LST T5 DNN

Attempts to integrate a DNN into LST

May 12th, 2023

P. Chang, ***J. Guiang***

UC San Diego

UF
UNIVERSITY of FLORIDA

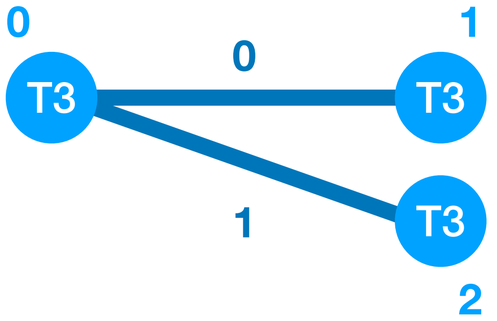
Overview

- Training a very simple NN to classify T5s
- Propagated simple NN to LST by writing matrix multiplication “by hand”
  - Explicit for-loops on each thread, hard-coded weights/biases
- Plots of input features available here
  - Norm: $p_T \rightarrow \log_{10}(p_T)$, radius $\rightarrow \log_{10}$(radius)
- **Seeing inconsistent results btwn. HPG & LST**

Object	Feature
T3	p_T
	Inner anchor hit r, z, ϕ, η, layer
	Middle anchor hit r, z, ϕ, η, layer
	Outer anchor hit r, z, ϕ, η, layer
T5 candidate	p_T, η, ϕ
	Inner T3 radius
	Bridge T3 radius
	Outer T3 radius

Scaled such that all features $\in$ [0, 1]

T5 DNN Details: Inputs



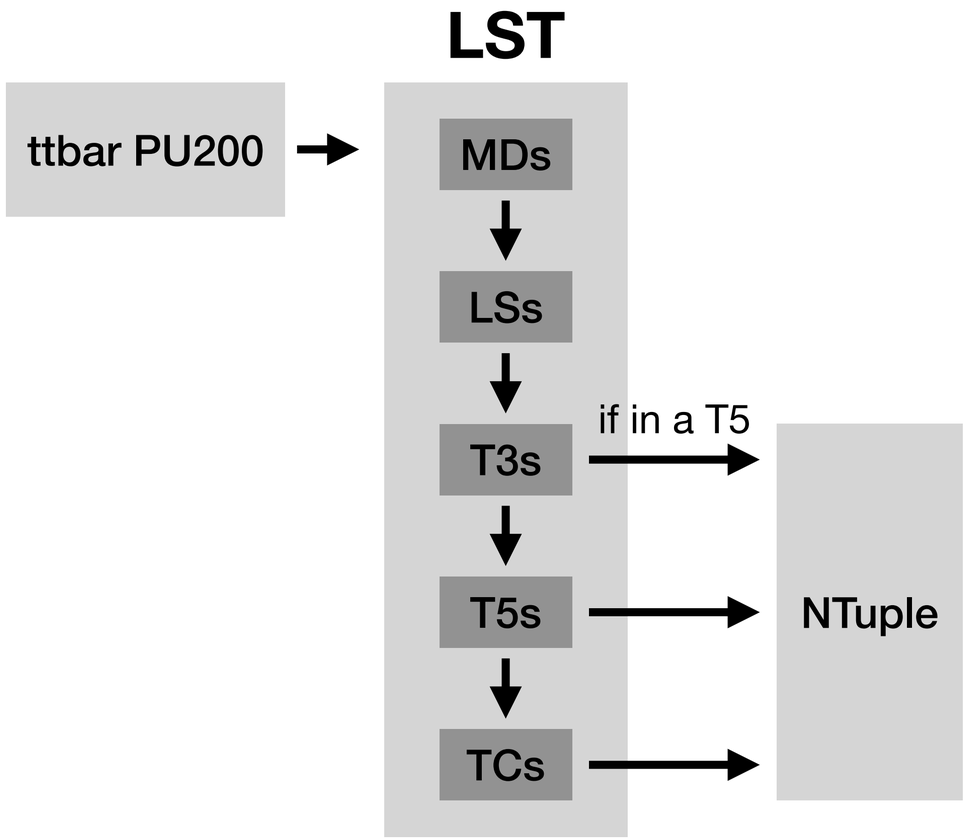
- Distinguish **HPG variables** from **LST variables**
  - **HPG variables** are arranged in a graph, input vector assembled T5-by-T5
  - **LST variables** are already assembled for each T5
- Train DNN with **HPG variables** on HiPerGator (HPG)
- Real inference done with **LST variables** on UCSD GPU
- **HPG** and **LST variables** are computed in different places, but **should be the same**!
  - Save both to LST output N-tuple so we can compare

HPG variables		**Event N**
T5	p_T	{3.5, 2.1}
	…	…
	Inner T3 idx	{0, 0}
	Outer T3 idx	{1, 2}
T3	p_T	{3.1, 3.7, 2.1}
	…	…

LST variables	**Event N**
T5 p_T	{3.5, 2.1}
…	…
Inner T3 p_T	{3.1, 3.1}
Outer T3 pT	{3.7, 2.1}
…	…

T5 DNN Details: N-tuples

- Produced 3 N-tuples:
  1. Run LST w/out χ^2 cuts → output **N-tuple 1**
     - Train **DNN** on this N-tuple
  2. Run LST w/out χ^2 cuts → output **N-tuple 2**
  3. Run LST w/ **DNN** cut → output **N-tuple 3**
- **N-tuple 1** and **N-tuple 2** should be *nearly* identical
- **N-tuple 3** only has T5s with DNN score > X

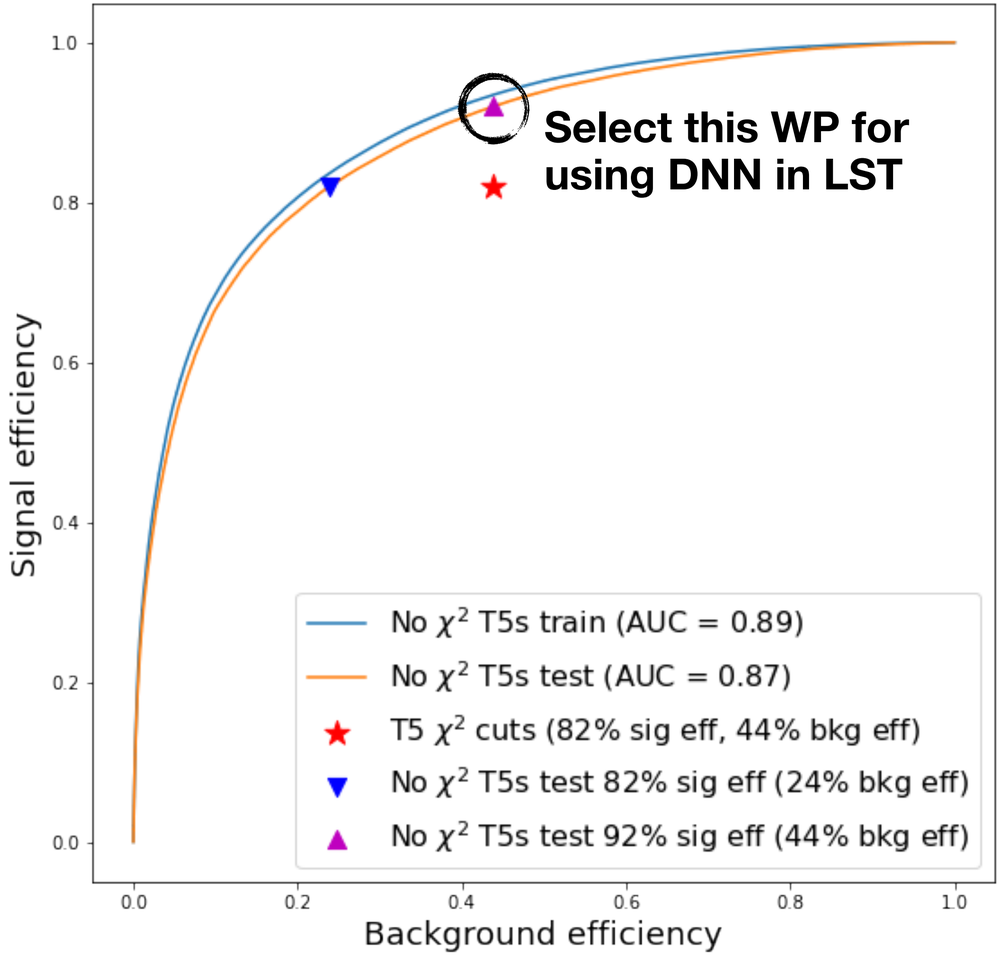


T5 DNN Results

It works…?

- Produced 3 N-tuples:
  1. Run LST w/out χ^2 cuts → output **N-tuple 1**
  2. Run LST w/out χ^2 cuts → output **N-tuple 2**
  3. Run LST w/ **DNN** cut → output **N-tuple 3**
- DNN trained on **N-tuple 1** gives similar performance to previous results
  - T5 p_T now calculated using both inner and outer radii (previously only inner)
  - T3 radius calculation slightly different

2 hidden layers + no δr + p_T fix



UC San Diego

T5 DNN Results

It works…?

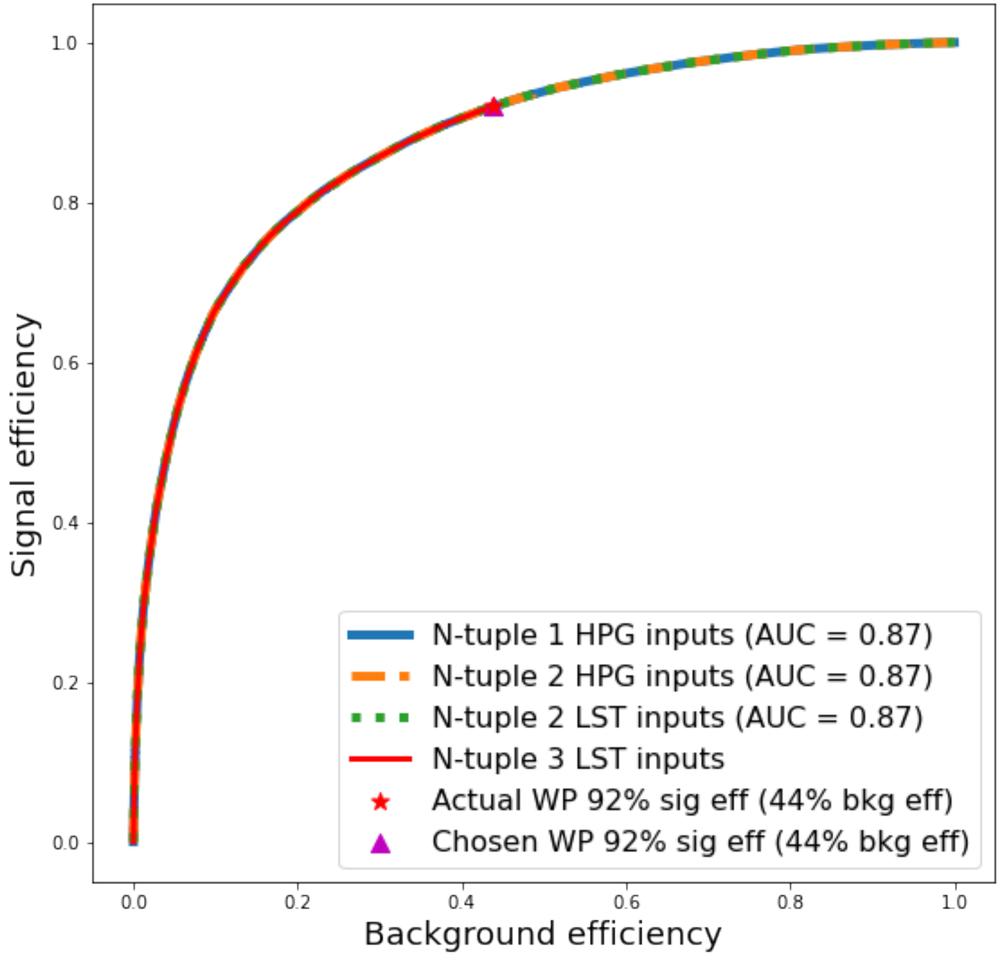
- Produced 3 N-tuples:
  1. Run LST w/out χ^2 cuts → output **N-tuple 1**
  2. Run LST w/out χ^2 cuts → output **N-tuple 2**
  3. Run LST w/ **DNN** cut → output **N-tuple 3**

✓ Running the DNN on HPG should give the same results for **N-tuple 1** and **N-tuple 2** (— vs. --)

✓ Using HPG inputs vs. LST inputs should give the same result (-- vs. ···)

✓ DNN on HPG vs. DNN in LST should give the same sig/bkg efficiency (▲ vs. ▲)

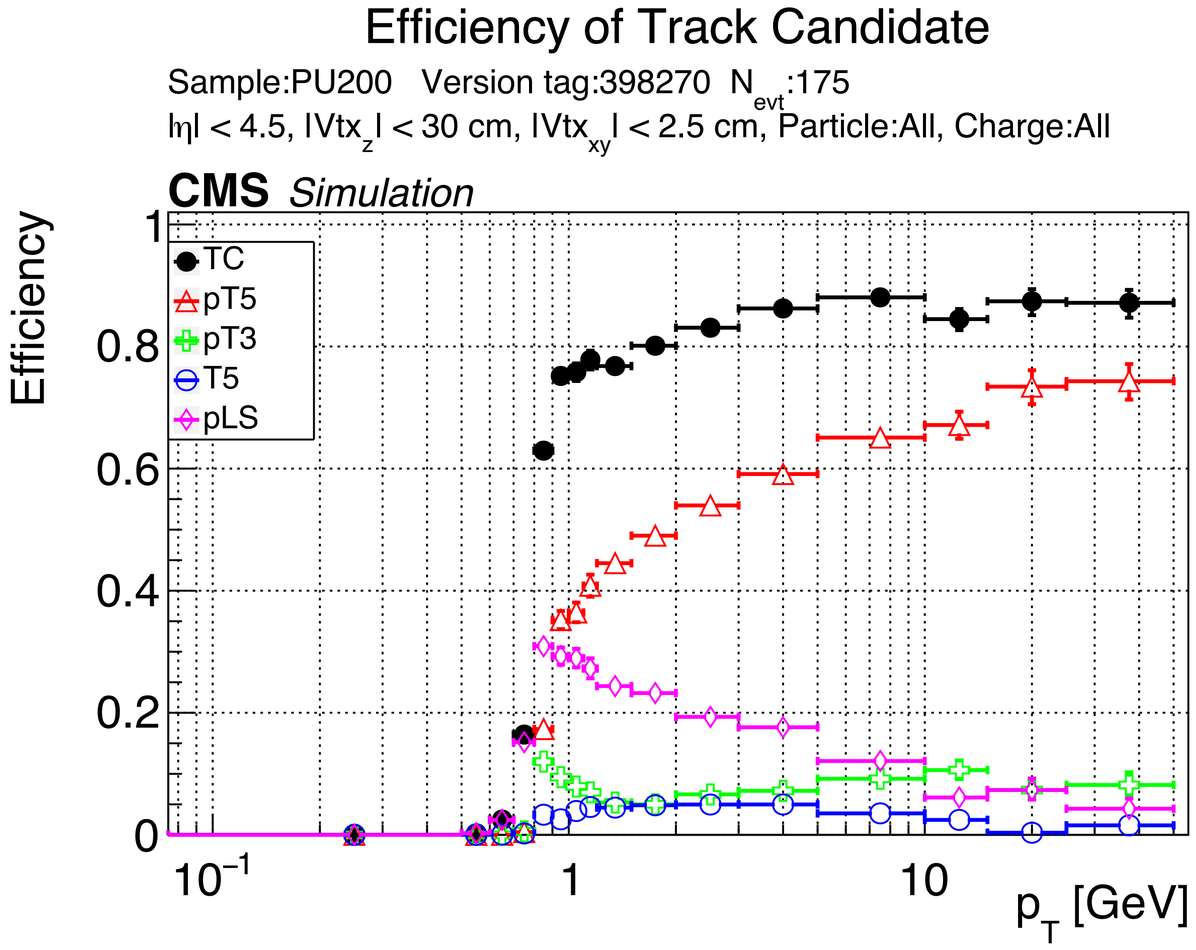
2 hidden layers + no δr + p_T fix

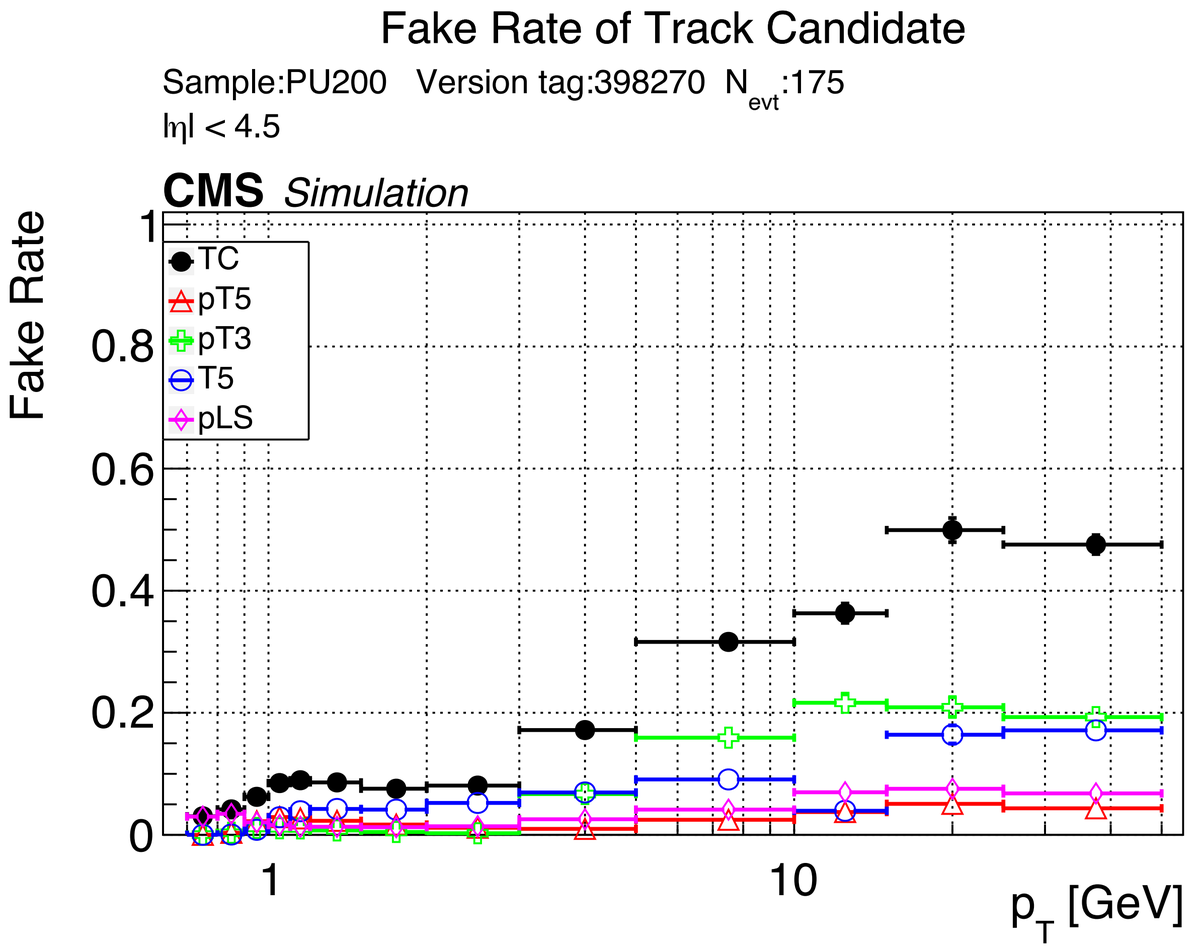


UC San Diego

LST Master Efficiency Plots

Showing p_T-binned efficiency and fake rate

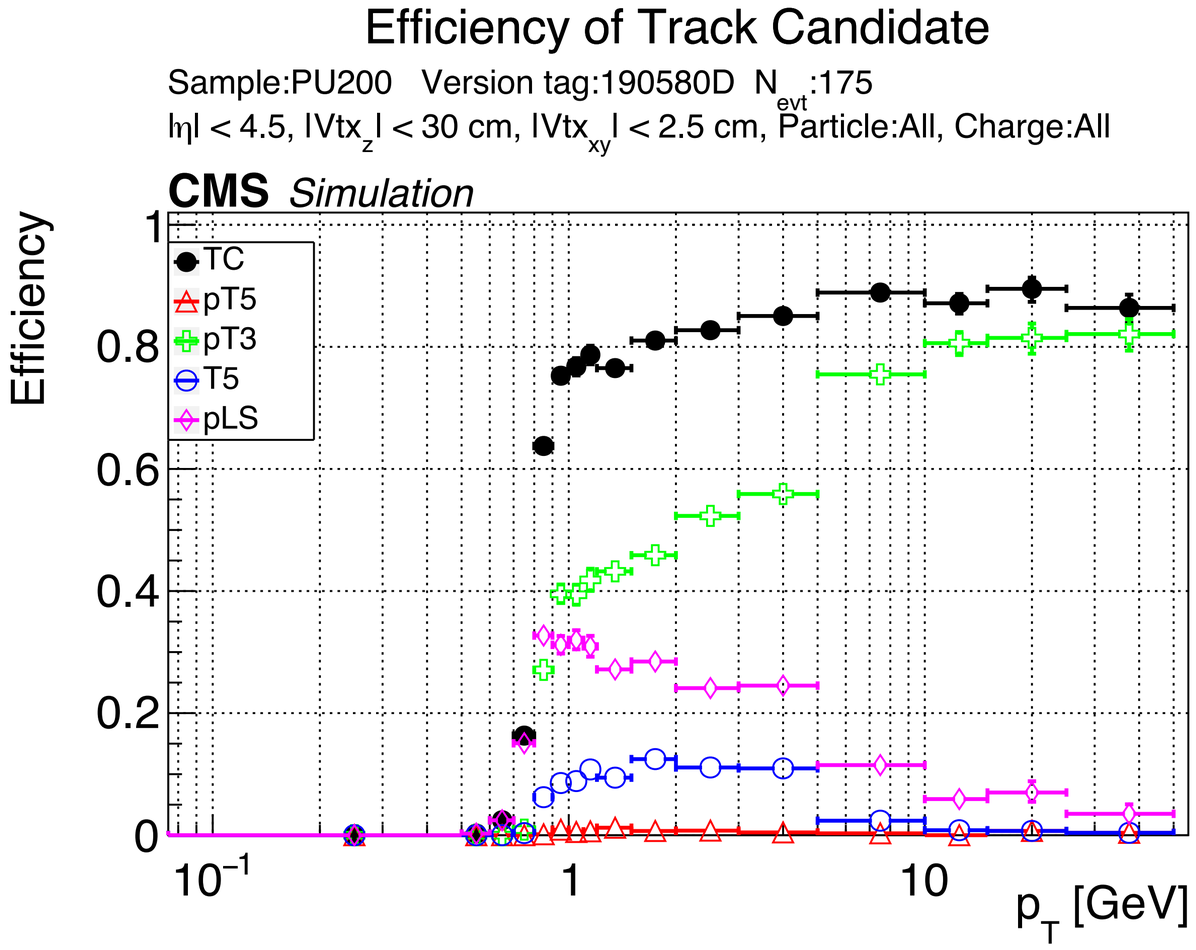


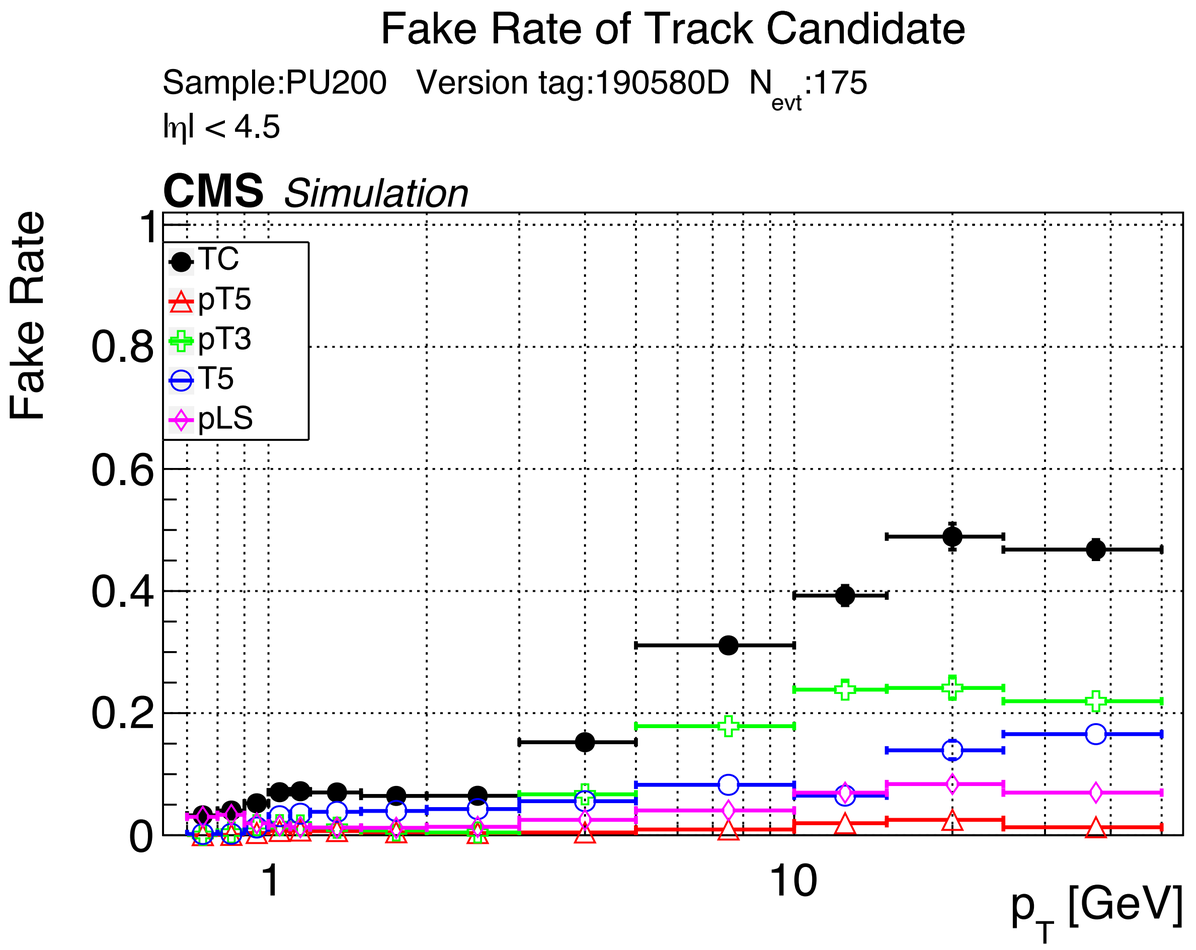


UC San Diego

LST DNN Efficiency Plots

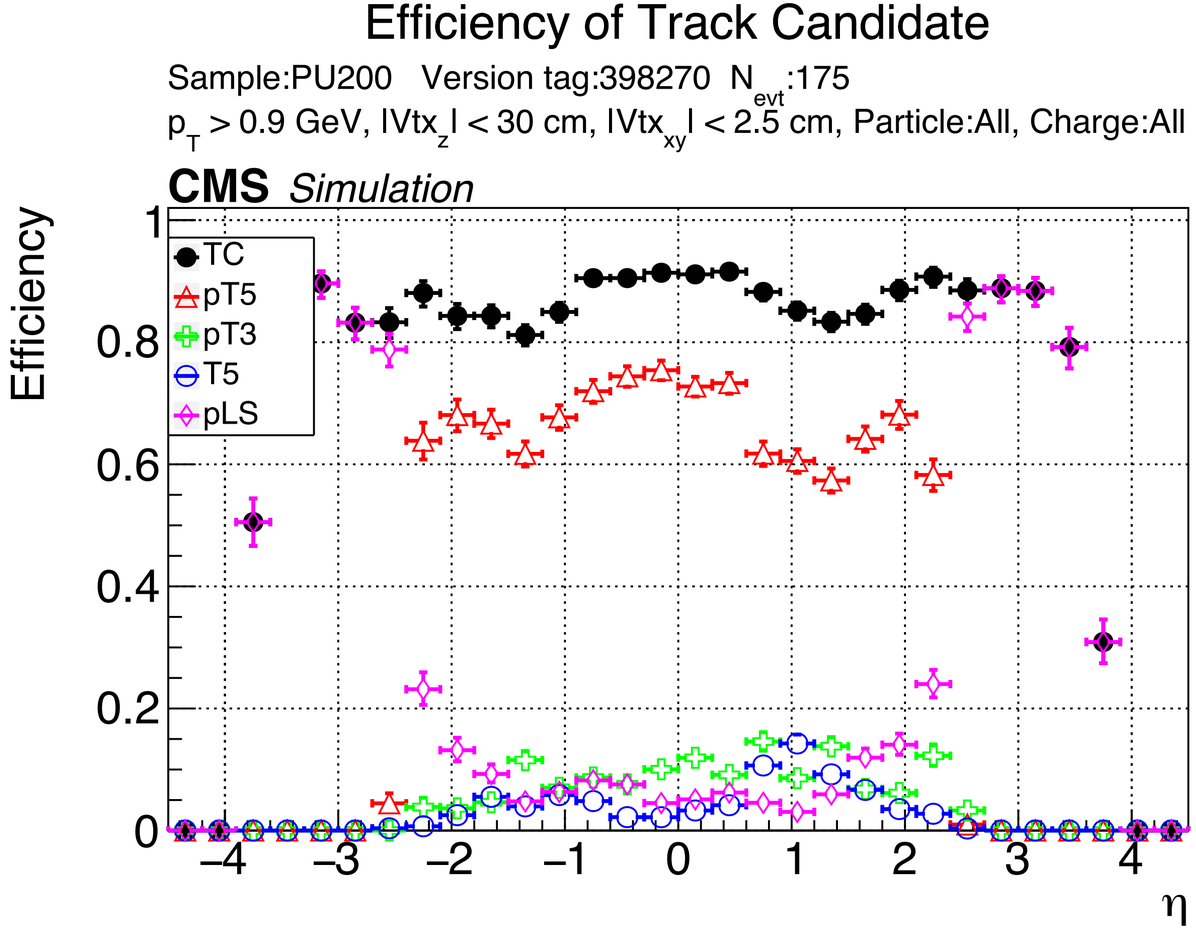
The DNN is somehow **killing all pT5s**

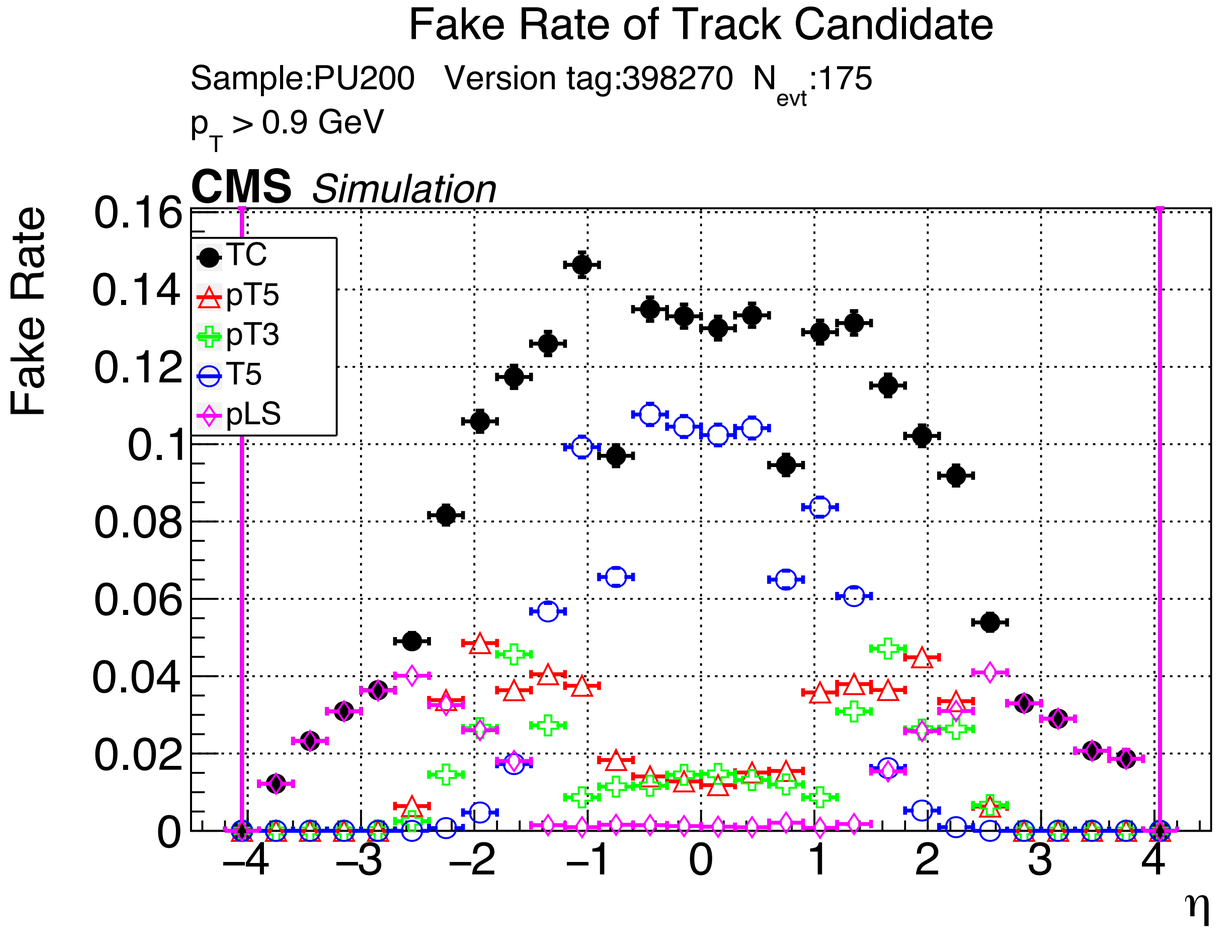




LST Master Efficiency Plots

Showing η-binned efficiency and fake rate

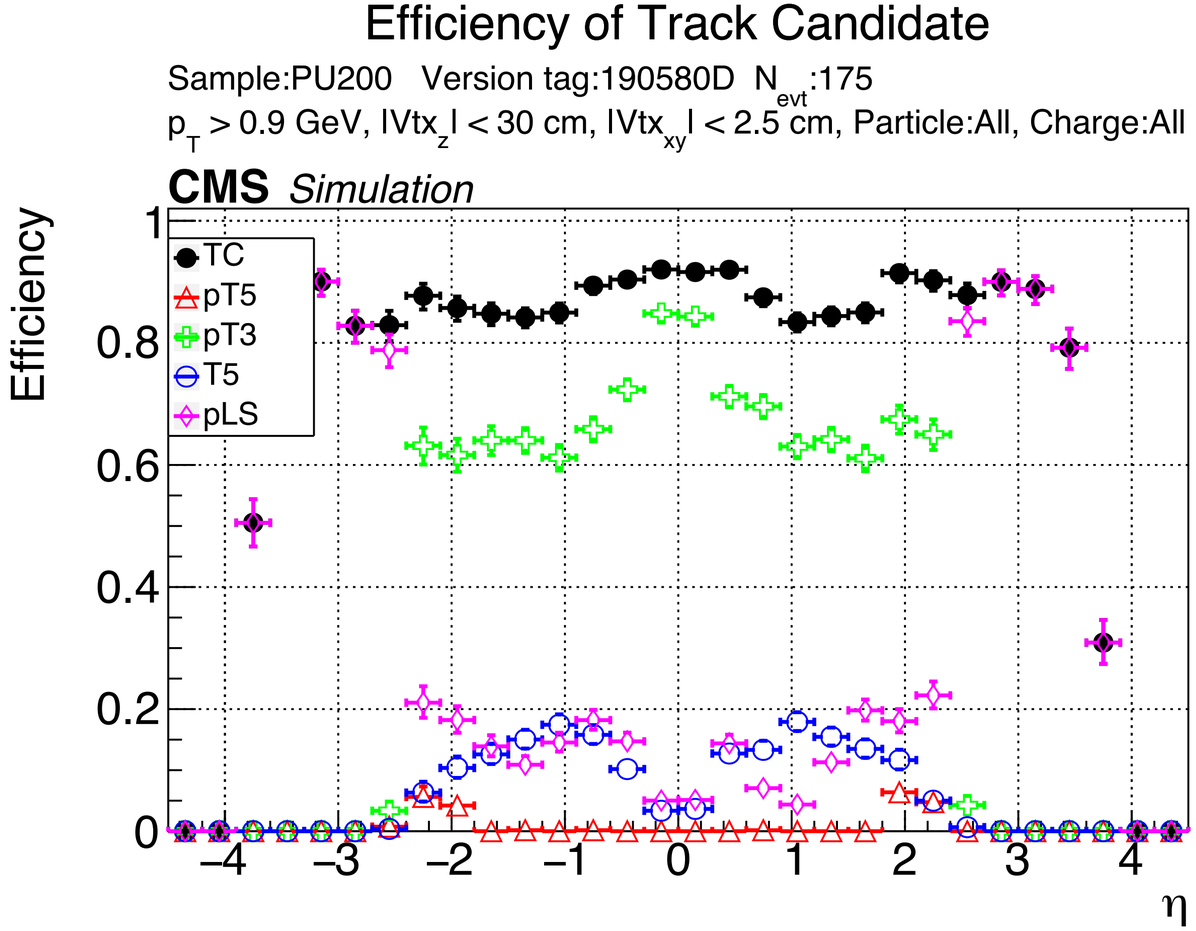


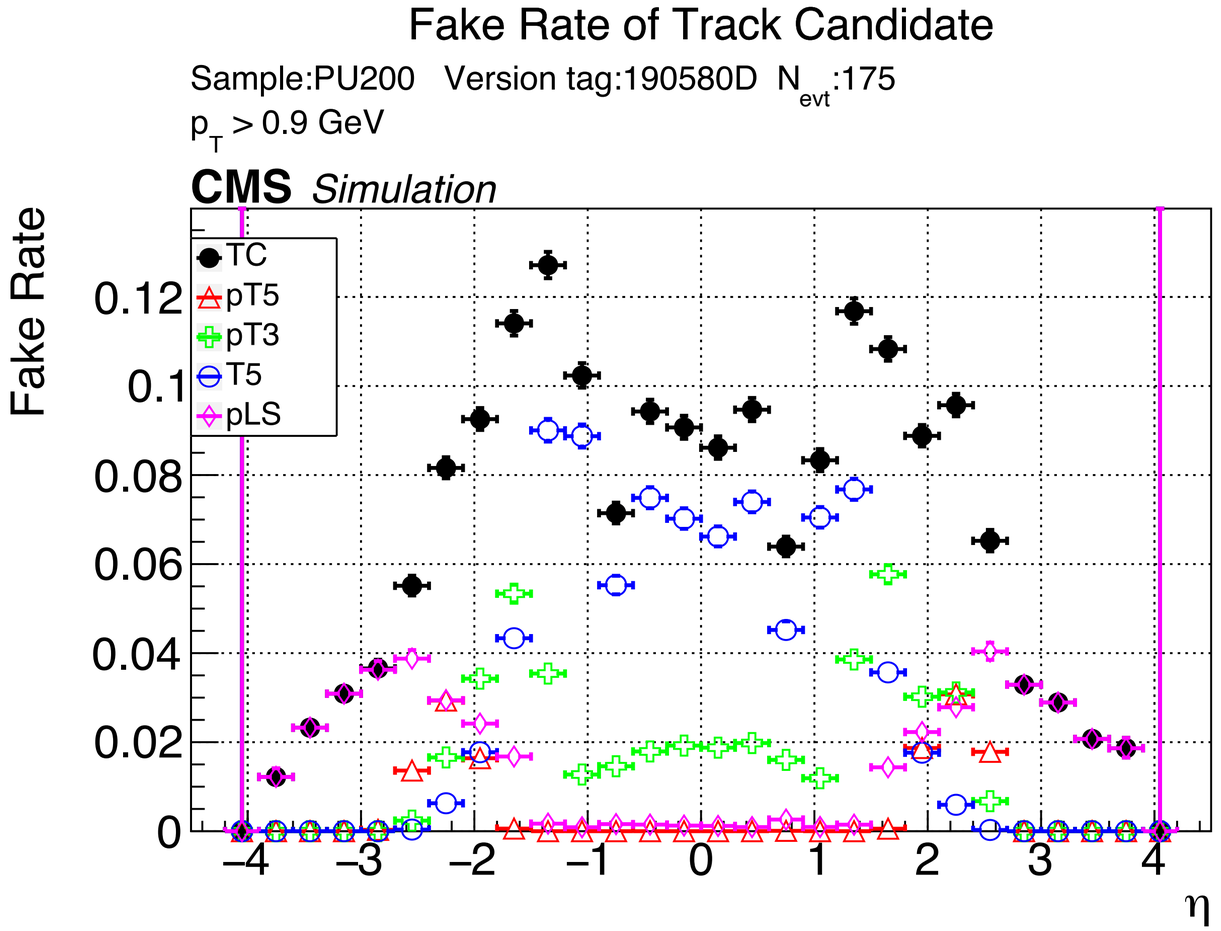


UC San Diego

LST DNN Efficiency Plots

DNN is somehow **killing all pT5s**





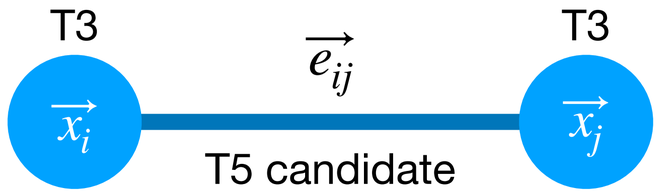
UC San Diego

Summary

- DNN gives consistent results everywhere
- However, we are somehow killing all pT5s!
  - This does not make sense: we are (we think) matching the χ^2 bkg. eff. while improving the sig. eff. by ~10%

Backup

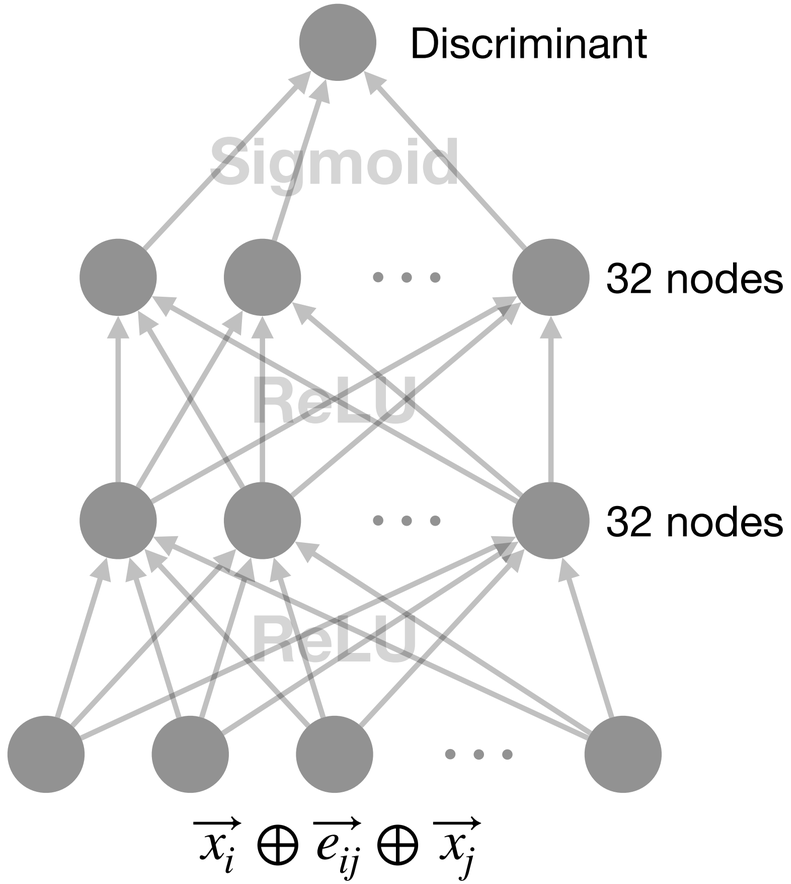
T5 DNN Configuration



Object	Feature
T3	p_T
	Inner anchor hit r, z, ϕ, η, layer
	Middle anchor hit r, z, ϕ, η, layer
	Outer anchor hit r, z, ϕ, η, layer
T5 candidate	p_T, η, ϕ
	Inner T3 radius
	Bridge T3 radius
	Outer T3 radius

Scaled such that all features ∈ [0, 1]

2 hidden layers

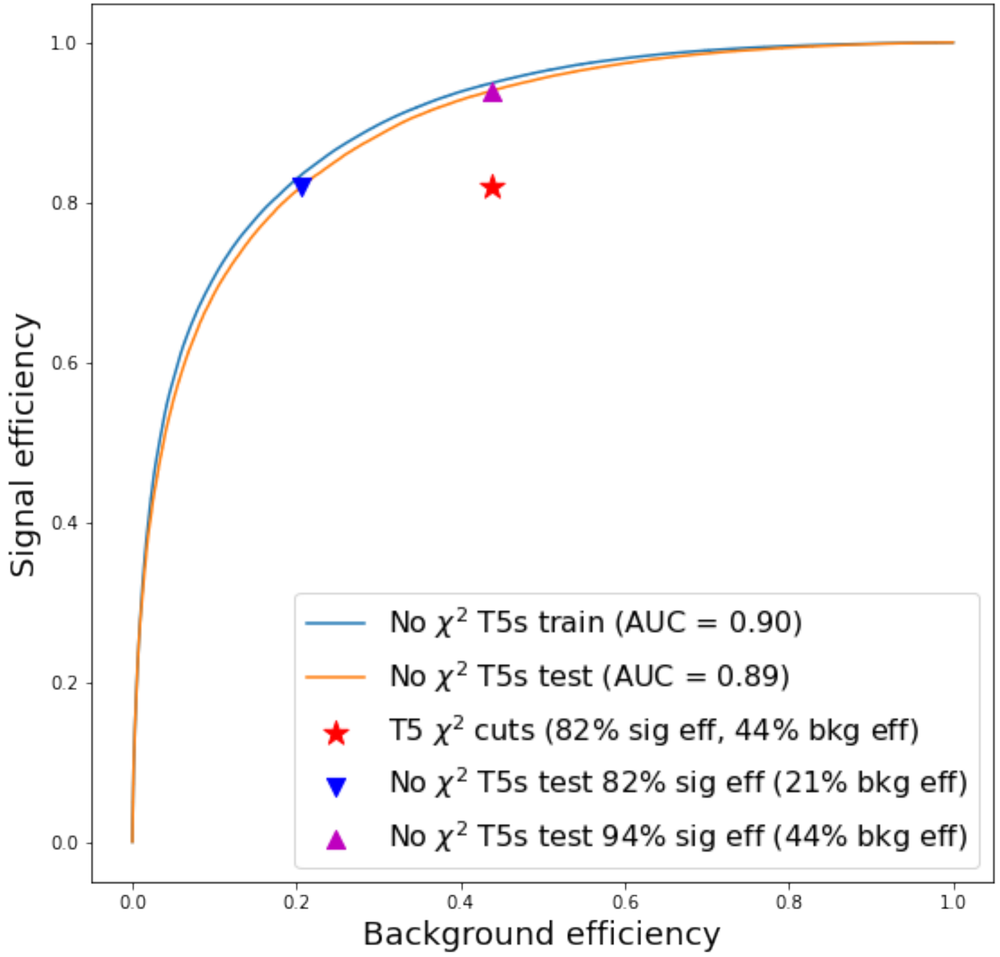


UC San Diego

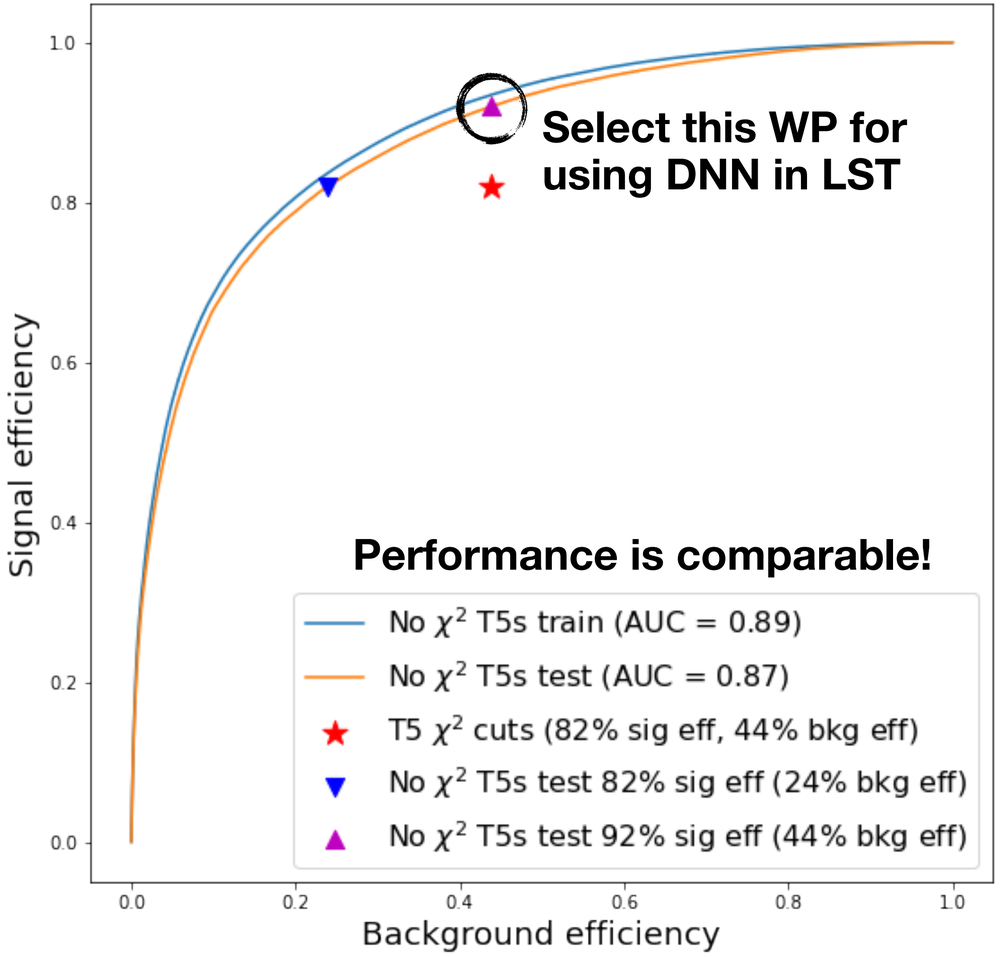
T5 DNN Performance

Performance is comparable to previous results

2 hidden layers (old result)



2 hidden layers + no δr + p_T fix

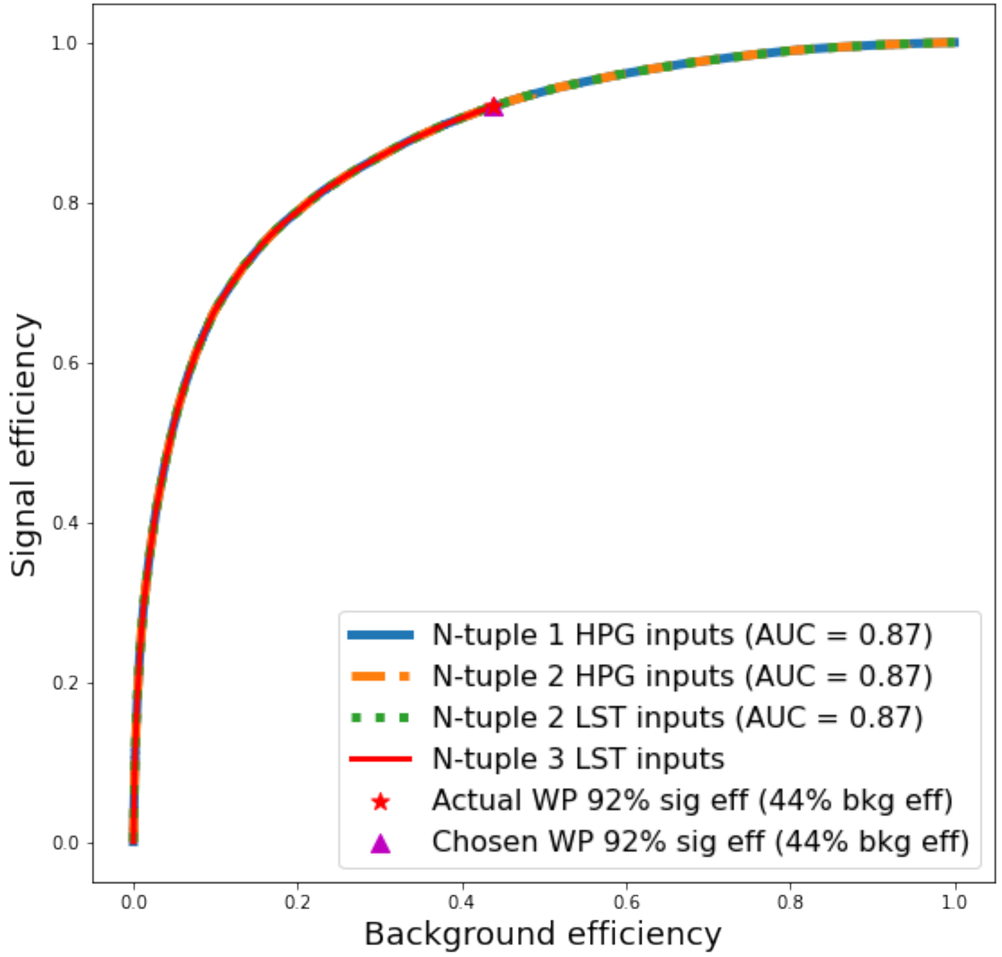


T5 DNN Results

It works…?

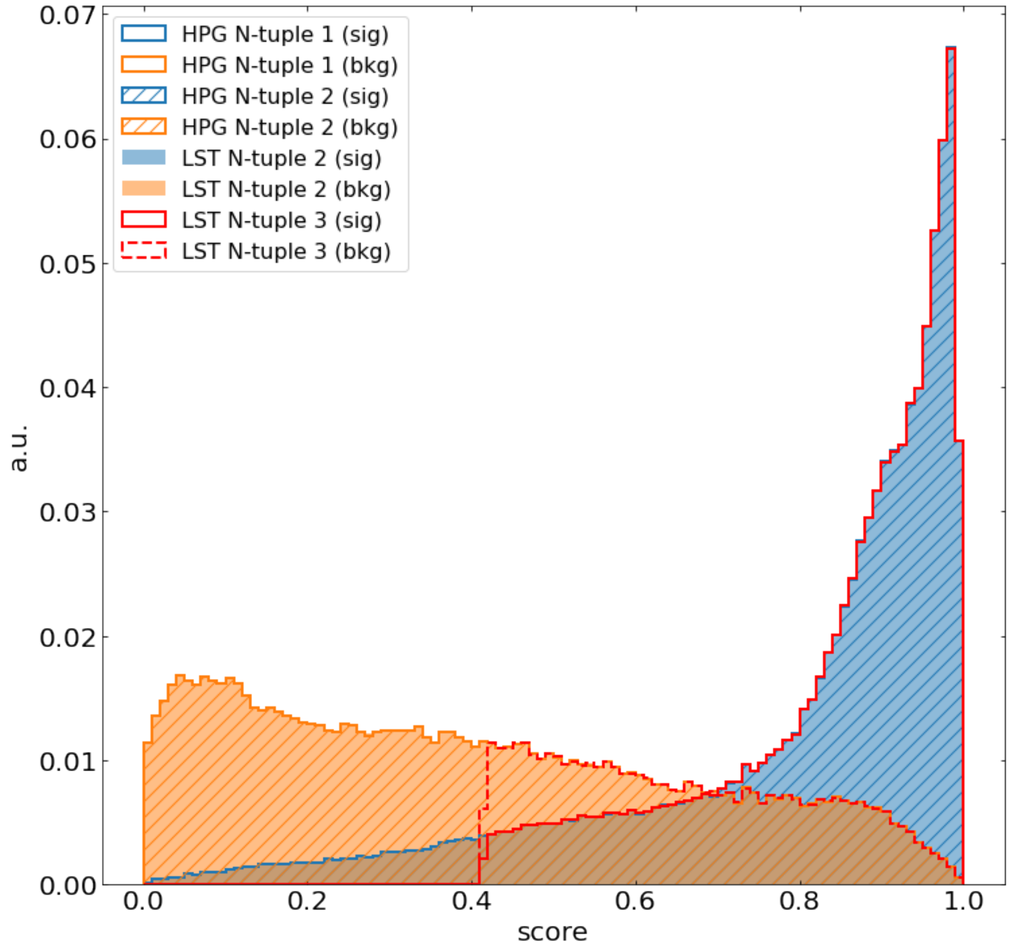
- — curve: $FPR_3 \times (N_3/N_2)$ vs. $TPR_3 \times (P_3/P_2)$
- ★ point: N_3/N_2 vs. P_3/P_2
- $FPR_i = FP_i/N_i$
  - FP_i = # false positives for DNN > X (N-tuple i)
  - N_i = # total negatives in N-tuple i
- $TPR_i = TP_i/P_i$
  - TP_i = # true positives for DNN > X (N-tuple i)
  - P_i = # total positives in N-tuple i

2 hidden layers + no δr + p_T fix

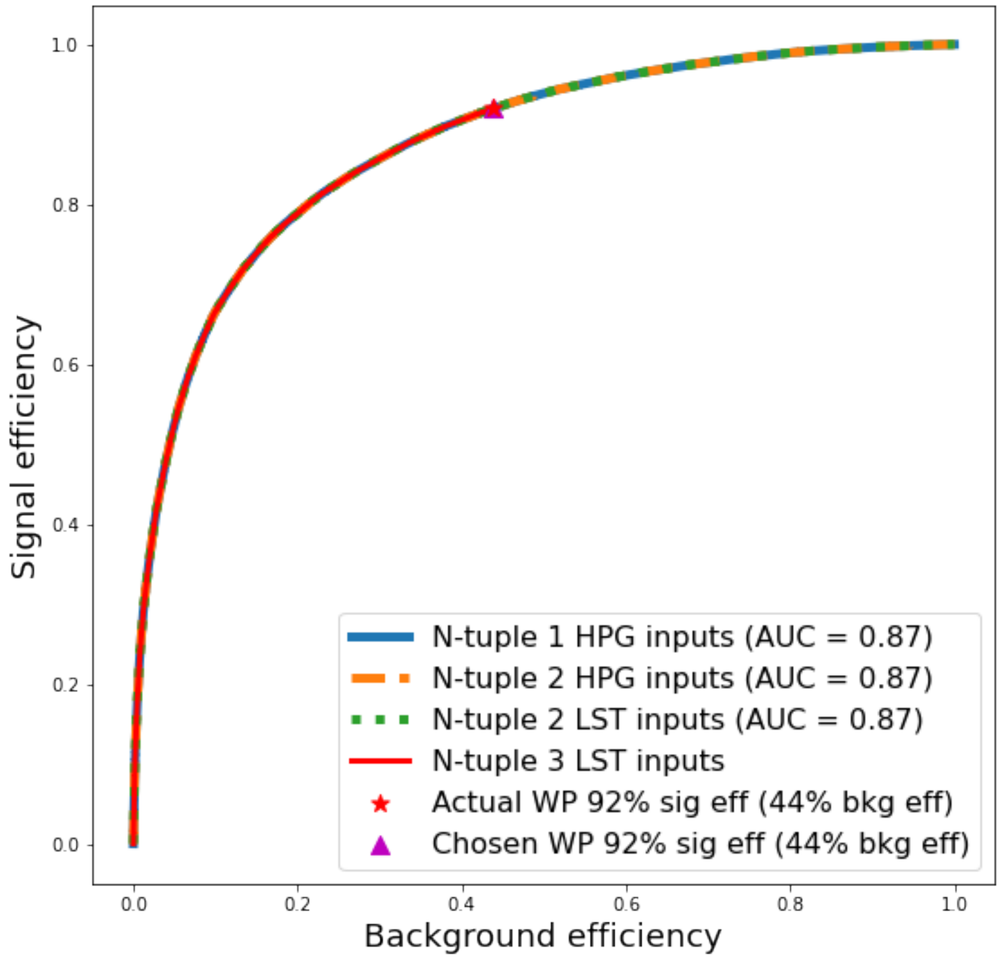


T5 DNN Results

It works…?



2 hidden layers + no δr + p_T fix



T5 DNN Results

layers + no δr + p_T fix