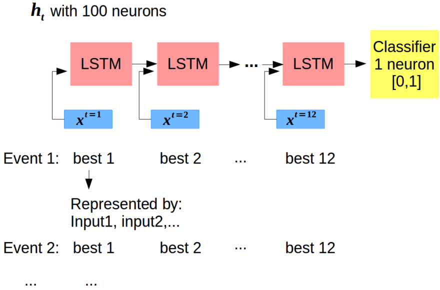


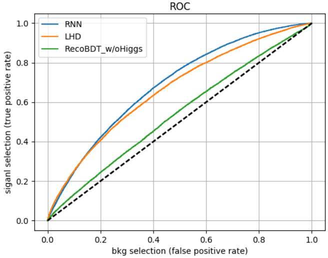


RNN sequence input

- Event = sequence, combinations = frames, sorted by recoBDT score

LHD: all combs, ✓ Higgs, ✓ b-tagging
RNN: 12 combs, ✗ Higgs, ✗ b-tagging

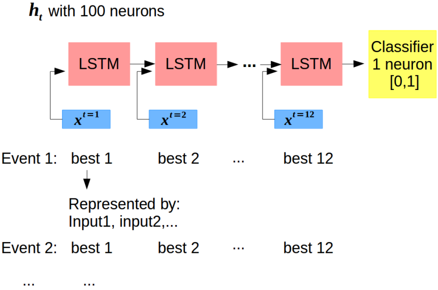




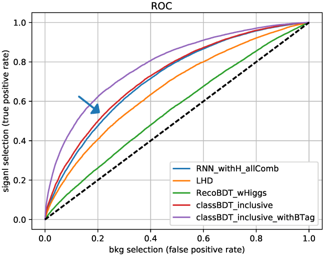
- Fixing sequence length to 12
  - ≥12 combinations (=12 in 6je4bi@85%)
  - Performance improved from 3 to 12
  - No impact of changing ordering

RNN sequence input

- Event = sequence, combinations = frames, sorted by recoBDT score



BDT: reco MVAs, ✓ Higgs, ✗ *b*-tagging
RNN: 12 combs, ✓ Higgs, ✗ *b*-tagging



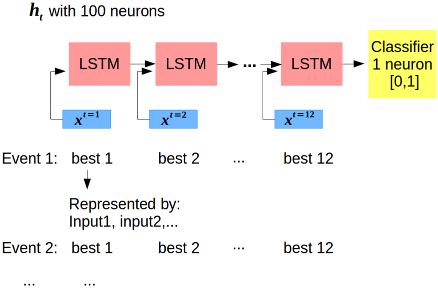
- Fixing sequence length to 12
  - $\geq$12 combinations (=12 in 6je4bi@85%)
  - Performance improved from 3 to 12
  - No impact of changing ordering
- Similar input to classification BDT, w/o LHD and MEM
  - Global kinematics, reco BDT inputs with Higgs info

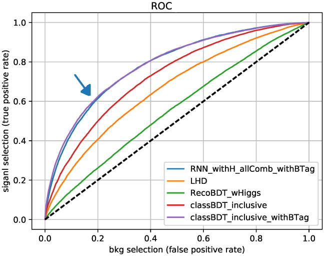
RNN sequence input

- Event = sequence, combinations = frames, sorted by recoBDT score

BDT: reco MVAs, ✓ Higgs, ✓ *b*-tagging

RNN: 12 combs. ✓ Higgs. ✓ *b*-tagging



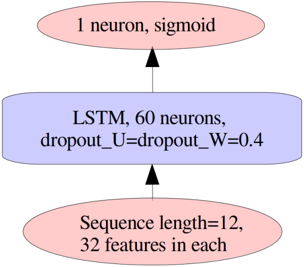


- Fixing sequence length to 12
  - ≥12 combinations (=12 in 6je4bi@85%)
  - Performance improved from 3 to 12
  - No impact of changing ordering
- Similar input to classification BDT, w/o LHD and MEM
  - Global kinematics, reco BDT inputs with Higgs info
  - 6 jets *b*-tagging scores

RNN performance

- Hyper-parameter optimization with tree-structured Parzen estimators (TPE)
- Same inputs as classification BDT

BDT	un-optimized RNN	optimized RNN
0.789	0.788	0.790

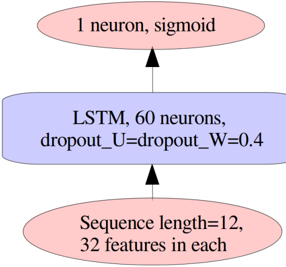


RNN performance

- Hyper-parameter optimization with tree-structured Parzen estimators (TPE)

- Same inputs as classification BDT

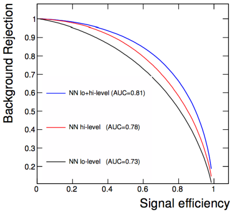
BDT	un-optimized RNN	optimized RNN
0.789	0.788	0.790

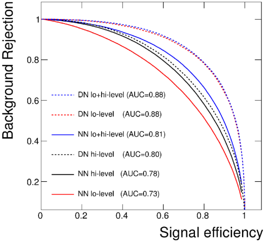


- RNN performs as good (or slightly better) as the two-step MVAs
  - Without using LHD and MEM as for BDT
- Solves reconstruction and classification in one step, using both correlations and combinations

Using low-level features as input variables

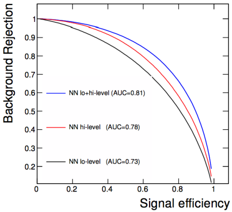
- Previous studies using simplified simulation have shown DNN + low-level features surpass shallow networks using high level features ▸ arXiv: 1402.4735

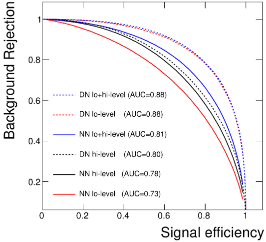




Using low-level features as input variables

- Previous studies using simplified simulation have shown DNN + low-level features surpass shallow networks using high level features ▸ arXiv: 1402.4735





High-level input features (physics motivated)

Same features as the previous binary RNN model

Low-level input features

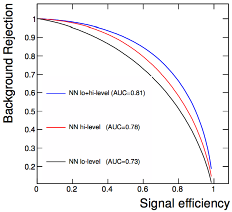
p_x, p_y, p_z, E and b-tagging of 8 objects: 6 jets + lepton and neutrino

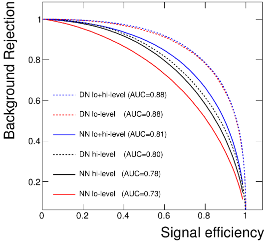
- DNN with best combination only
- RNN: 12 combinations

	AUC on test
DNN low level	.772
DNN high level	.787
RNN low level	.781
RNN high level	.790

Using low-level features as input variables

- Previous studies using simplified simulation have shown DNN + low-level features surpass shallow networks using high level features ▸ arXiv: 1402.4735





High-level input features (physics motivated)

Same features as the previous binary RNN model

Low-level input features

p_x, p_y, p_z, E and b-tagging of 8 objects: 6 jets + lepton and neutrino

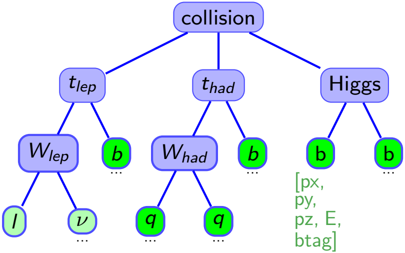
- DNN with best combination only
- RNN: 12 combinations
- Using low-level features gives worse performance

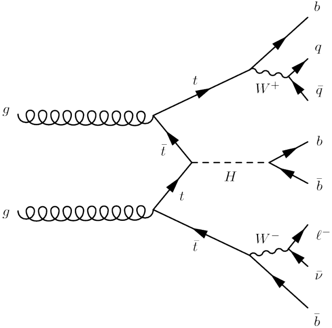
	AUC on test
DNN low level	.772
DNN high level	.787
RNN low level	.781
RNN high level	.790

Using physics domain knowledge inside the NN

Incorporate domain knowledge into NN design (inspired by arXiv: 1702.00748)

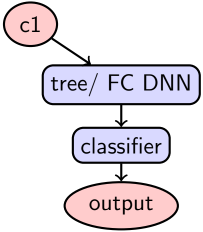
- Design a tree structure analogous to physical process (Feynman diagram)
- From leaves to the collision node, embed the low-level input space to another n-dimensional space
  - Leaves:
    - Input: for each jet, lepton and neutrino, $o = [px, py, pz, E, btag]$
  - Internal nodes:
    - Children nodes information summed through tree structure

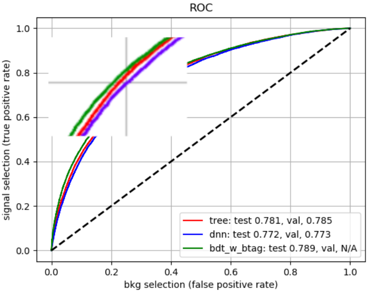




Using physics domain knowledge inside the NN

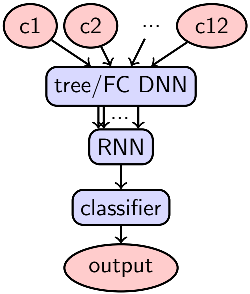
- Signal-like tree, using best combination only
- Or replace tree with FC DNN for comparison





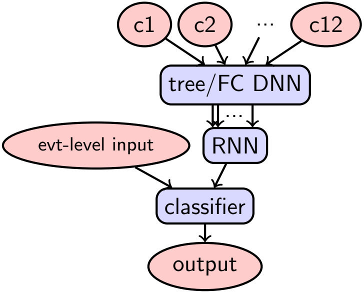
Using physics domain knowledge inside the NN

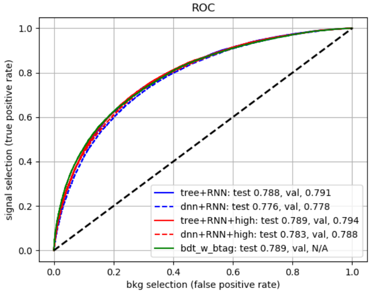
- Signal-like tree, using best combination only
- Or replace tree with FC DNN for comparison
- Use tree embedding for each combination, making up sequence input for RNN



Using physics domain knowledge inside the NN

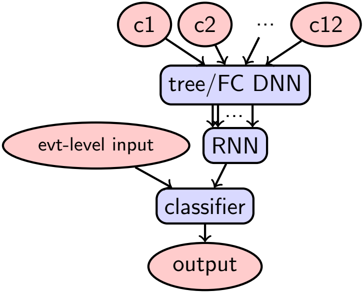
- Signal-like tree, using best combination only
- Or replace tree with FC DNN for comparison
- Use tree embedding for each combination, making up sequence input for RNN
- Also add in high-level inputs, used by BDT as well

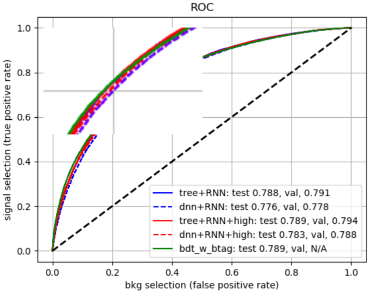




Using physics domain knowledge inside the NN

- Signal-like tree, using best combination only
- Or replace tree with FC DNN for comparison
- Use tree embedding for each combination, making up sequence input for RNN
- Also add in high-level inputs, used by BDT as well

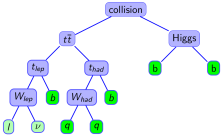


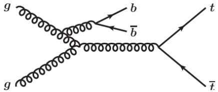


- Tree performance always better than regular DNN
  $\Rightarrow$ tree structure helps to learn from low level features

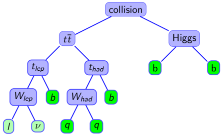
Tree mutations

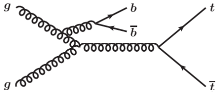
- Mutated tree structures to be more signal-like or $t\bar{t}$-like



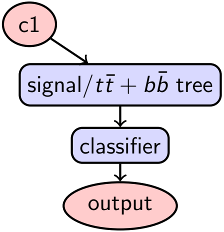


- Mutated tree structures to be more signal-like or $t\bar{t}$-like





- Using either signal or $t\bar{t} + b\bar{b}$-like tree and the best combination to separate $t\bar{t}H$ vs. $t\bar{t}$: small AUC difference

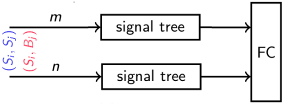


Models	AUC	
	test	val.
single tree + 1 FCC		
signal tree	0.781	0.785
$t\bar{t} + b\bar{b}$ tree	0.784	0.787

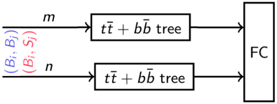
- $t\bar{t} + b\bar{b}$-like tree gives marginal improvement on $t\bar{t}$ events labeling 57.0% $\rightarrow$ 58.8%, deterioration on $t\bar{t}H$ events 77.4% $\rightarrow$ 76.1%

Siamese training: using two tree topologies

- Goal: exploit both signal- and $t\bar{t}+b\bar{b}$-like trees
- Siamese training: two trees with same architecture and shared weights
  - FC classifier: L1 distance between two events in embedding space
- signal-like tree model: (S_i, S_j) closer, (S_i, B_j) farther away
- $t\bar{t}+b\bar{b}$-like tree model: (B_i, B_j) closer, (B_i, S_j) farther away

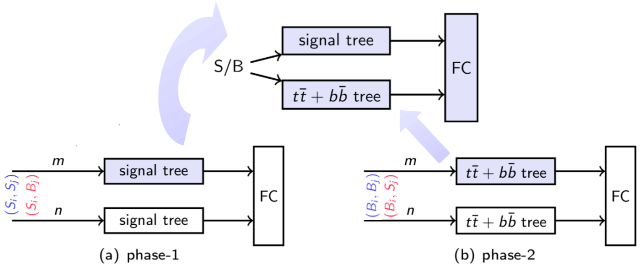


(a) phase-1



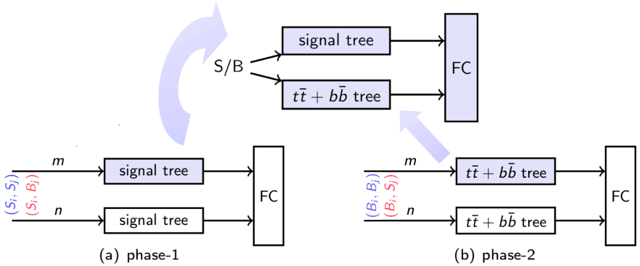
(b) phase-2

- Goal: exploit both signal- and $t\bar{t}+b\bar{b}$-like trees
- Siamese training: two trees with same architecture and shared weights
  - FC classifier: L1 distance between two events in embedding space
- signal-like tree model: (S_i, S_j) closer, (S_i, B_j) farther away
- $t\bar{t}+b\bar{b}$-like tree model: (B_i, B_j) closer, (B_i, S_j) farther away
- Transfer Siamese-trained trees into new binary classifier: $t\bar{t}H$ (S) vs. $t\bar{t}$(B)
  - Feed in one event each time: S or B
  - Concatenate trees + FCs



Siamese training: using two tree topologies

- Goal: exploit both signal- and $t\bar{t}+b\bar{b}$-like trees
- Siamese training: two trees with same architecture and shared weights
  - FC classifier: L1 distance between two events in embedding space
- signal-like tree model: (S_i, S_j) closer, (S_i, B_j) farther away
- $t\bar{t}+b\bar{b}$-like tree model: (B_i, B_j) closer, (B_i, S_j) farther away
- Transfer Siamese-trained trees into new binary classifier: $t\bar{t}H$ (S) vs. $t\bar{t}$(B)
  - Feed in one event each time: S or B
  - Concatenate trees + FCs

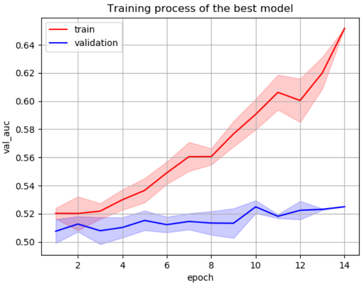


- Unfortunately: Siamese models lead to almost the same performance

Difference between nominal and syst. samples

- Dominant impact on final fit performance: $t\bar{t} + \geq 1b$ MVA shape difference of nominal and systematic samples

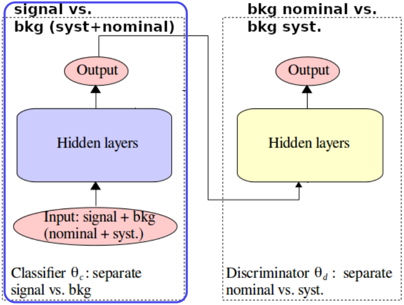
Models	AUC
trained on nominal only	
RNN nominal	0.790 ± 0.001
RNN syst.	0.787 ± 0.001
BDT nominal	0.788
BDT syst.	0.784
trained on nominal+syst.	
RNN nominal	0.785 ± 0.001
RNN syst.	0.785 ± 0.001



- Difference exists in the nominal and syst samples, but small (but quite large compared to $t\bar{t}H$ presence)
- Goal: train a classifier insensitive to the difference between nominal and systematic samples (following Learning to Pivot with Adversarial Networks)

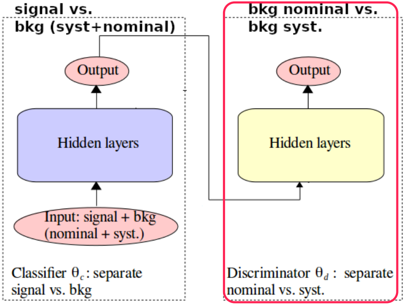
Adversarial training to reduce syst. uncertainties

- Idea: train a discriminator adversarially to constrain the classifier to have similar outputs (or representations) for nominal & systematic samples:



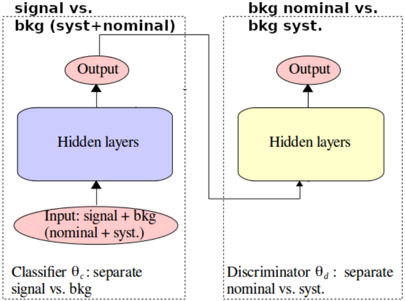
- Alternating training:
  - Train classifier, discriminator fixed:
    - Goal 1: $t\bar{t}H$ vs. $t\bar{t}$
    - Goal 2: fool discriminator to have nominal output close to systematic one

- Idea: train a discriminator adversarially to constrain the classifier to have similar outputs (or representations) for nominal & systematic samples:



- Alternating training:
  - Train classifier, discriminator fixed:
    - Goal 1: $t\bar{t}H$ vs. $t\bar{t}$
    - Goal 2: fool discriminator to have nominal output close to systematic one
  - Train discriminator, classifier fixed:
    - Goal: discriminate nominal vs. systematic samples

- Idea: train a discriminator adversarially to constrain the classifier to have similar outputs (or representations) for nominal & systematic samples:

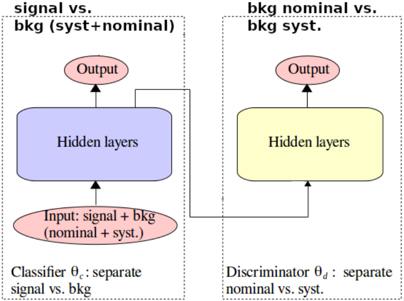


- Alternating training:
  - Train classifier, discriminator fixed:
    - Goal 1: $t\bar{t}H$ vs. $t\bar{t}$
    - Goal 2: fool discriminator to have nominal output close to systematic one
  - Train discriminator, classifier fixed:
    - Goal: discriminate nominal vs. systematic samples

- Repeated till discriminator cannot distinguish nominal from systematic

Adversarial training to reduce syst. uncertainties

- Idea: train a discriminator adversarially to constrain the classifier to have similar outputs (or representations) for nominal & systematic samples:



- Alternating training:
  - Train classifier, discriminator fixed:
    - Goal 1: $t\bar{t}H$ vs. $t\bar{t}$
    - Goal 2: fool discriminator to have nominal output close to systematic one
  - Train discriminator, classifier fixed:
    - Goal: discriminate nominal vs. systematic samples

- Repeated till discriminator cannot distinguish nominal from systematic
- Tuning hyper-parameters
  - also tried feeding discriminator with last hidden layer of classifier

Adversarial training to reduce syst. uncertainties

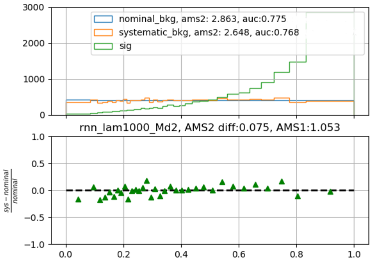
- Figure of merit: binned AMS1 ▸ HiggsML, significance depending on discriminant shape and uncertainty
- Improved AMS1 (with large uncertainty), decreased AUC, as expected
- BDT (trained on nominal only) AMS1: 0.752, AUC: 0.789

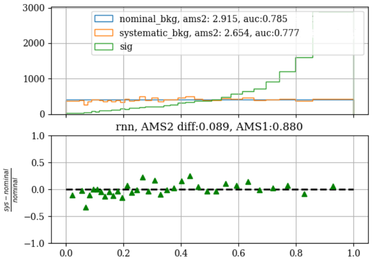
With adversarial training

	AUC	AMS1
nominal	0.771 ± 0.004	0.993 ± 0.189
syst.	0.762 ± 0.005	

Without adversarial training

	AUC	AMS1
nominal	0.784 ± 0.001	0.942 ± 0.149
syst.	0.778 ± 0.001	





- Unclear that it helps

Conclusion

- Played with deep learning in complex particle physics analysis in realistic setting
- Baseline BDTs: reconstruction and classification in two steps
- Replaced with LSTM with same high-level inputs $\Rightarrow$ similar performance in single step
- Using low level features instead $\Rightarrow$ not so good
- Introducing domain knowledge via parse trees:
  - recovers performance, using only low level features
  - with far fewer hyper-parameters
    $\Rightarrow$ even if no performance improvement, could mean rethinking of analysis optimisation (e.g., no variable list dependence)
- Adversarial training with pivot technique to decrease impact of systematics: not clear it helps here
- To keep in mind: BDTs are not dead yet!

Conclusion

- Played with deep learning in complex particle physics analysis in realistic setting
- Baseline BDTs: reconstruction and classification in two steps
- Replaced with LSTM with same high-level inputs $\Rightarrow$ similar performance in single step
- Using low level features instead $\Rightarrow$ not so good
- Introducing domain knowledge via parse trees:
  - recovers performance, using only low level features
  - with far fewer hyper-parameters

    $\Rightarrow$ even if no performance improvement, could mean rethinking of analysis optimisation (e.g., no variable list dependence)
- Adversarial training with pivot technique to decrease impact of systematics: not clear it helps here
- To keep in mind: BDTs are not dead yet!
- Note about collaboration with ML experts: think hard about publication policy beforehand