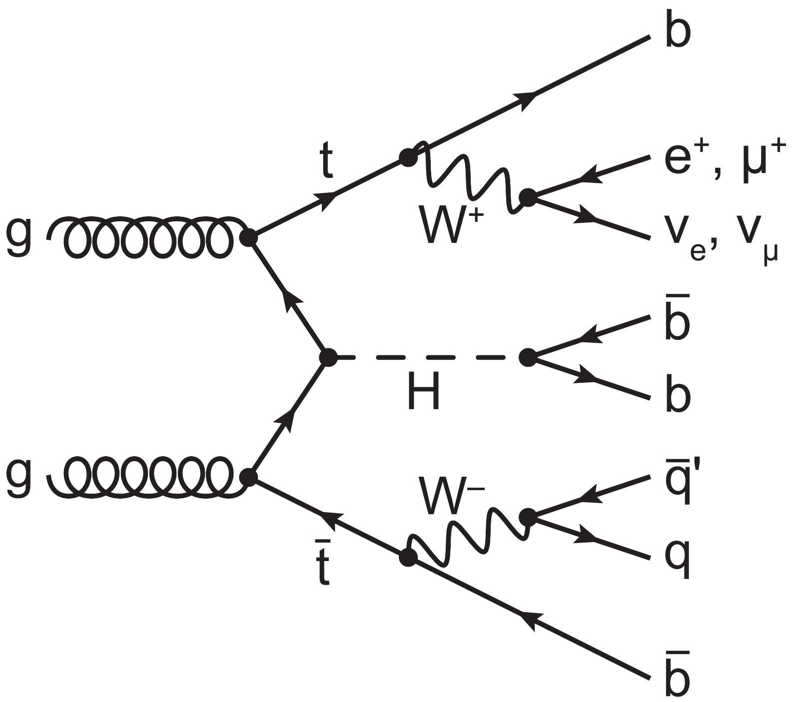
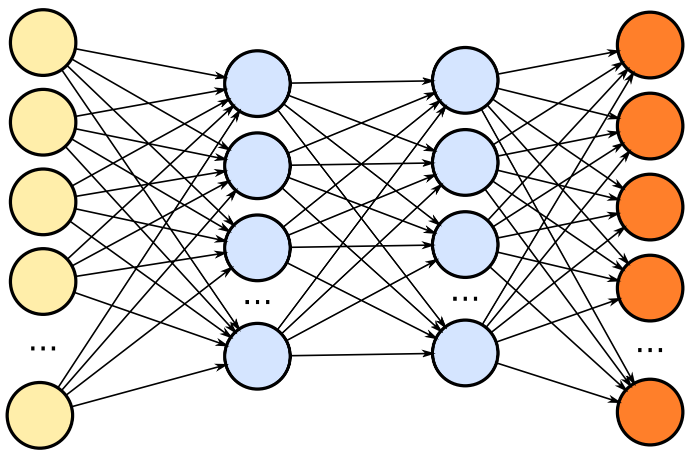


1. Event categorization
2. Neural network setup
3. Variable validation



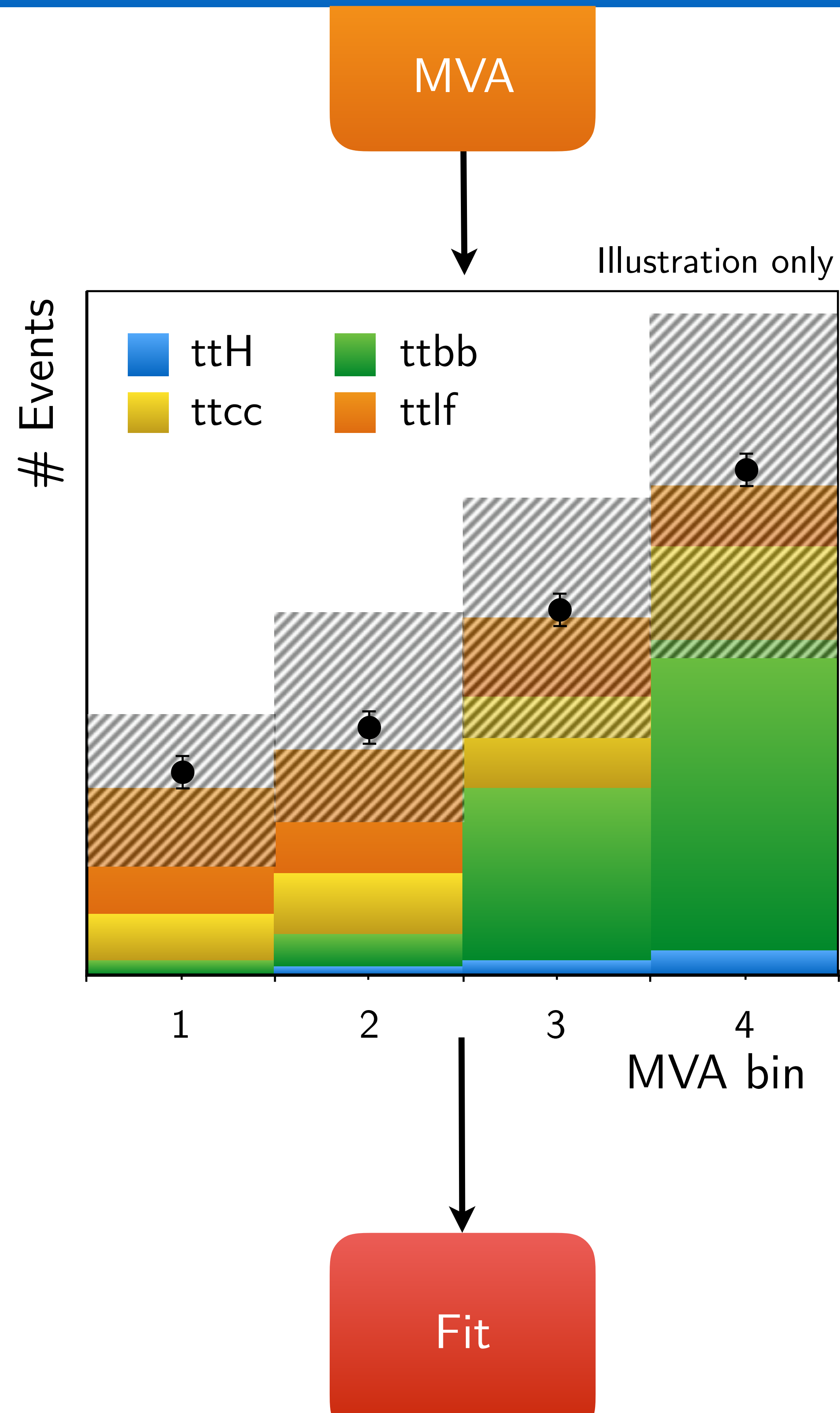
Event Categorization using Deep Neural Networks for the ttH (H→bb) Analysis at CMS

Marcel Rieger, Martin Erdmann, Yannik Rath

Higgs Toppings Workshop 2018

31 May 2018

Simultaneous fit & event categorization



- Fit expected yields (MC) to data simultaneously in all categories / bins

→ Signal extraction crucially depends on ability to measure backgrounds

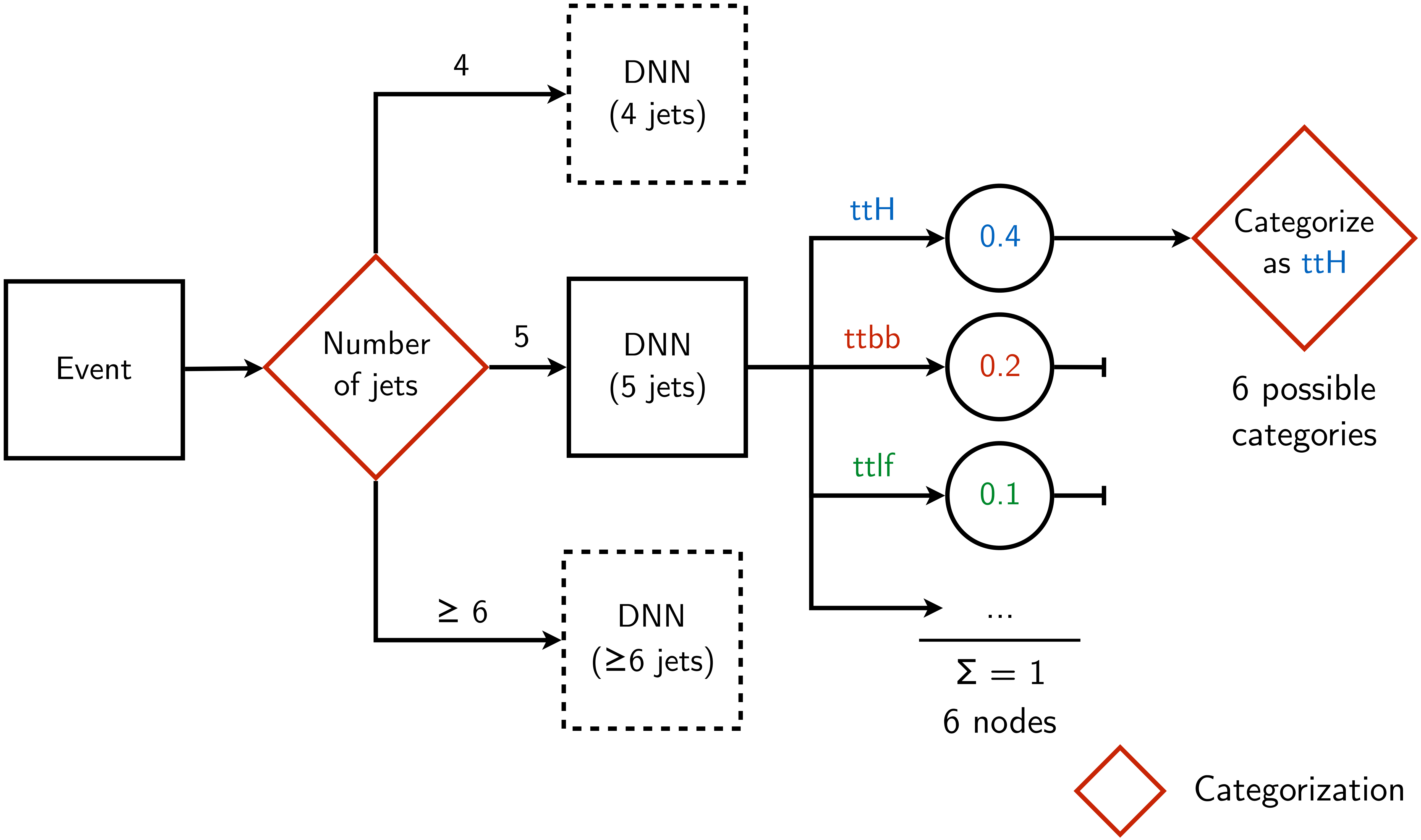
- *ttH* situation:
  - Large backgrounds (e.g. *ttlf*)
  - Irreducible backgrounds (e.g. *ttbb)*

  → Create enriched categories for **signal** and **each background** with DNNs

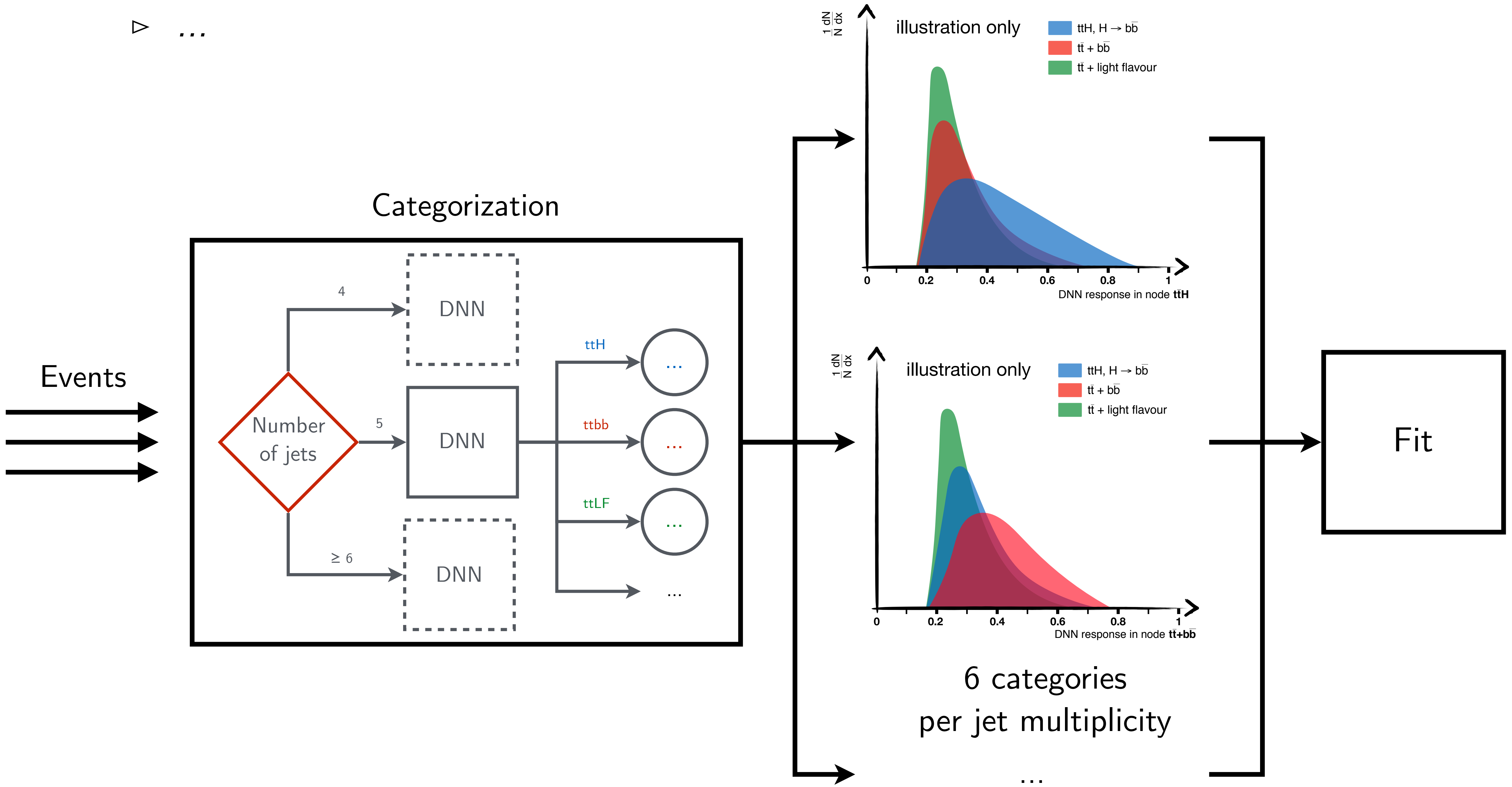
- See Matthias' talk

- b-tagging: $\varepsilon_b \cong 70\%$ → 4 b-tags found with only 25% probability

Per event

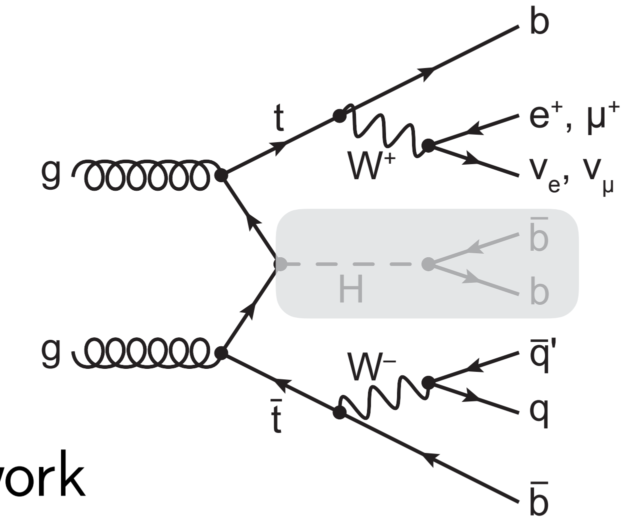

Event
Number of jets
4
5
≥ 6
DNN (4 jets)
DNN (5 jets)
DNN (≥6 jets)
ttH
ttbb
ttlf
0.4
0.2
0.1
...
Σ = 1
6 nodes
Categorize as ttH
6 possible categories
Categorization

- Output nodes of categorization networks yield powerful discriminators
  - → Use output node associated to category for simultaneous fit
    - ▹ *ttH* node for events categorized as *ttH*
    - ▹ *ttbb* node for events categorized as *ttbb*
    - ▹ ...



- Events of same class can have different topologies:
  - Jets out of acceptance
  - Merged jets
  - ...

  → Just training on bare event classes will confuse the network

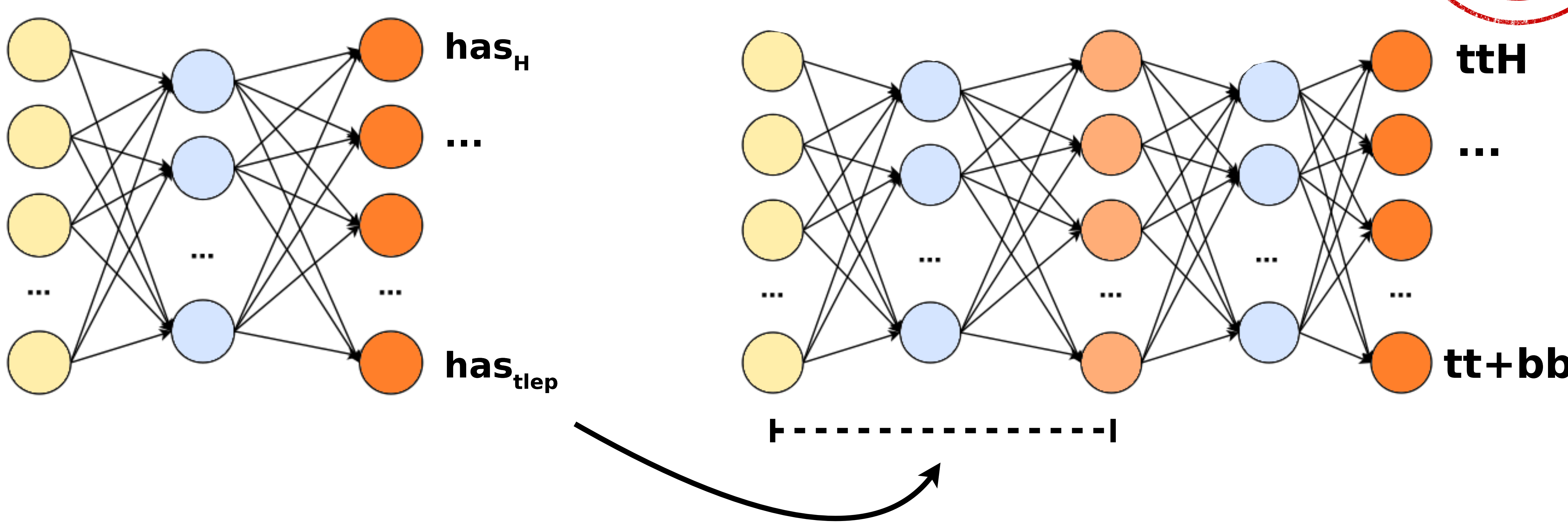


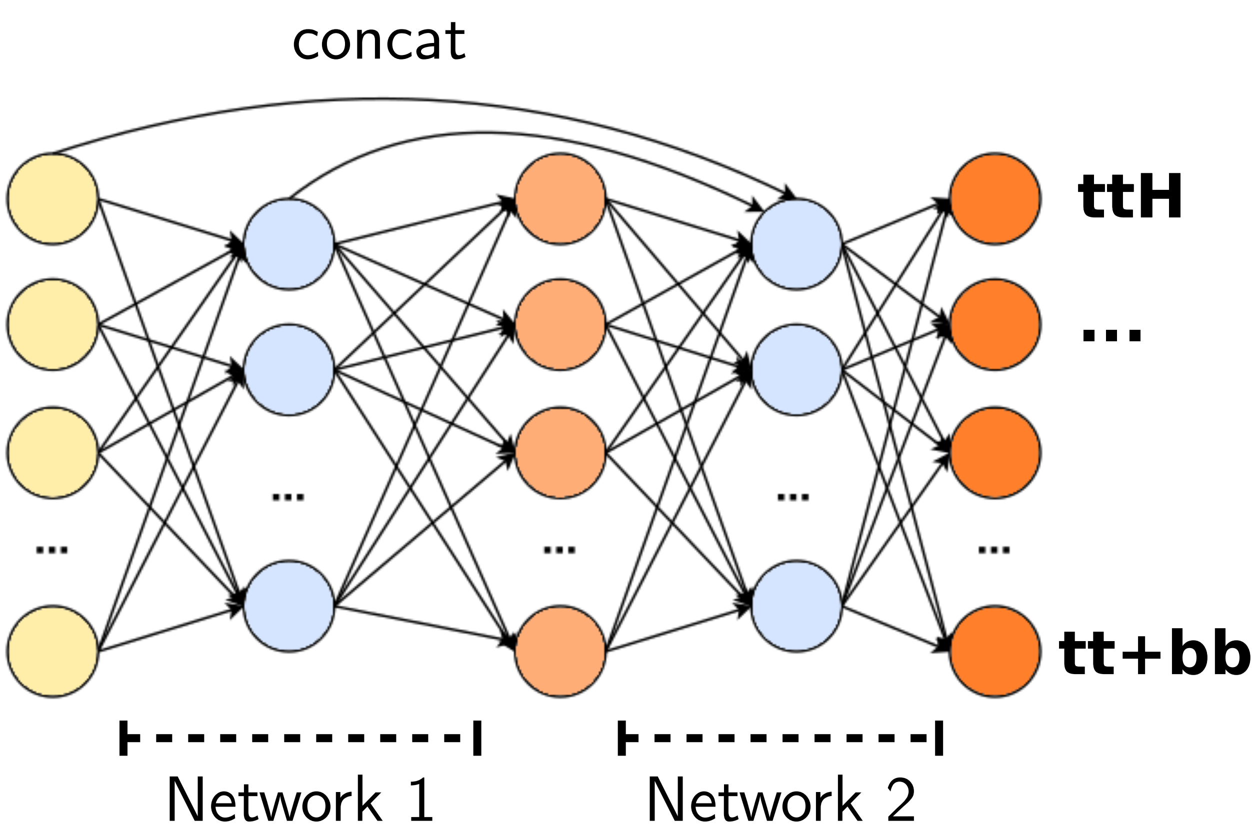
ttH or
ttbb?

- Idea:
  1. Pre-training on event content from generator % selection (has_H, has_{bH}, has_{blep}, ...)
  2. Extend network and train on actual classes

Pre-training

Extended training





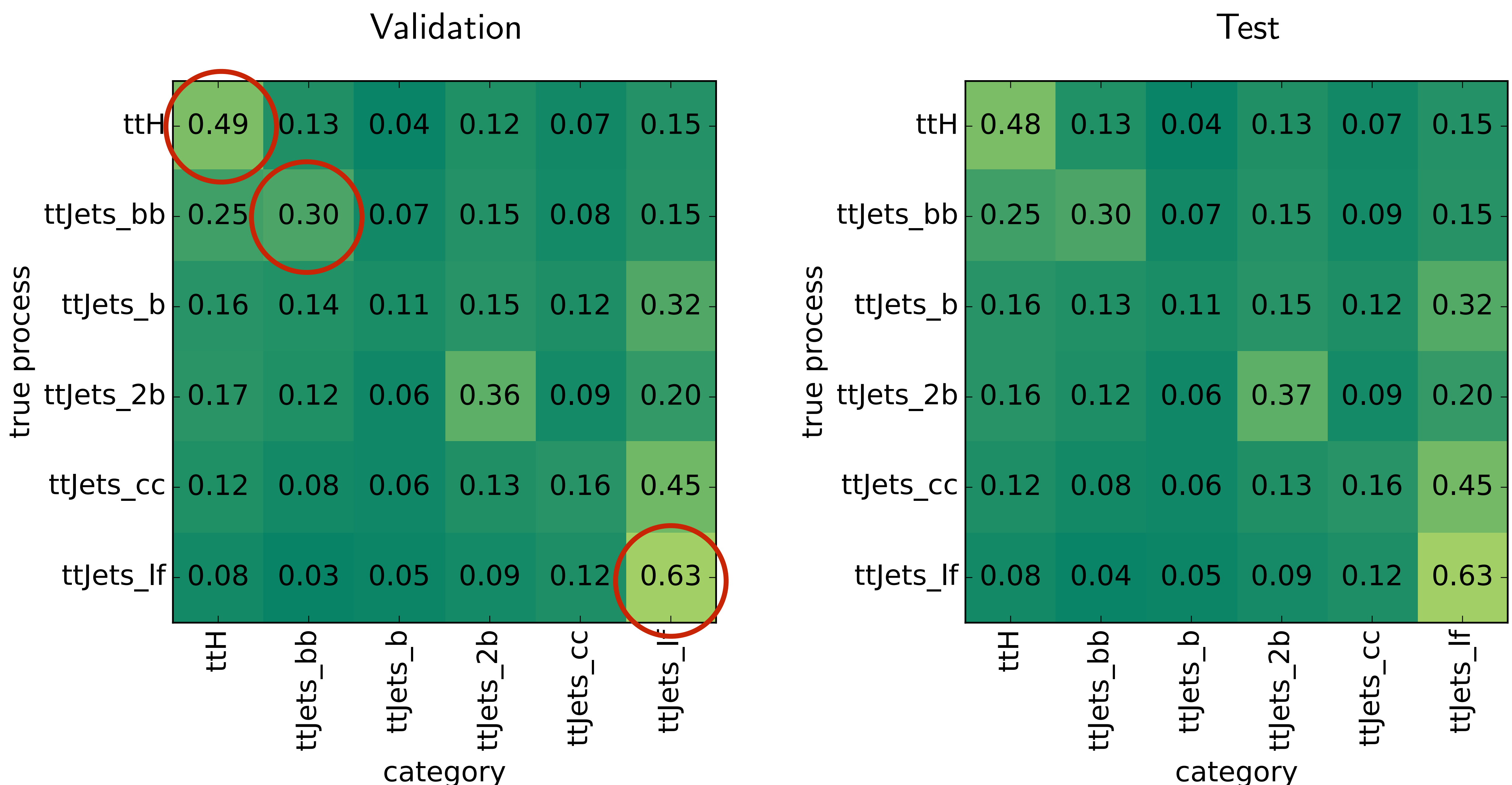
- Network architecture (≥ 6 jets):

Network 1	Network 2	Activation	L2	Dropout (keep prob.)	Learning rate (ADAM)
100,100	100,100	ELU	10^{-5}	0.7	10^{-4}

- Training time ~20 min on 980 Ti
- Implementation using plain TensorFlow

Better network architectures?
Maybe physics-motivated?

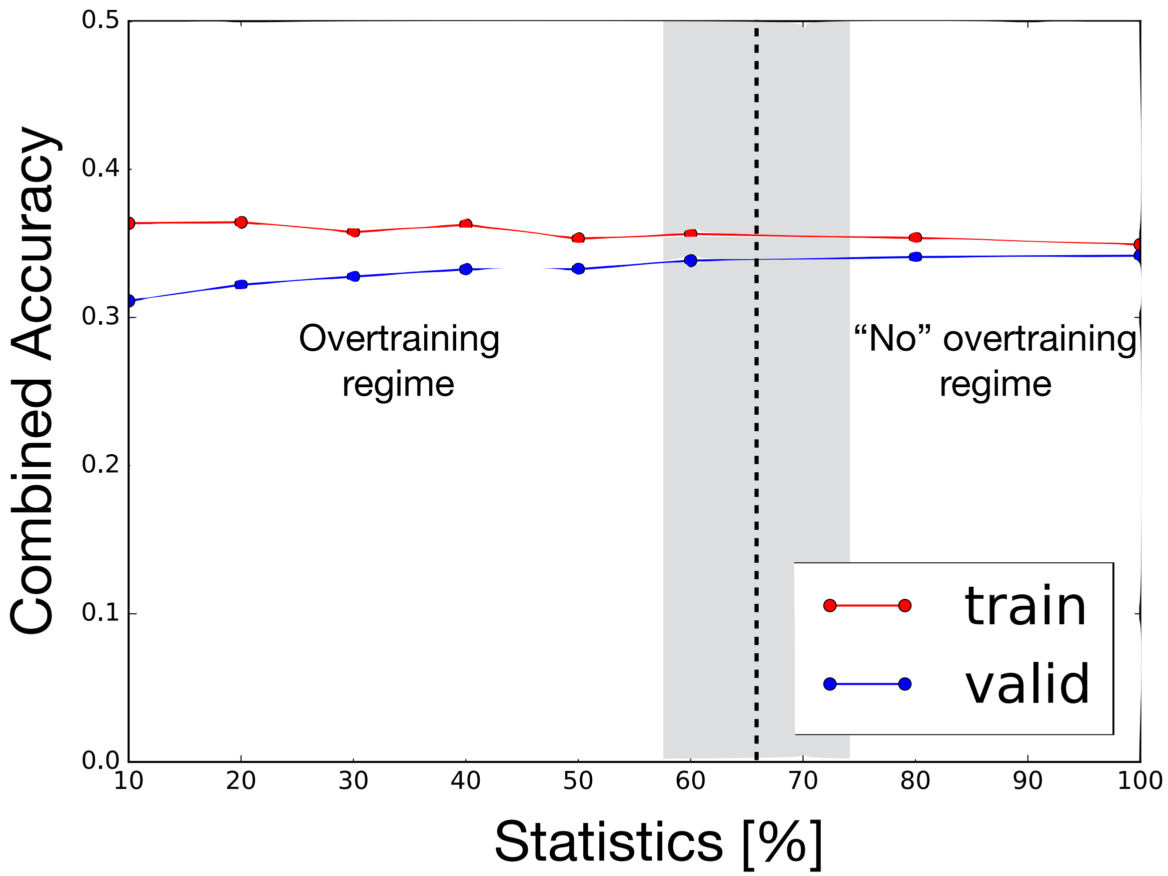
Classification accuracies



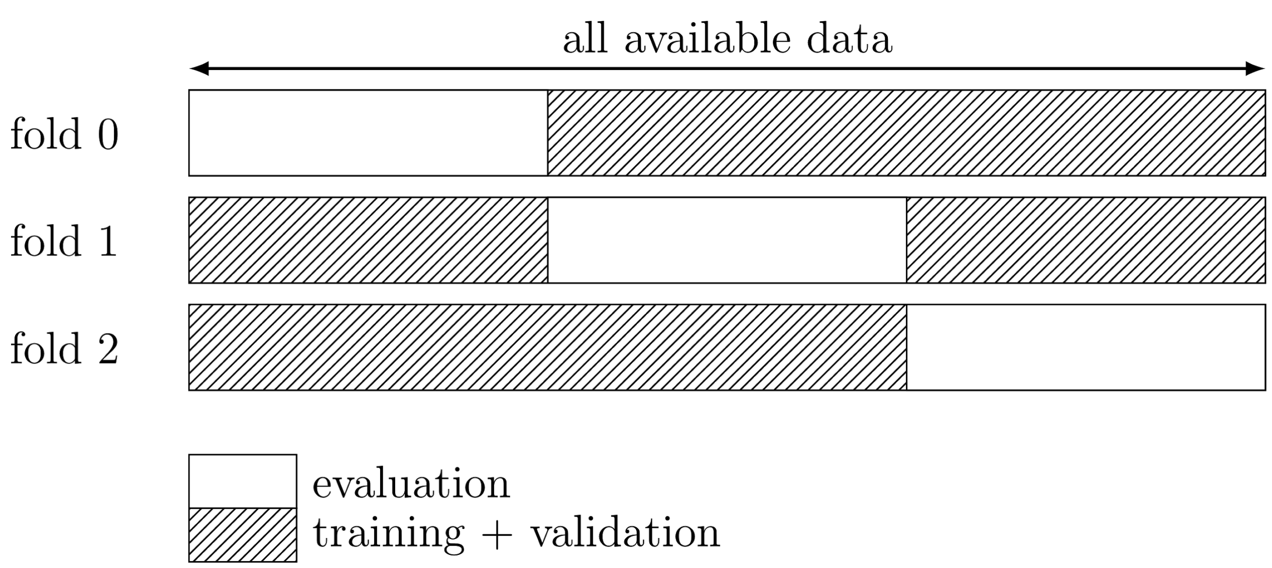
No signs of overtraining

Overtraining check and data handling

- Overtraining check:
  - Common trade-off:
    - → Network size **vs.** amount of data
  - Artificially force overtraining by reducing statistics
    - → Results stable down to ~⅔ of events



- Data handling:
  - ttH(bb): **50%** for analysis, **30%** training, **20%** validation (for optimization)
    - → Main constraint for network design
  - Alternative: n-fold cross validation



↓

cross evaluation (random for data)

MVA variable validation

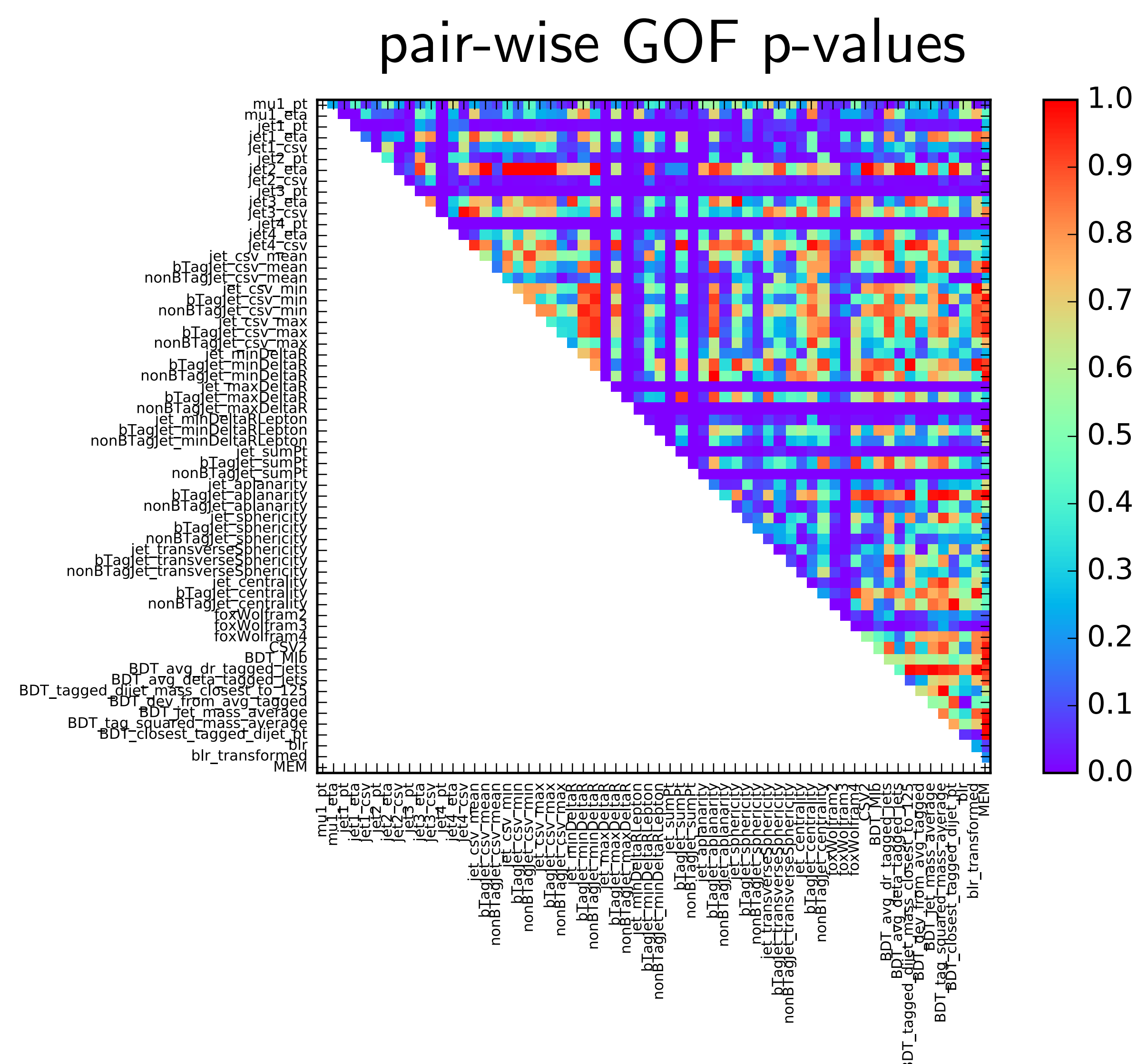
Marcel Rieger - 31.5.18

- Check of agreement between data and MC necessary, but 1D not sufficient
  - → MVA techniques exploit deep correlations
  - → Need to prove agreement of correlations in addition to 1D shapes

- Compare 2D correlation coefficients
  - → Mix low- and high-level variables to cover even deeper correlations

- Recipe:
  1. Create TH2F's for all pairs of input variables
  2. Determine goodness of fit p-value for data MC agreement (frequentist toys)
  3. Remove variables that yield a bad correlation agreement with other variables, criterion:

p-value $<$ 0.3 for $\geq$ 50% of variables



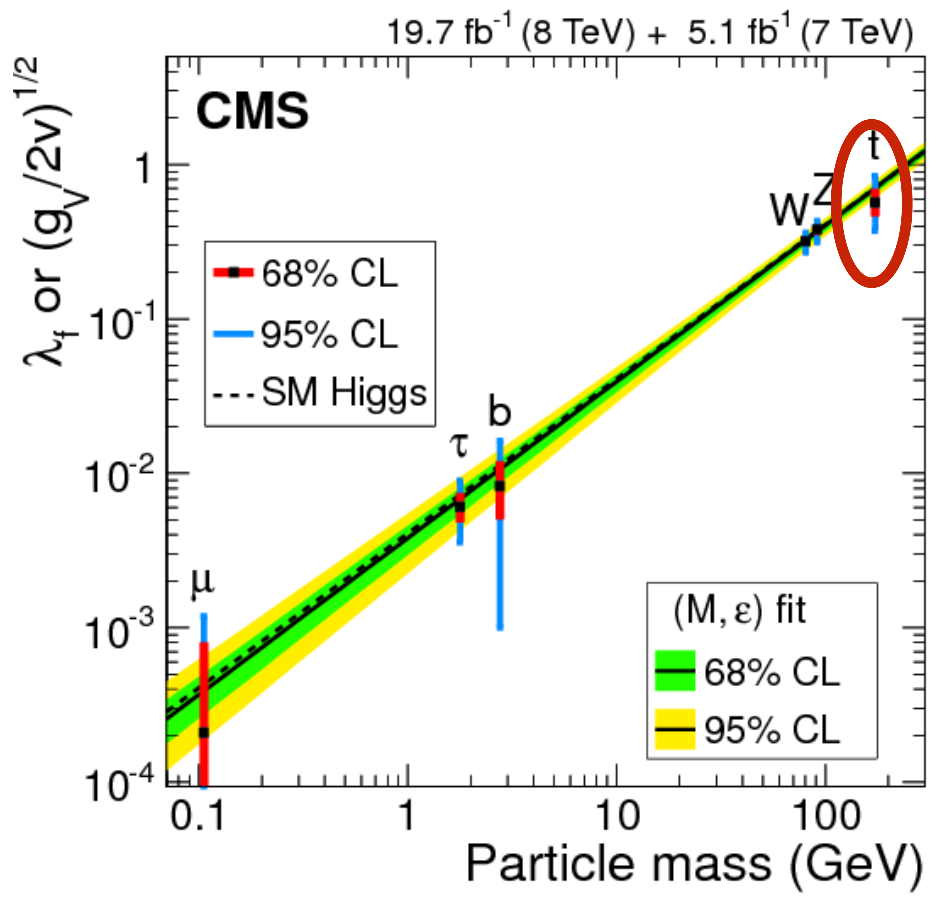
Backup

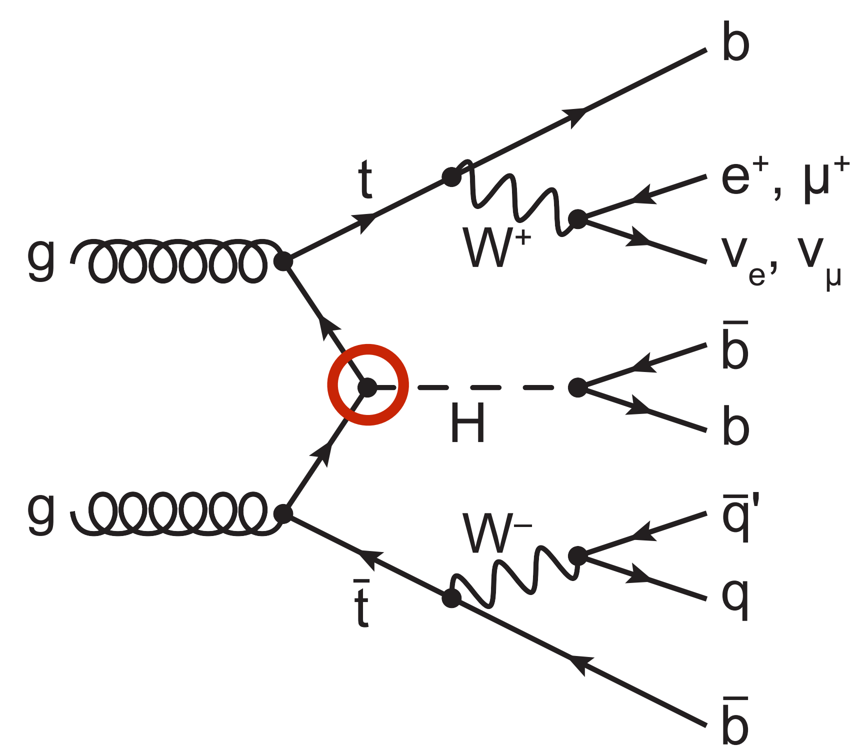
- *ttH*, $H \to bb$
  - Direct probe of top-Higgs coupling
  - Very rare, $\sigma_{ttH} = 0.51$ pb
  - $H \to bb$: largest BR (0.58)

- Backgrounds from *tt*+jets,
  - Esp. *ttbb* irreducible
  - Relatively large cross section uncertainties ($\gtrsim 35\%$)

- Complex final state
  - High combinatorics due to many jets

No "direct" measurement via (e.g.) mass peak
→ Simultaneous fit to MVA distribution(s)





- Used in CMS ttH(bb) leptonic analysis (CMS-HIG-PAS-17-026)
  - Results shown at Moriond 2018
  - Two methods:
    - DNN (with MEM as input variable)
    - Combination of BDT + MEM

From Moriond 2018

Channel & Analysis	$\mu \pm$ tot ($\pm$stat $\pm$ syst)
single lepton 2D BDT+MEM	$0.35^{+0.62}_{-0.62} \left({}^{+0.27}_{-0.27} \; {}^{+0.55}_{-0.55}\right)$
single lepton DNN	$0.84^{+0.52}_{-0.50} \left({}^{+0.27}_{-0.26} \; {}^{+0.44}_{-0.43}\right)$
primary result	$0.72^{+0.45}_{-0.45} \left({}^{+0.24}_{-0.24} \; {}^{+0.38}_{-0.38}\right)$

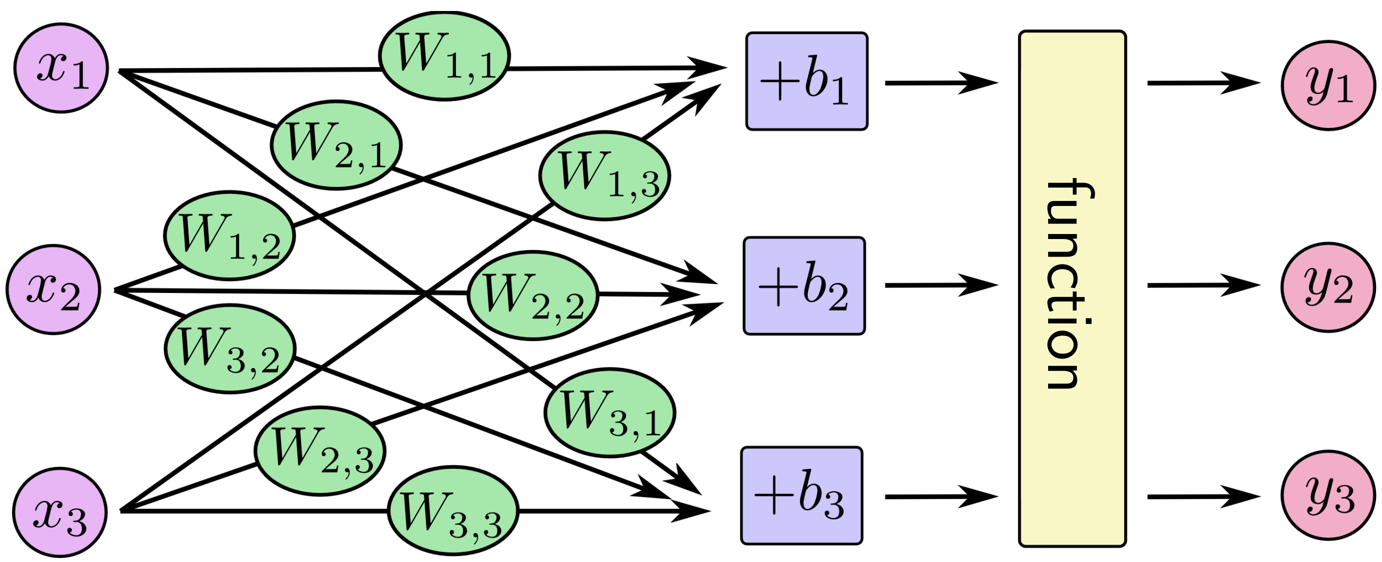
(excerpt)

"primary result": single lepton DNN & dilepton BDT+MEM

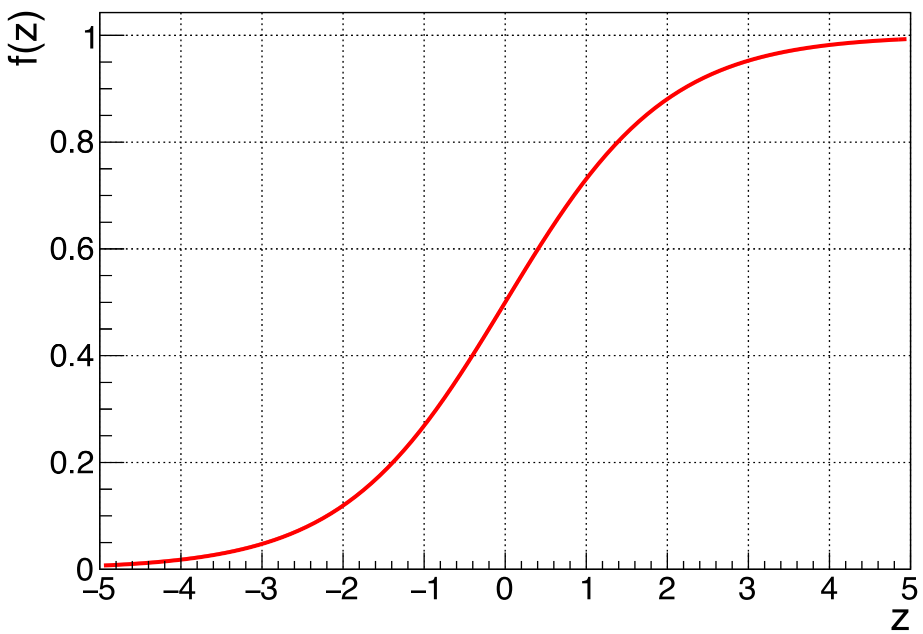
- Map input variables x to outputs y: $\vec{x} \rightarrow \vec{y} = \vec{D}(\vec{x}; \boldsymbol{W}, \vec{b}) \in \mathbb{R}^n$
  - D is the model which has to be defined
  - W and b are parameters, or weights, to be learned
  - n is the output dimension, BDT: 1, DNN: ≥ 1

- One layer network with logistic function f:

$$\vec{y} = f(\boldsymbol{W} \cdot \vec{x} + \vec{b}) \quad \text{with} \quad f(z) = \frac{1}{1 + e^{-z}}$$



tensorflow.org



- Plug *y* into *cost function*, compares to expected outputs y_{exp} (e.g. χ^2)
- Minimize costs using *gradient descent* algorithms

  Direction of minimization ≙ current slope

  Give feedback to weights

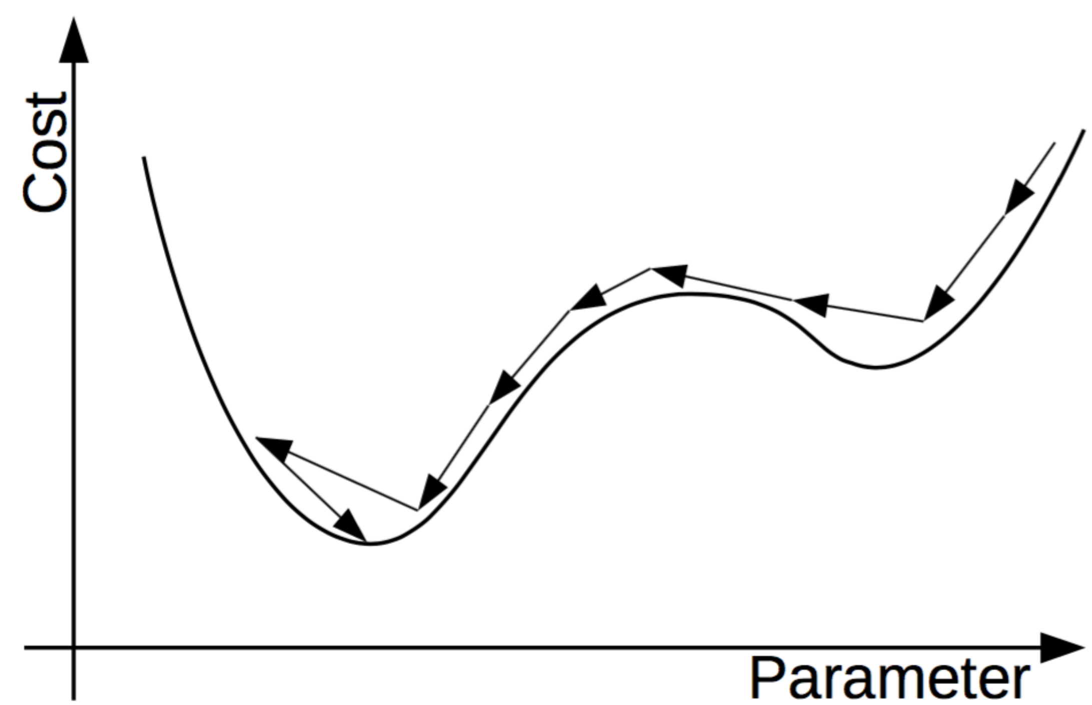
  → But: computational too expensive to evaluate all derivations of cost function w.r.t. all weights ($\mathcal{O}(10^5)$)

  → Back-propagation: change of weights ∝ costs

- Combine layers to build deep networks:
  - 2 layers: $\vec{y} = f(\boldsymbol{W_2} \cdot \vec{y_1} + \vec{b_2})$
  - n layers: $\vec{y} = (f_1 \circ f_2 \circ ... \circ f_n)(\boldsymbol{W} \cdot \vec{x} + \vec{b})$

- Many matrix and vector operations

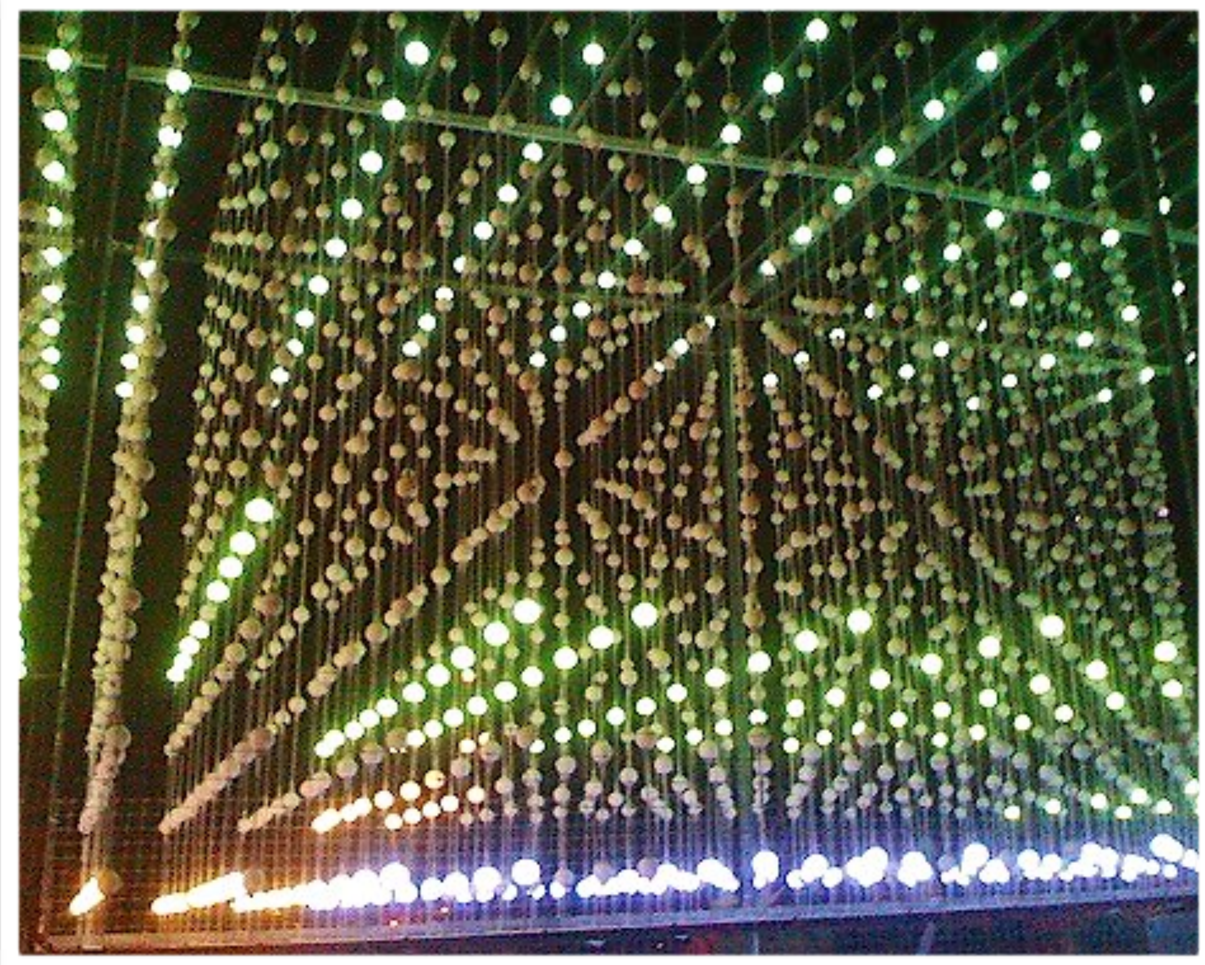
  → GPUs are mandatory!



Deep Neural Networks (3)

- Network architecture
- Layer activations
- Optimization algorithm
- Overtraining suppression:
  - L2 normalization
  - Random unit dropout
- Event weights
- Feature scaling
- …

Challenging hyper-parameter space



→ Many hyper-parameter combinations but only a few appear to work

- Challenge: "No separation in output distribution. Reason?"
  - BDT: Unfortunate variable selection
  - DNN: Unfortunate variable selection or network architecture not optimal (more likely)

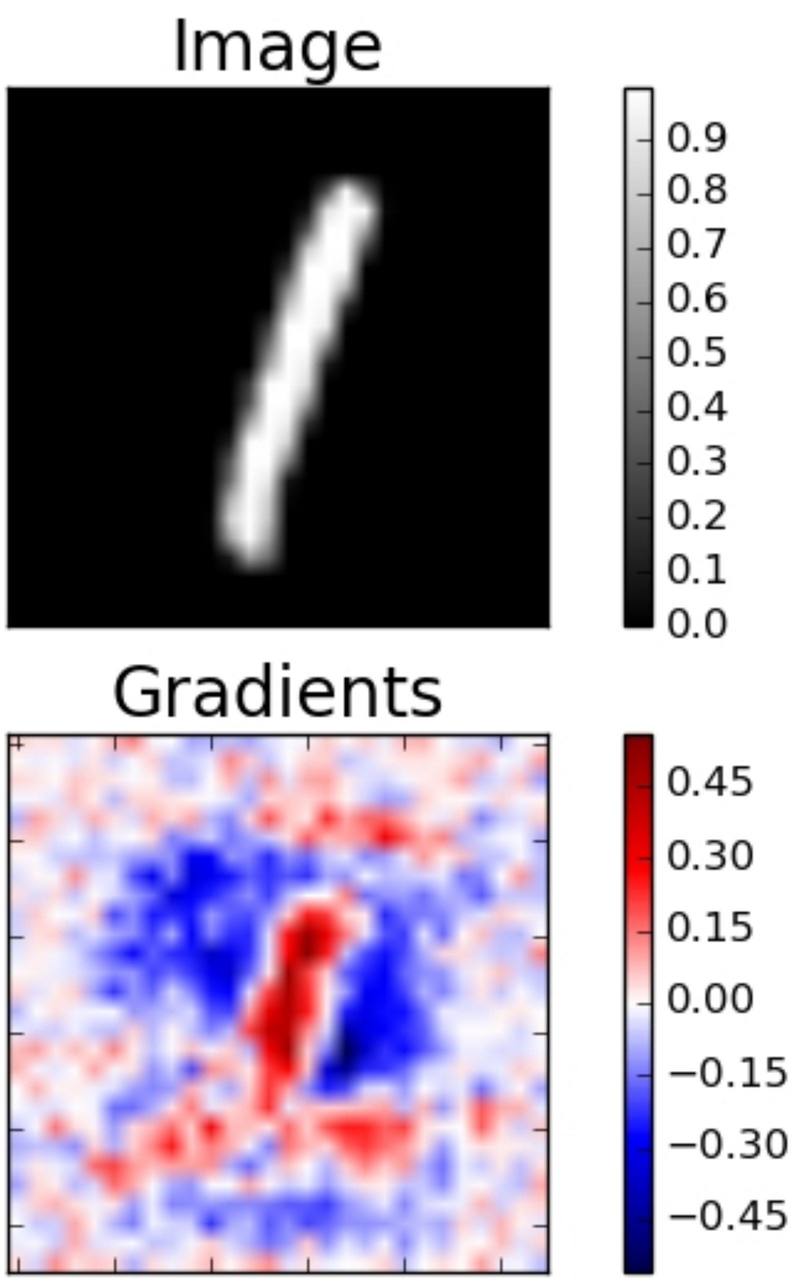
Category →

Output node ↓

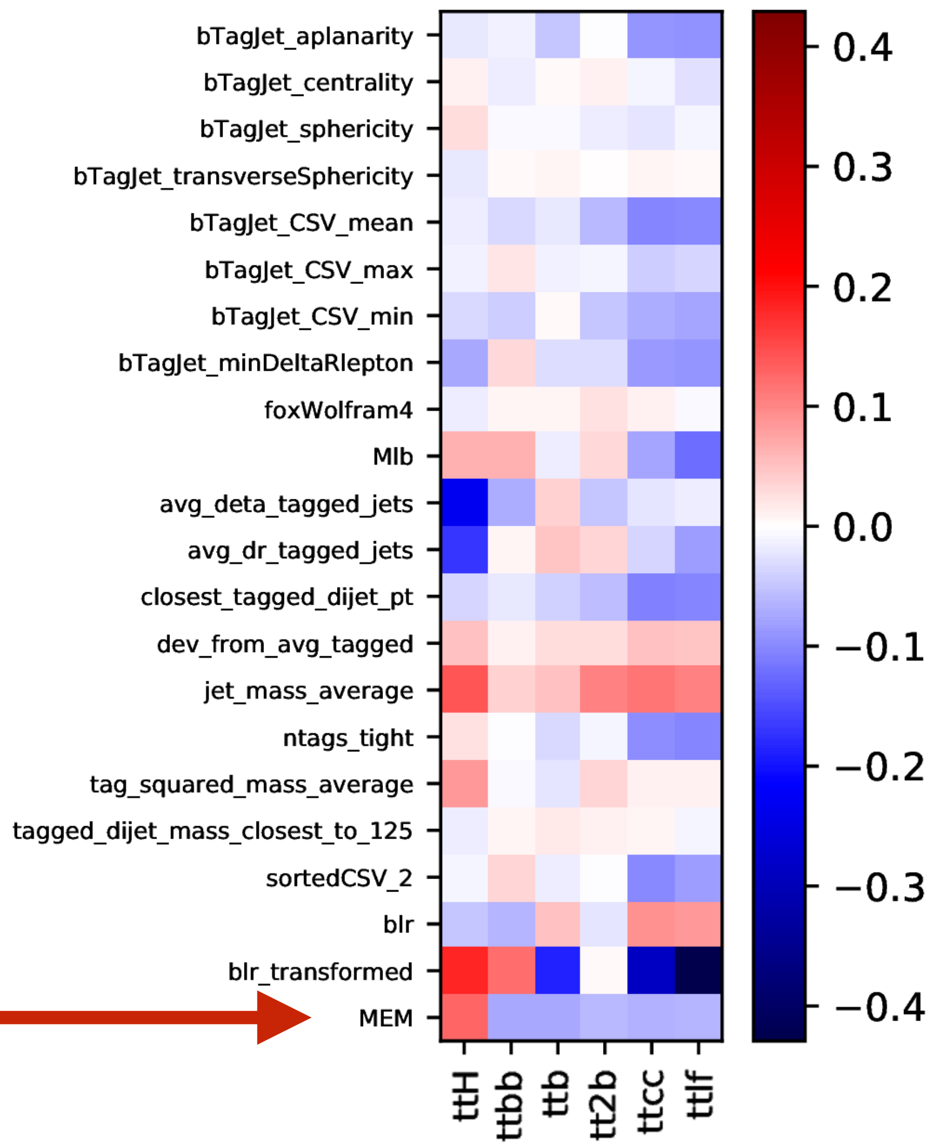
	ttH	ttbb	ttb	tt2b	ttcc	ttlf
ttH	✔					
ttbb		✔				
ttb			✔			
tt2b				✔		
ttcc					✔	
ttlf						✔

DNN interpretation

- Methods to interpret NN predictions, inspired by image recognition
- Define sensitivity via gradient of output w.r.t. inputs
  → "If input is varied, how does the output change?"
  - Determine derivative via `tf.gradient()`
- Other approaches possible
  (e.g layer-wise relevance propagation)

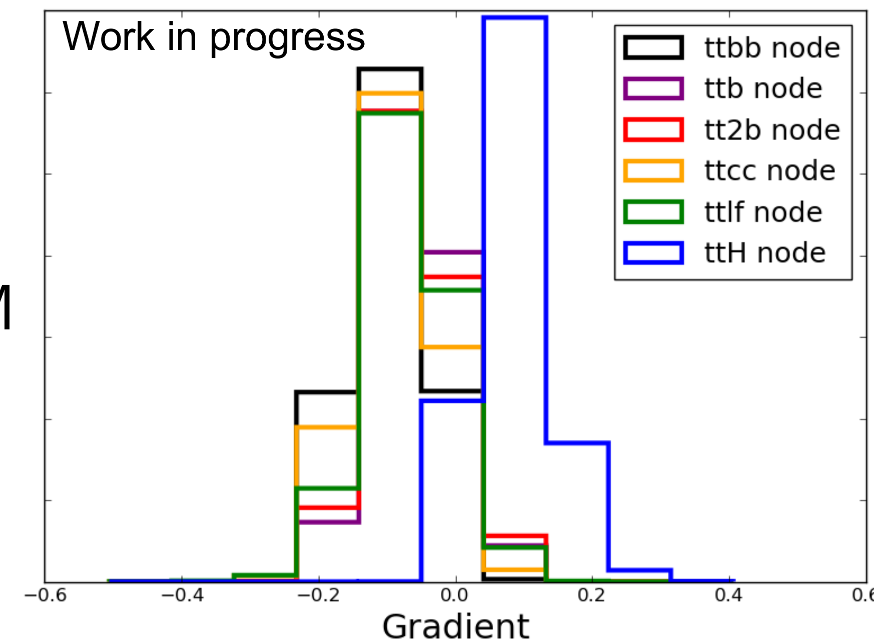


Apply per event *(here: ttH)*

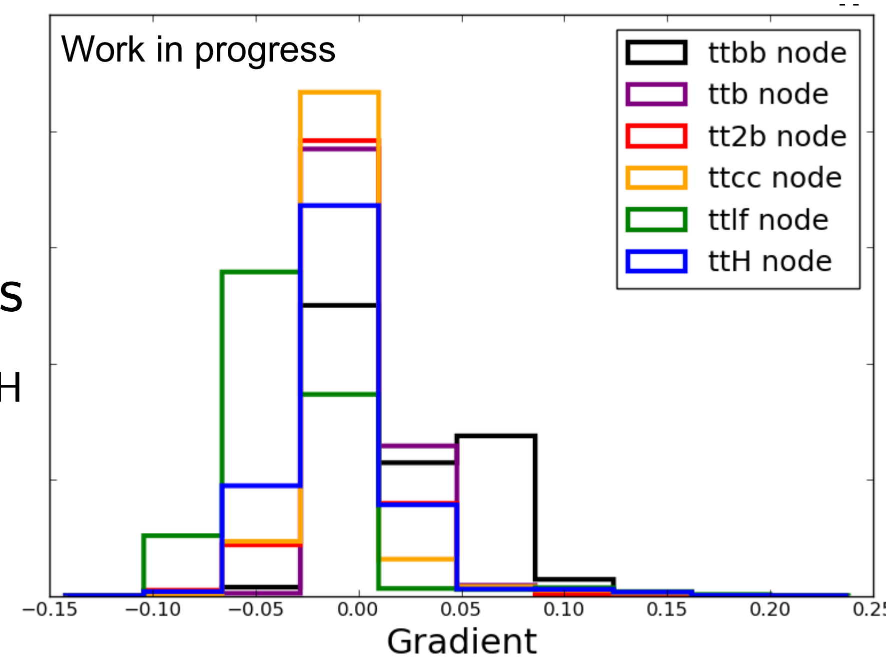


Apply to all events

MEM



Jet pair mass closest to m_H



- Method to open the network "black box"
- Possible to check impact of 2D correlations with 2nd derivatives
- Sensitivity can be used for variable ranking (e.g. "rank = mean(abs(sensitivity))")

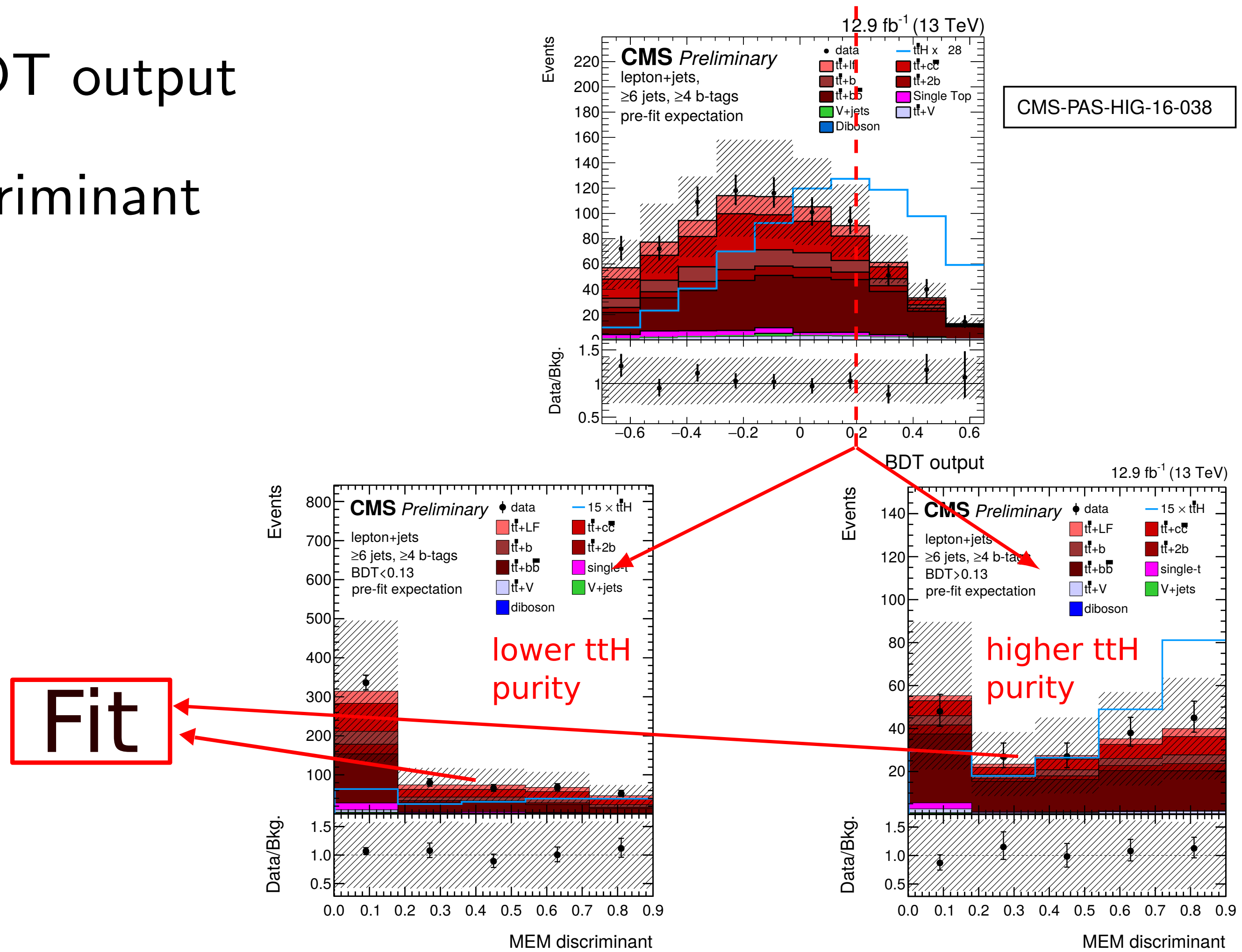
DL: BDT+MEM 2D method

- Well established method in previous Run II analyses (HIG-16-038, HIG-16-004)
- Jet - b-tag categorization: (≥4j,3t), (≥4j,≥4t)
  - (≥4j,3t): use BDT output as discriminant
  - (≥4j,≥4t): combine strengths of BDT and MEM
    - ▹ BDTs trained to discriminate ttH(bb) vs. inclusive tt+jets
    - ▹ Matrix element discriminants constructed to separate ttH(bb) and tt+bb

    → Split events at median of signal BDT output

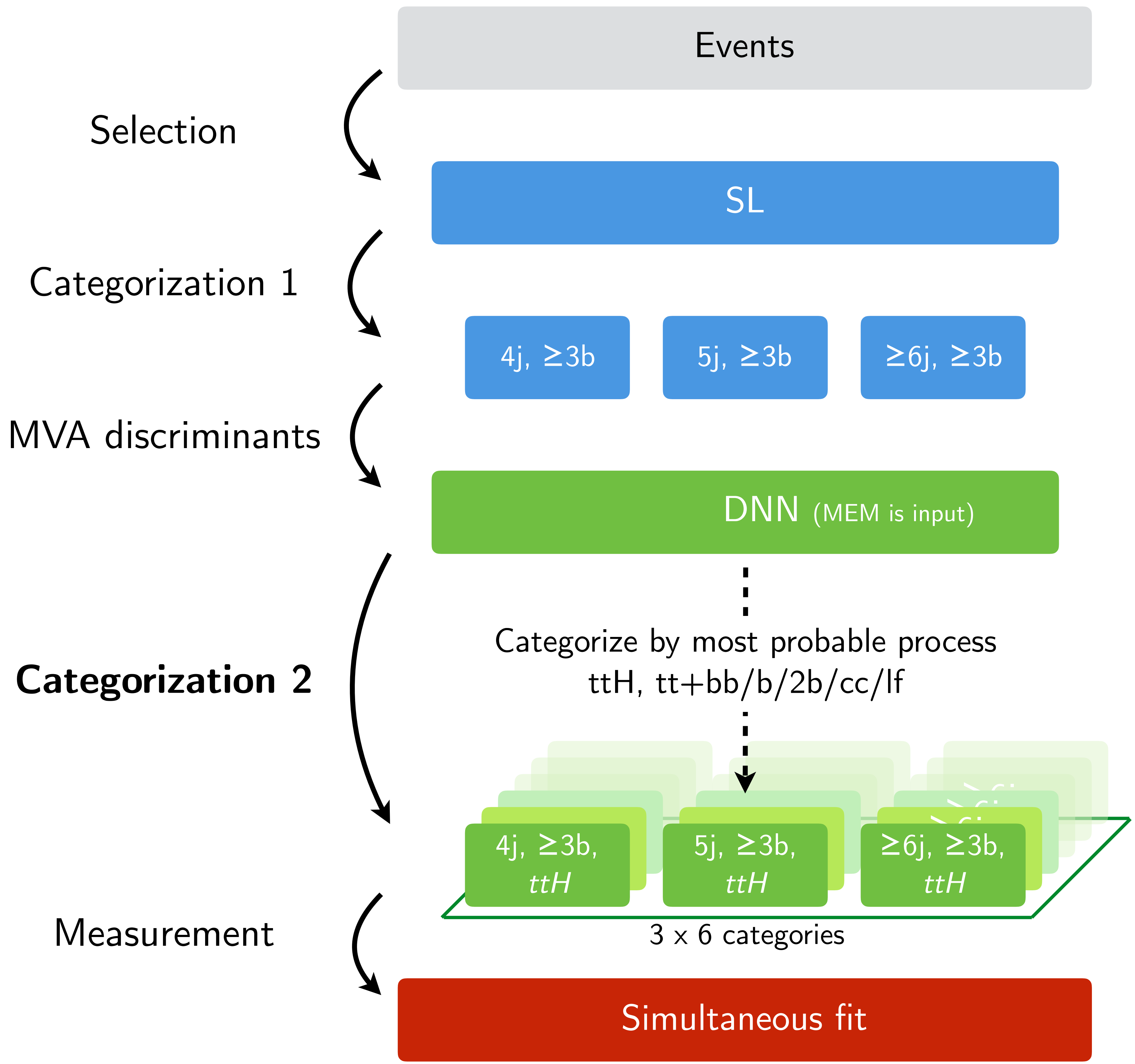
    → In each *category*, use MEM as discriminant

- 3 categories in total



Overall analysis strategy

Marcel Rieger - 31.5.18



Overall analysis strategy