LST T5 DNN

Successful integration of DNN into LST

May 19th, 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)
- **Now able to actually assess the performance of the DNN (after fixing bug on next slide)**

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]

Bug Fixing

BadQuintuplet.cu

```
__device__ bool SDL::runQuintupletDefaultAlgo(..., regressionRadius, ...)
{
    bool pass = true;

    ...

    return pass and (inference > 0.6121...); // skip everything below

    ...

    computeSigmasForRegression(...);
    regressionRadius = computeRadiusUsingRegression(...);

    //extra chi squared cuts!
    if(regressionRadius < 5.0f/(2.f * k2Rinv1GeVf))
    {
        pass = pass and passChiSquaredConstraint(...);
        if(not pass) return pass;
    }

    //compute the other chisquared
    //non anchor is always shifted for tilted and endcap!
    float nonAnchorDelta1[5], nonAnchorDelta2[5], nonAnchorSlopes[5];
    float nonAnchorxs[] = {...};
    float nonAnchorys[] = {...};

    computeSigmasForRegression(...);
    nonAnchorChiSquared = computeChiSquared(...);
    return pass;
}
```

Quintuplet.cu

```
__device__ bool SDL::runQuintupletDefaultAlgo(..., regressionRadius, ...)
{
    bool pass = true;

    ...

    // return pass and (inference > 0.6121...);

    ...

    computeSigmasForRegression(...);
    regressionRadius = computeRadiusUsingRegression(...);

    //extra chi squared cuts!
    if(regressionRadius < 5.0f/(2.f * k2Rinv1GeVf))
    {
        // pass = pass and passChiSquaredConstraint(...);
        // if(not pass) return pass;
    }

    //compute the other chisquared
    //non anchor is always shifted for tilted and endcap!
    float nonAnchorDelta1[5], nonAnchorDelta2[5], nonAnchorSlopes[5];
    float nonAnchorxs[] = {...};
    float nonAnchorys[] = {...};

    computeSigmasForRegression(...);
    nonAnchorChiSquared = computeChiSquared(...);
    return pass and (inference > 0.6121...);
}
```

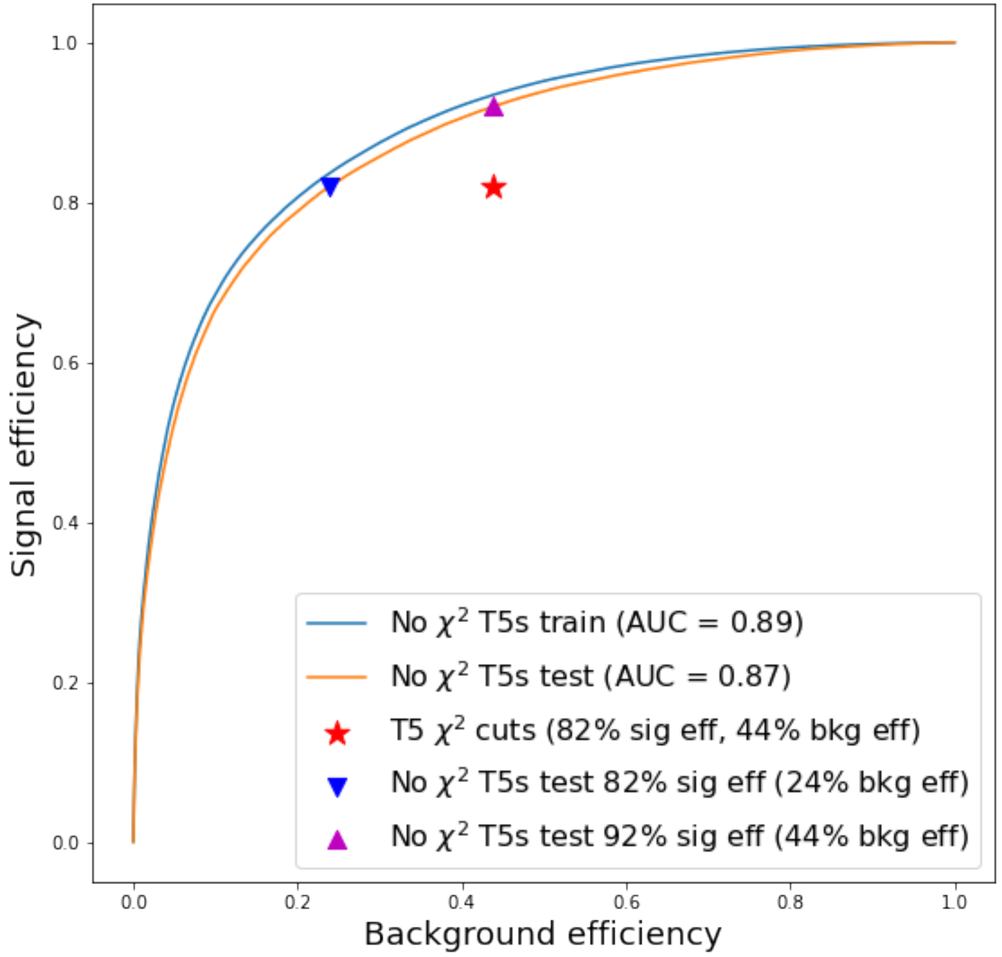
The `regressionRadius` was not being calculated, but gets used in pT5 building!

Does `regressionRadius` really need to be used?

T5 DNN Results

- Trying two working points:
  - ▼ = Match sig eff, 20% better bkg eff
  - ▲ = Match bkg eff, 10% better sig eff

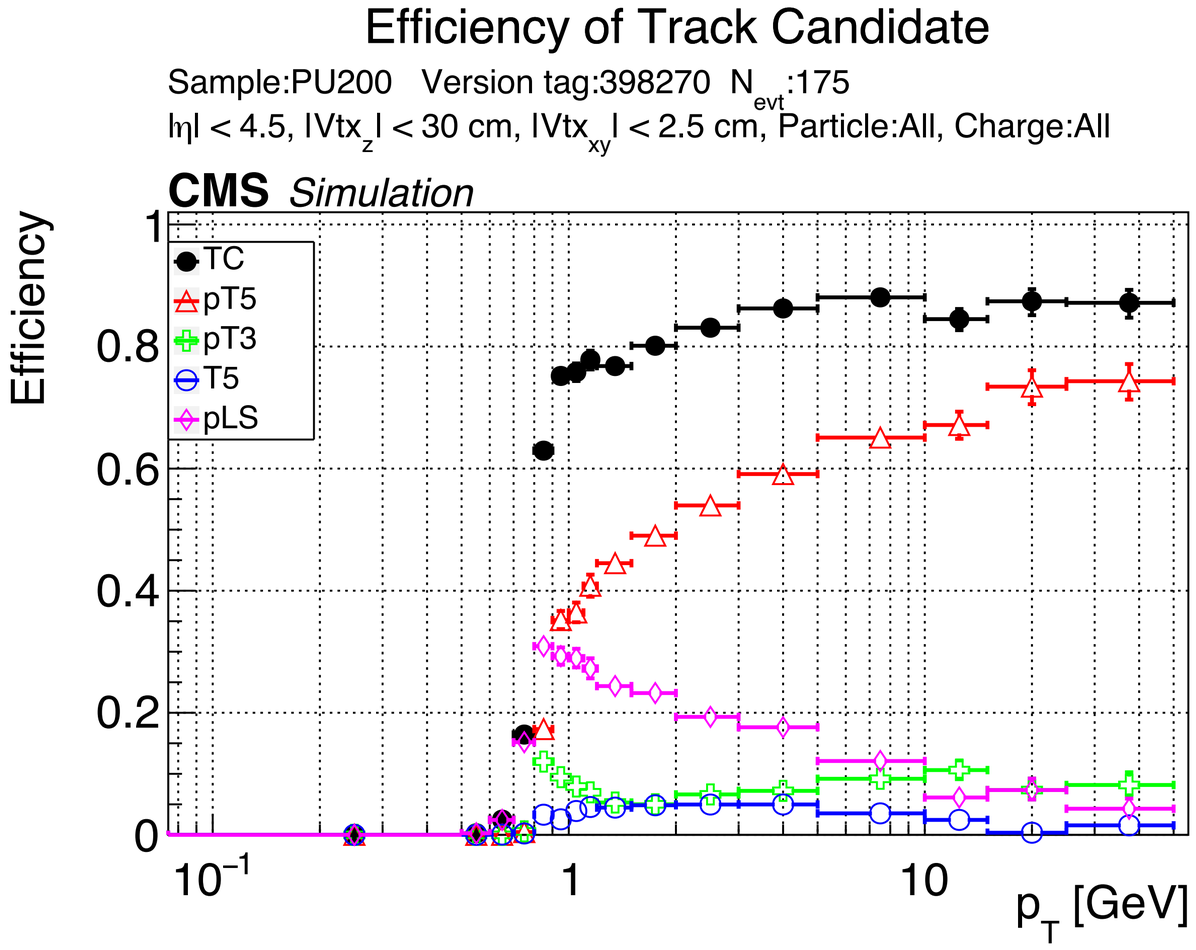
2 hidden layers + no δr + p_T fix

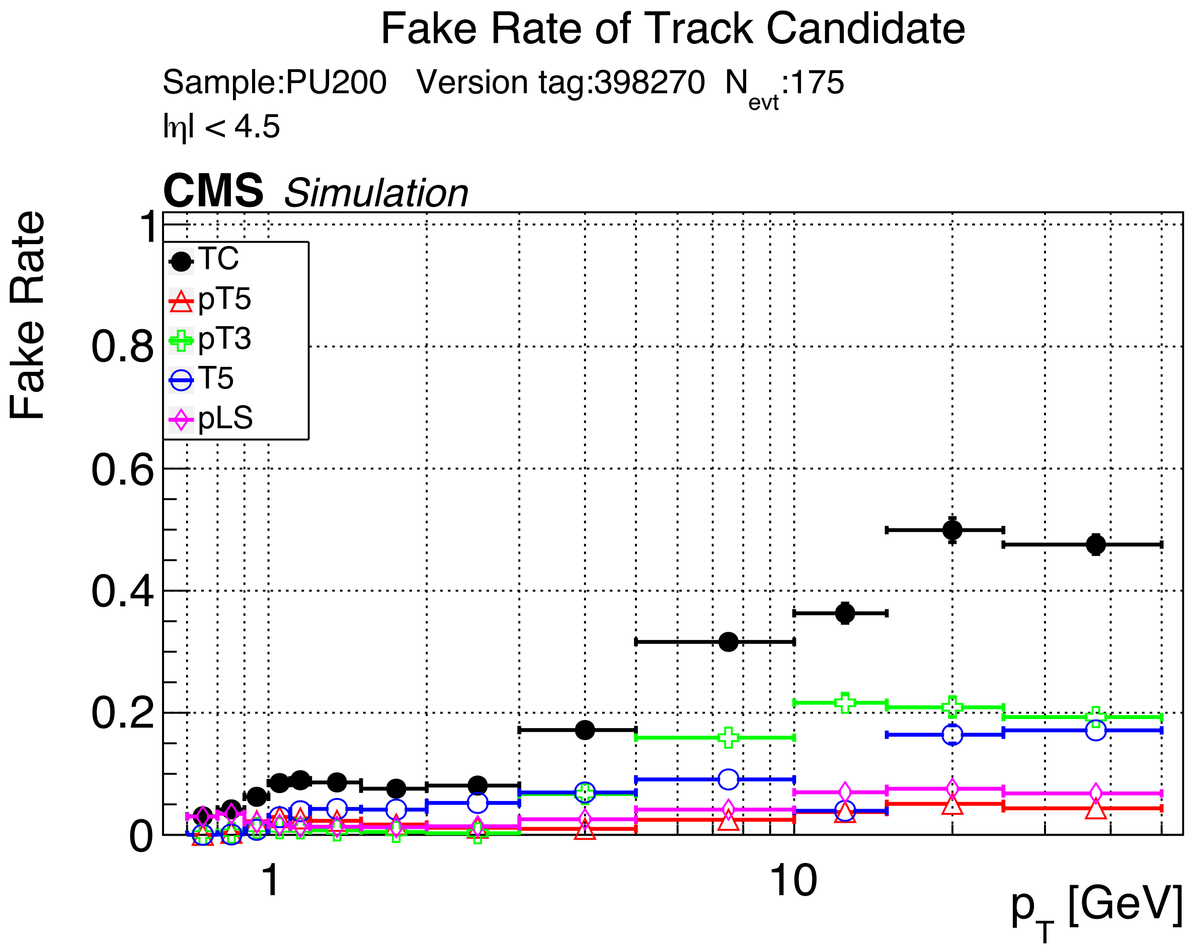


UC San Diego

LST Master Efficiency Plots

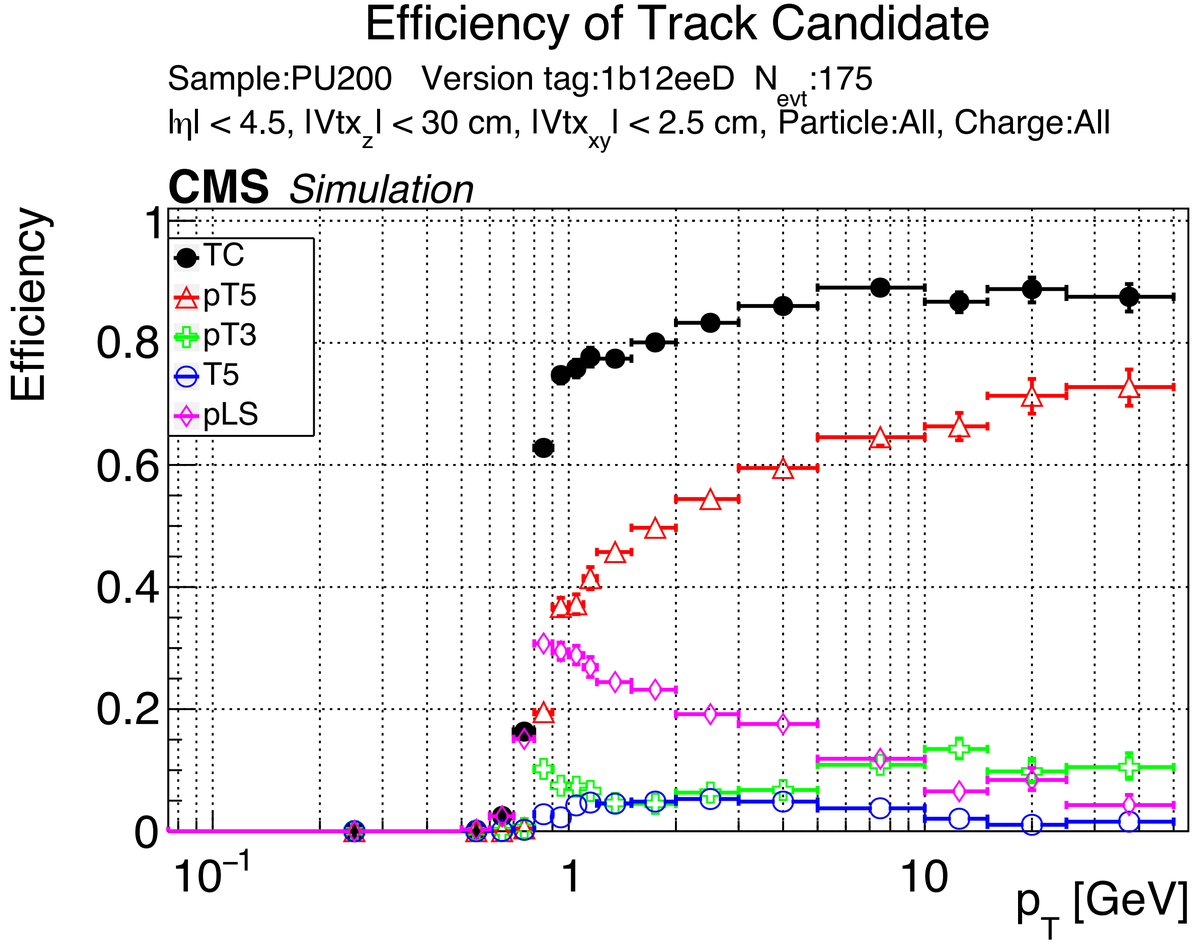
Showing p_T-binned efficiency and fake rate

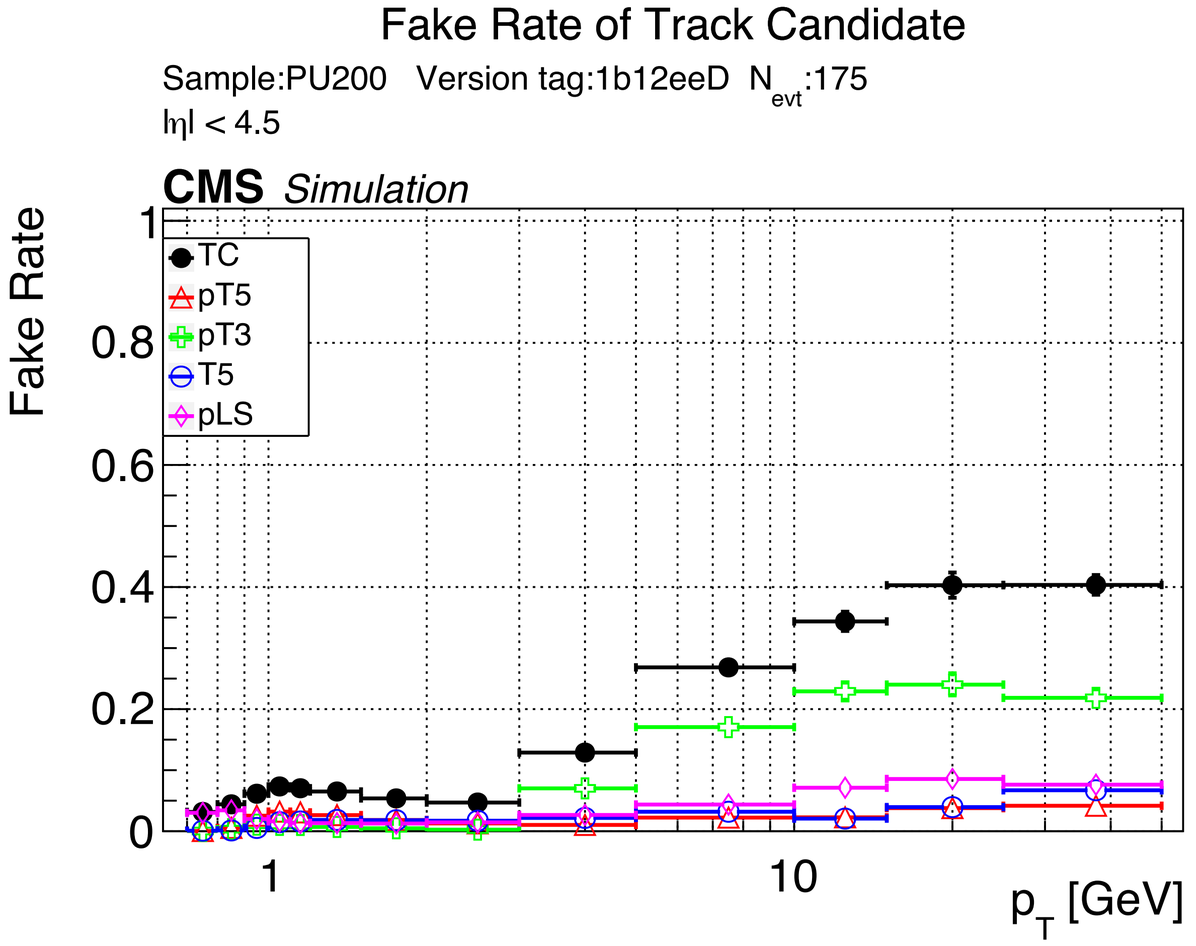




LST DNN Efficiency Plots

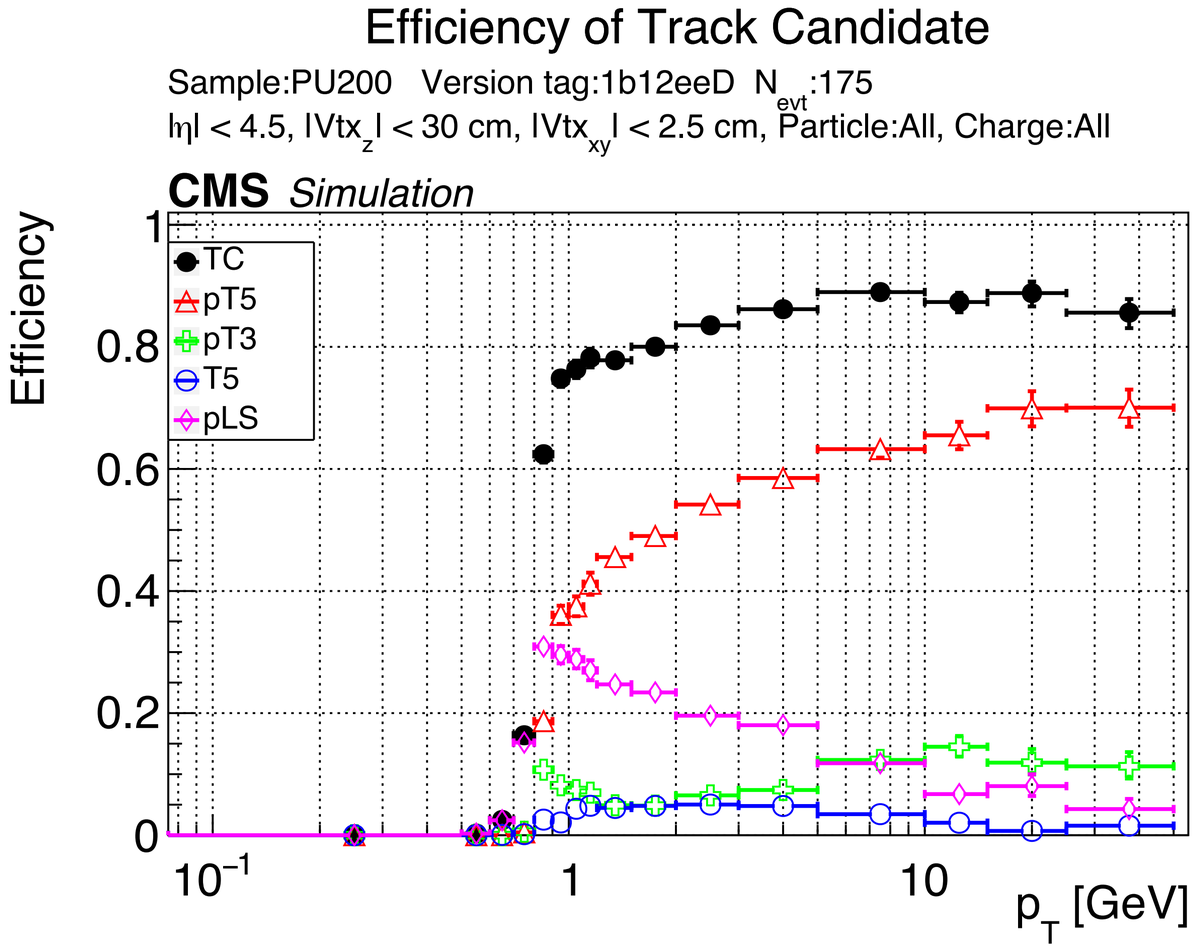
Significant fake rate reduction (▲ WP)!

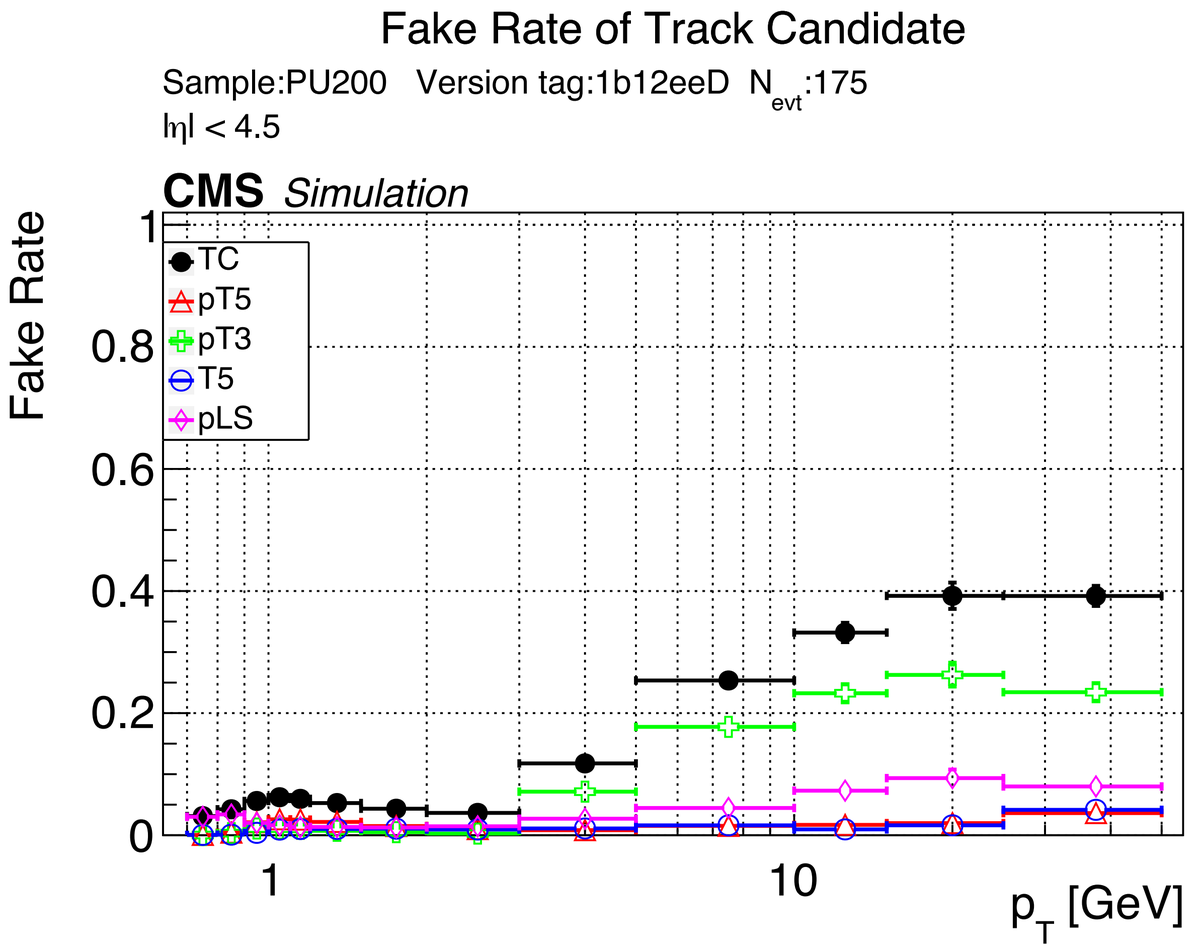




LST DNN Efficiency Plots

Even more fake rate reduction (▼ WP)!

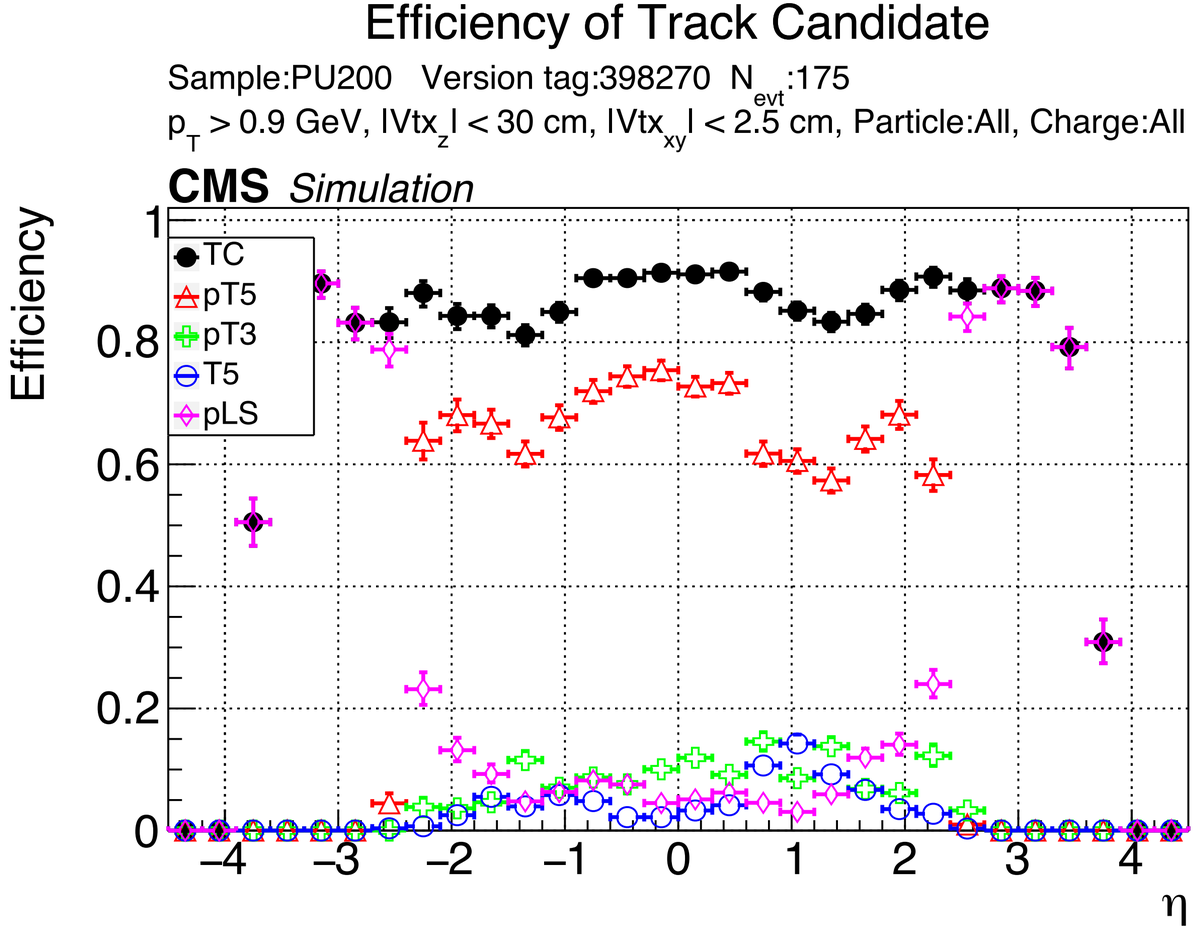


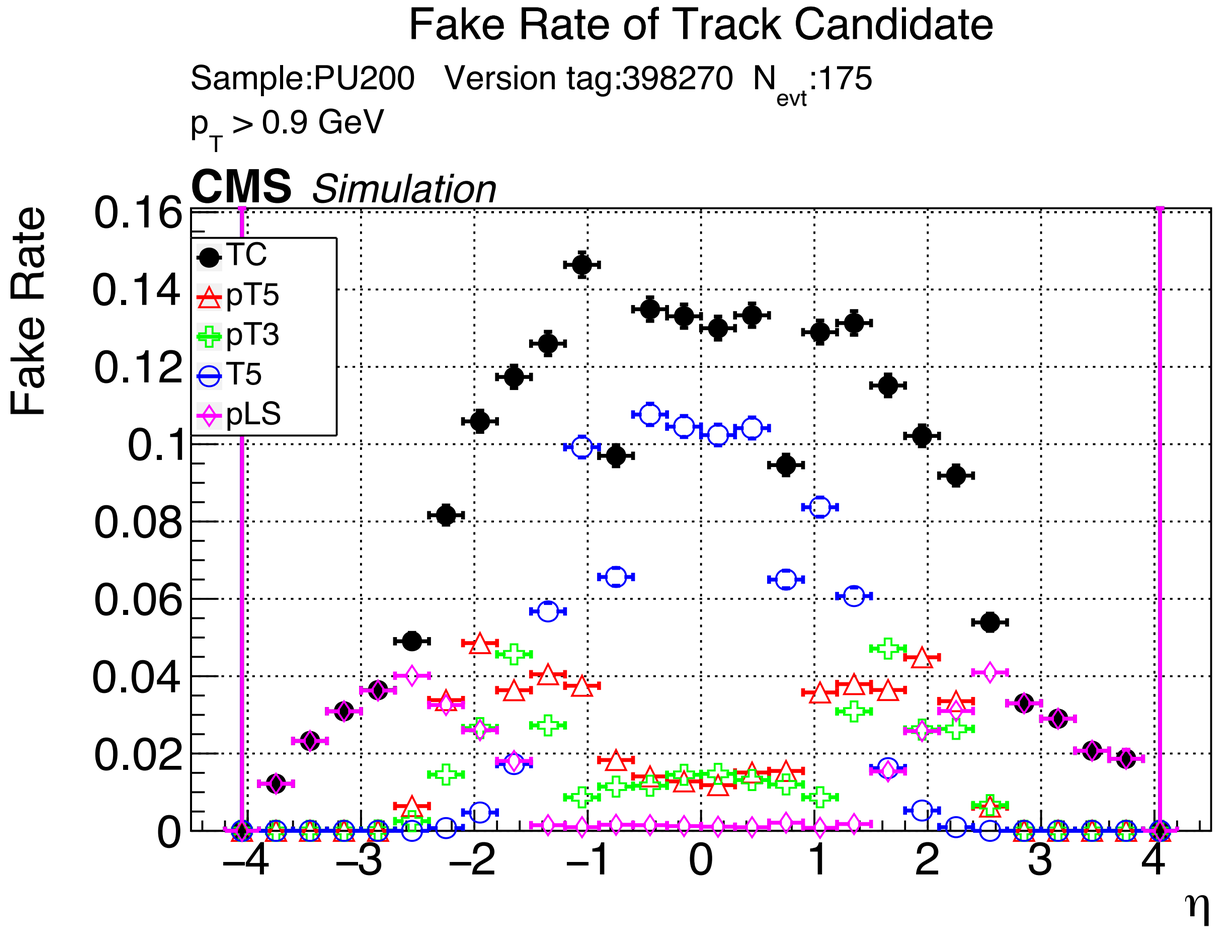


UC San Diego

LST Master Efficiency Plots

Showing η-binned efficiency and fake rate

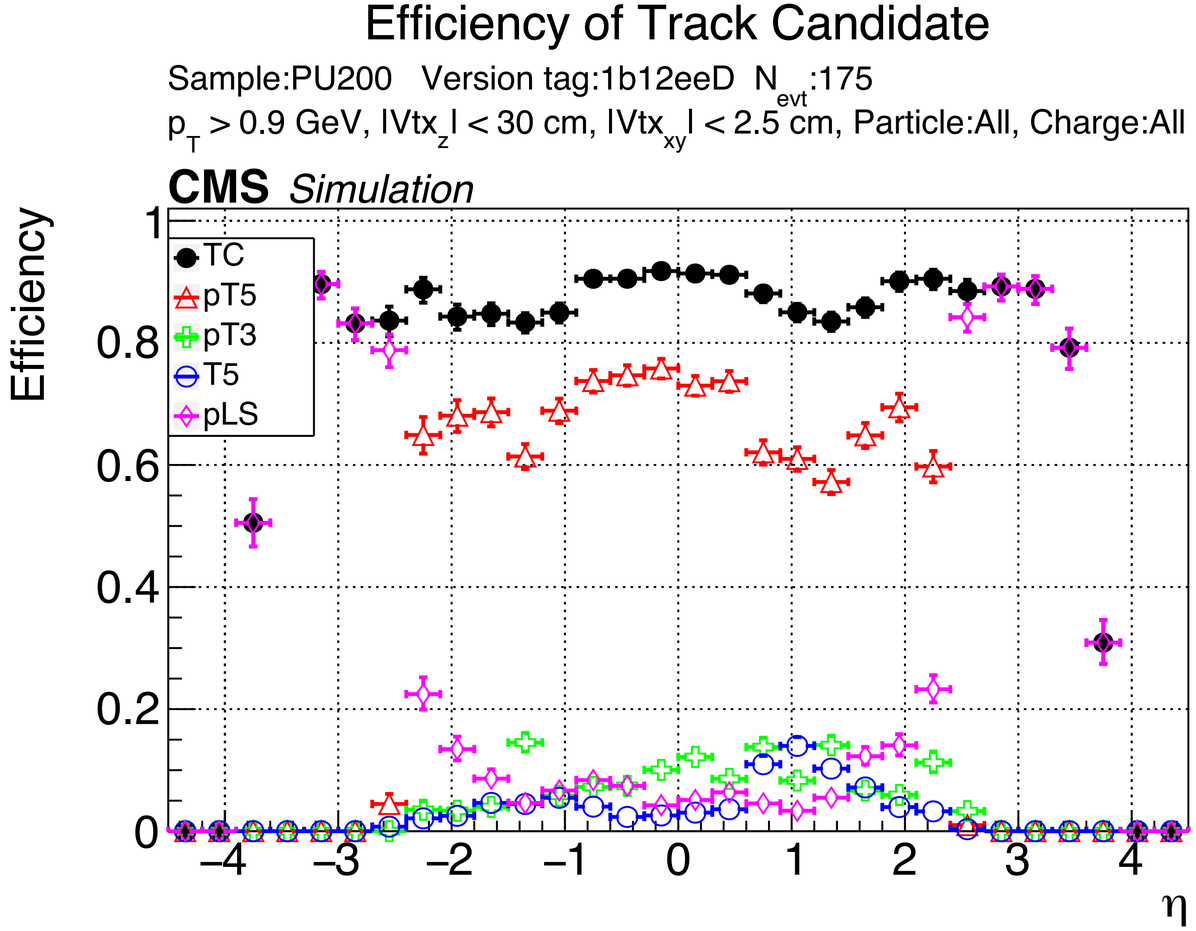


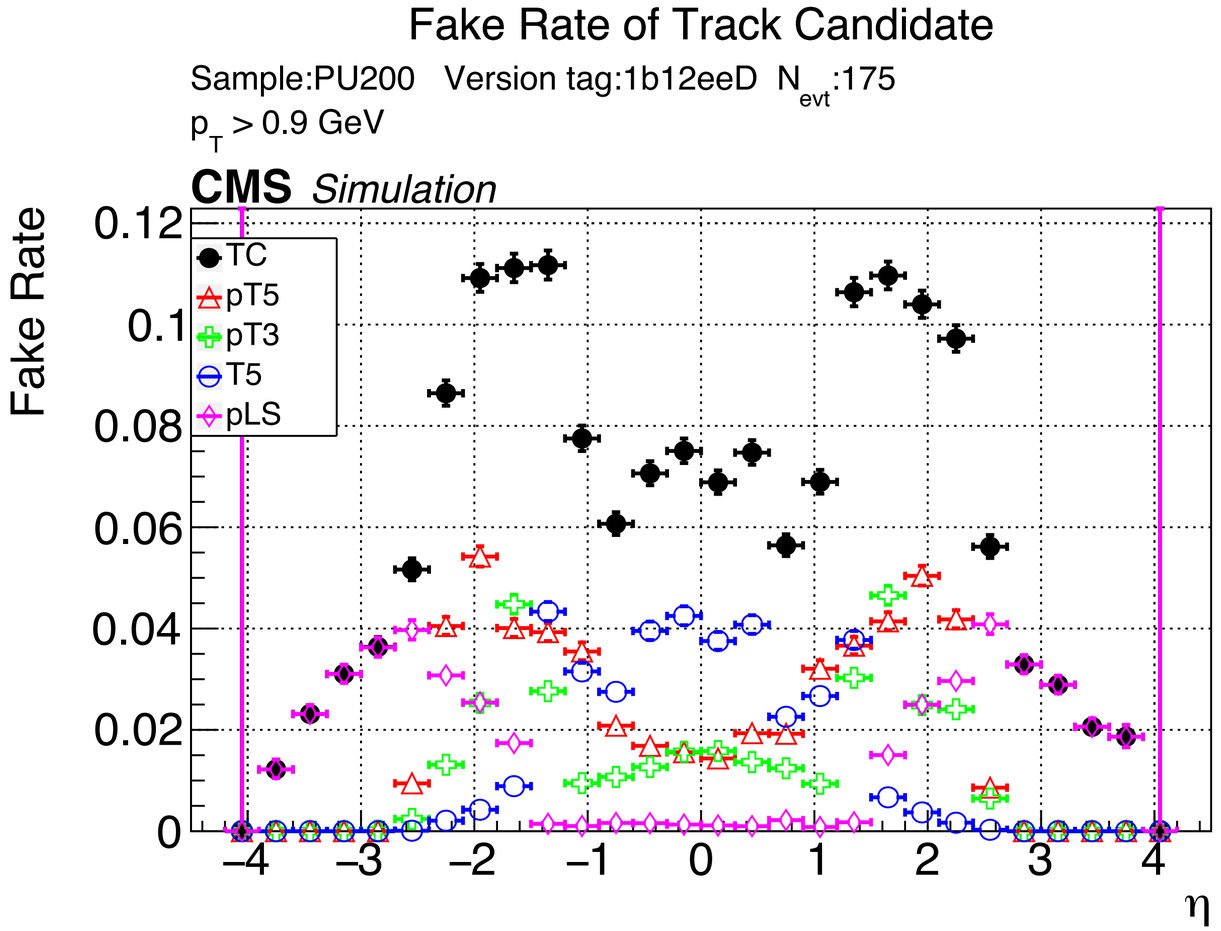


UC San Diego

LST DNN Efficiency Plots

Significant fake rate reduction (▲ WP) **in the barrel!**

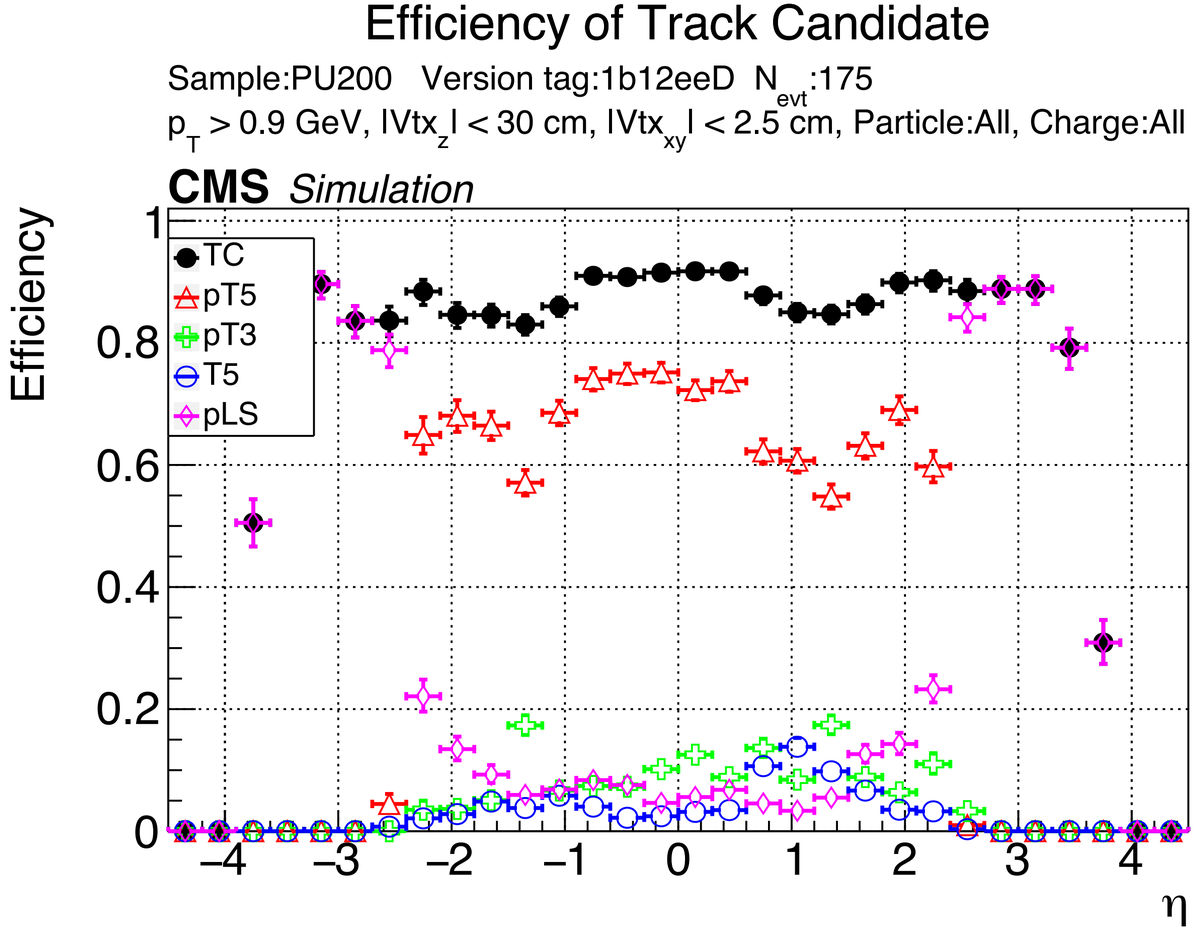


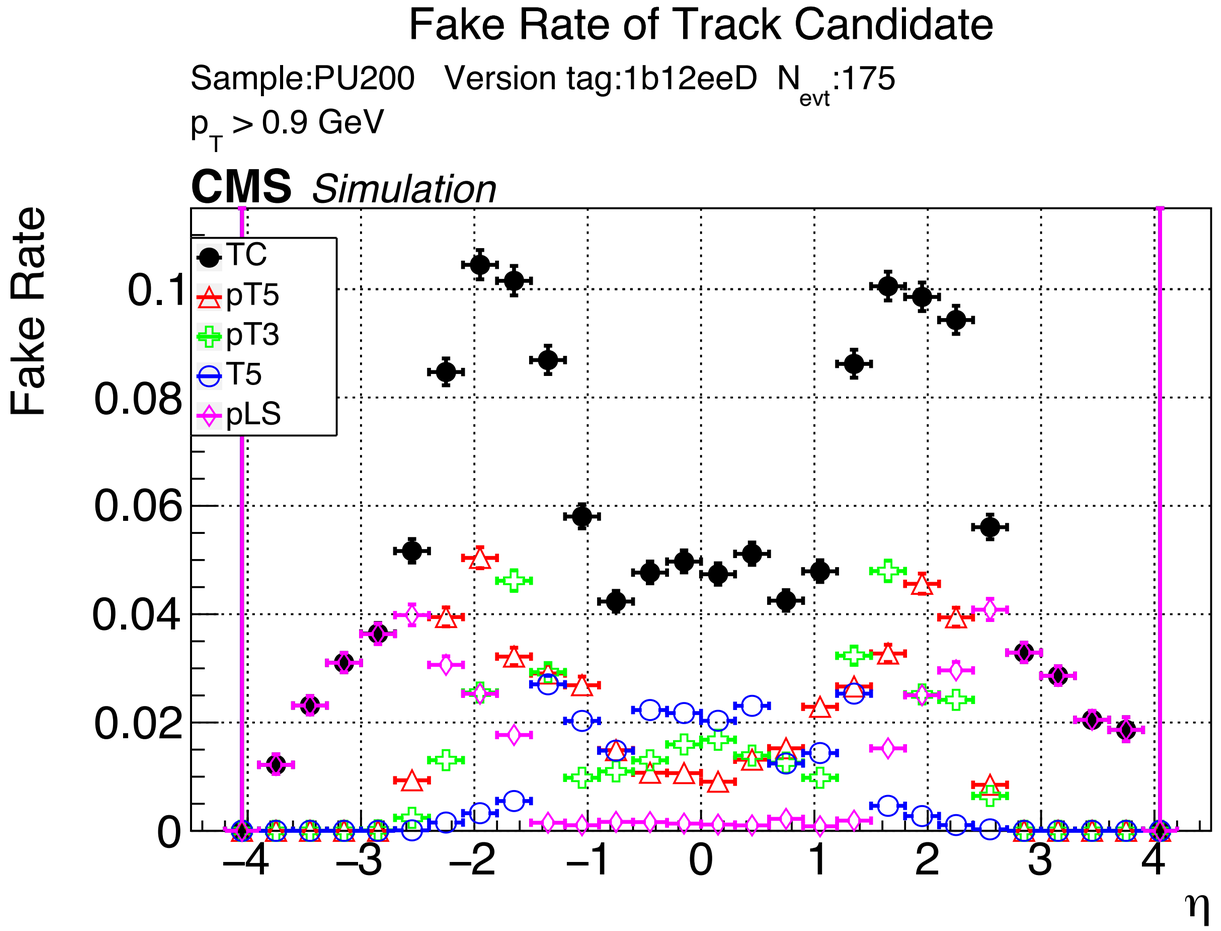


UC San Diego

LST DNN Efficiency Plots

Even more fake rate reduction (▼ WP) **in the barrel!**





UC San Diego

LST Timing

LST does run slower with the DNN

DNN

Evt	Hits	MD	LS	T3	T5	pLS	pT5	pT3	TC	Event	Short		Rate	
avg	8.5	2.0	1.4	3.7	6.7	1.4	1.5	1.1	2.1	28.5	18.6+/-	3.9	32.1	explicit[s=1]
avg	12.5	3.3	1.8	6.2	10.0	1.5	4.0	2.5	3.4	45.3	31.3+/-	6.6	27.2	explicit[s=2]
avg	25.1	4.6	2.7	10.2	16.2	1.7	9.5	5.1	5.0	80.0	53.3+/-	10.3	28.8	explicit[s=4]
avg	34.9	4.5	2.7	13.7	22.6	1.9	17.3	8.1	5.8	111.6	74.9+/-	18.3	27.7	explicit[s=6]
avg	56.1	4.6	2.5	17.5	30.8	1.7	25.5	13.6	5.8	158.1	100.3+/-	27.9	29.1	explicit[s=8]
avg	1.6	2.0	1.4	3.3	6.1	1.4	1.3	1.0	2.1	20.3	17.3+/-	3.9	21.9	explicit_cache[s=1]
avg	4.7	3.7	2.0	5.4	8.1	1.5	3.0	1.9	3.6	33.8	27.7+/-	6.4	19.5	explicit_cache[s=2]
avg	5.5	5.4	3.3	10.5	12.8	1.7	7.0	3.3	6.3	55.8	48.5+/-	10.2	15.1	explicit_cache[s=4]
avg	11.6	7.3	4.7	14.9	17.8	2.2	11.0	5.4	9.6	84.5	70.7+/-	15.8	15.1	explicit_cache[s=6]
avg	17.1	8.7	5.9	20.0	24.0	2.8	15.1	7.5	12.9	114.1	94.2+/-	18.8	15.1	explicit_cache[s=8]

Master

Evt	Hits	MD	LS	T3	T5	pLS	pT5	pT3	TC	Event	Short		Rate	
avg	8.9	2.0	1.4	3.7	4.1	1.4	1.5	1.1	2.2	26.2	15.9+/-	3.3	29.4	explicit[s=1]
avg	13.6	3.5	1.9	6.3	6.3	1.5	3.6	2.5	3.7	43.0	27.9+/-	5.7	27.1	explicit[s=2]
avg	20.2	3.7	2.3	10.0	12.2	1.5	8.8	5.1	4.6	68.4	46.6+/-	10.2	25.8	explicit[s=4]
avg	31.1	5.1	2.9	12.9	15.6	1.8	14.1	7.8	5.9	97.2	64.3+/-	16.4	24.6	explicit[s=6]
avg	49.5	4.1	2.5	16.5	26.5	1.7	23.3	12.9	5.6	142.5	91.3+/-	27.1	26.3	explicit[s=8]
avg	1.6	2.0	1.5	3.4	3.4	1.4	1.4	1.0	2.2	17.9	14.9+/-	3.5	19.1	explicit_cache[s=1]
avg	3.4	3.1	1.8	4.9	4.6	1.5	2.7	1.7	3.5	27.2	22.3+/-	5.9	16.2	explicit_cache[s=2]
avg	5.8	5.1	3.0	8.9	8.3	1.8	5.8	3.0	6.3	47.9	40.4+/-	7.8	12.9	explicit_cache[s=4]
avg	10.8	6.3	3.9	12.2	11.6	2.1	8.7	4.7	9.1	69.5	56.5+/-	12.2	12.6	explicit_cache[s=6]
avg	15.2	8.3	5.3	16.6	16.6	2.3	12.7	6.1	12.1	95.1	77.6+/-	11.1	12.3	explicit_cache[s=8]

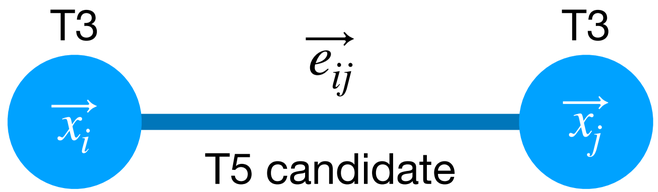
ΔT5 = (1-DNN/master)
63%
59%
33%
45%
16%
79%
76%
54%
53%
45%

Summary

- Bugs plaguing previous DNN implementation have been squashed
- DNN indeed gives a significant fake rate reduction
  - ▲ WP (92% sig eff, 44% bkg eff): good FR reduction
  - ▼ WP (82% sig eff, 24% bkg eff): best FR reduction w/ small loss of overall eff
- DNN also leads to significantly longer runtime for T5 building as implemented currently
- Next steps:
  - Try to slim `runQuintupletDefaultAlgo` as much as possible
  - Improve DNN performance (can possibly remove 1 hidden layer with better inputs)

Backup

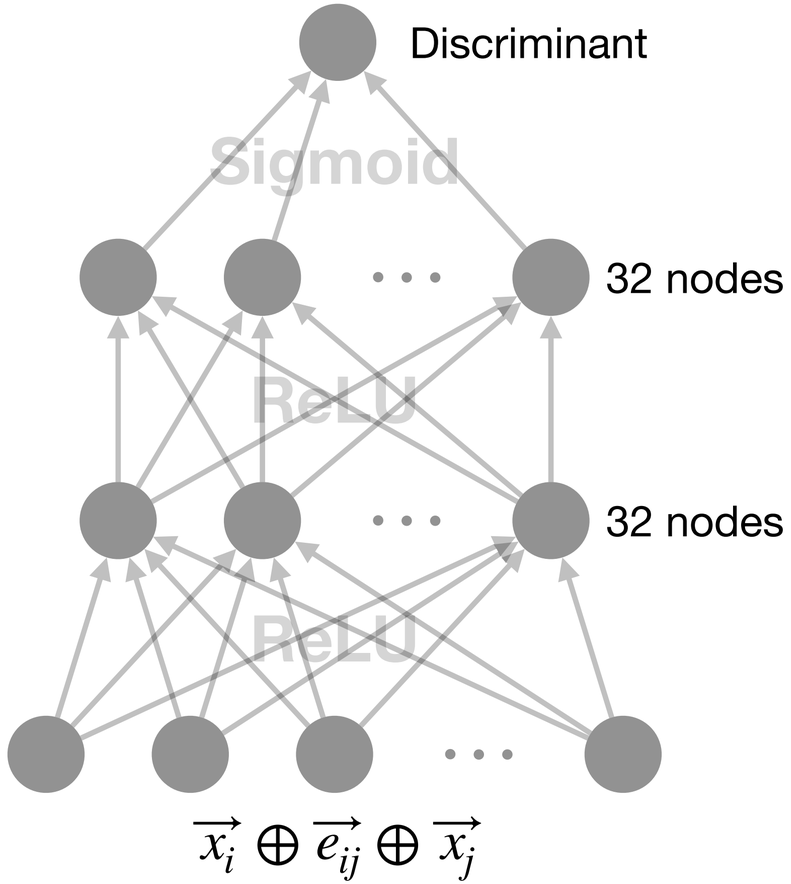
T5 DNN Configuration



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

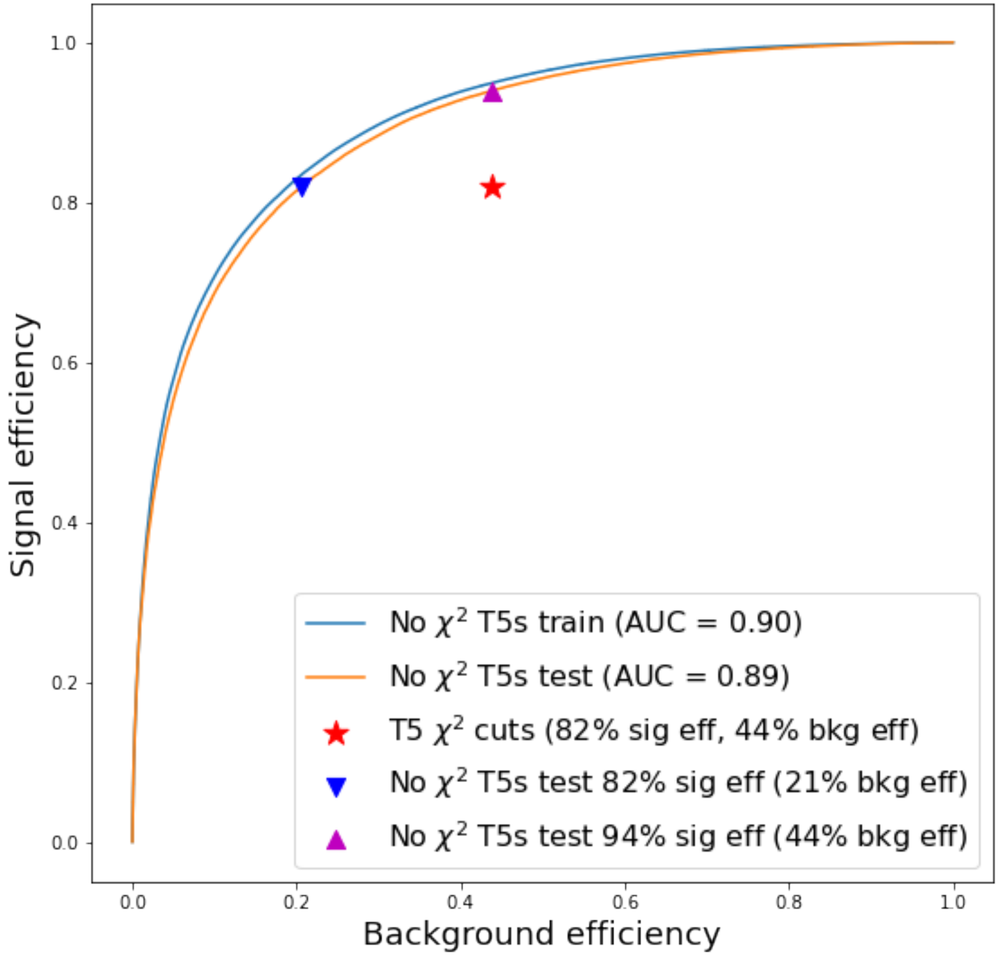


UC San Diego

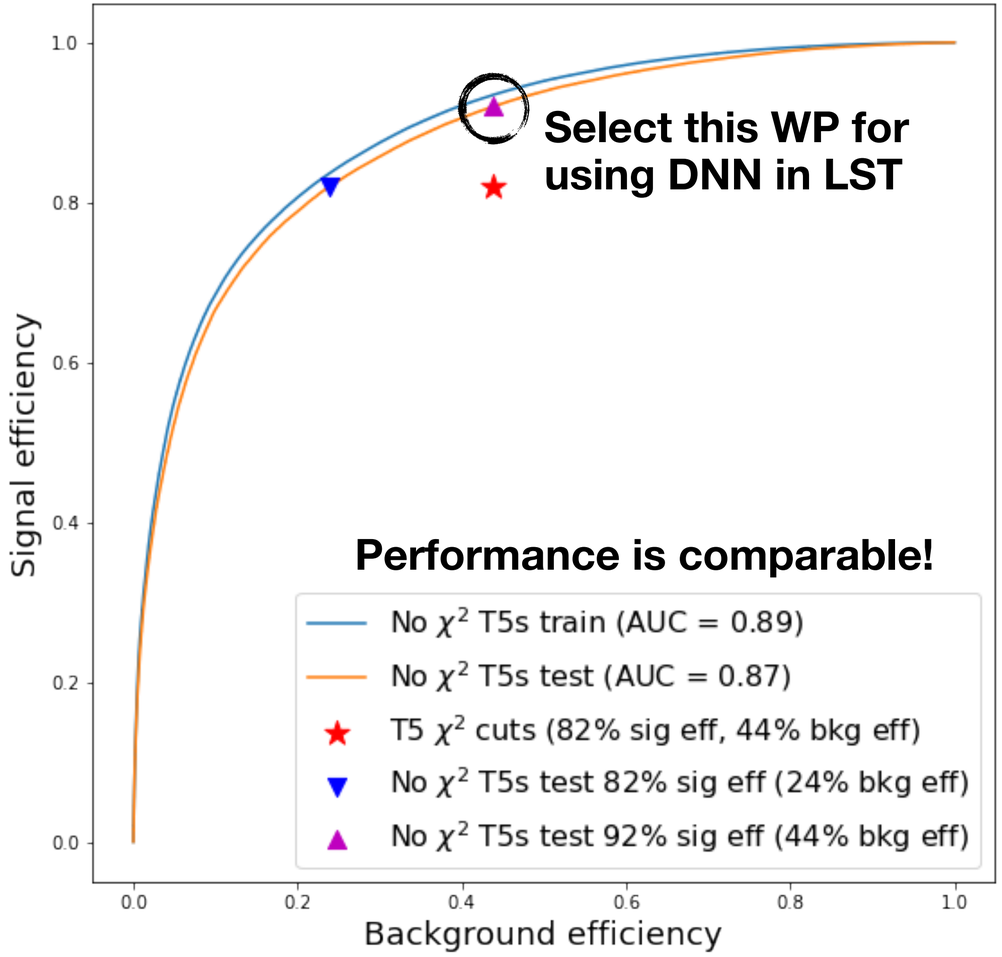
T5 DNN Performance

Performance is comparable to previous results

2 hidden layers (old result)



2 hidden layers + no δr + p_T fix

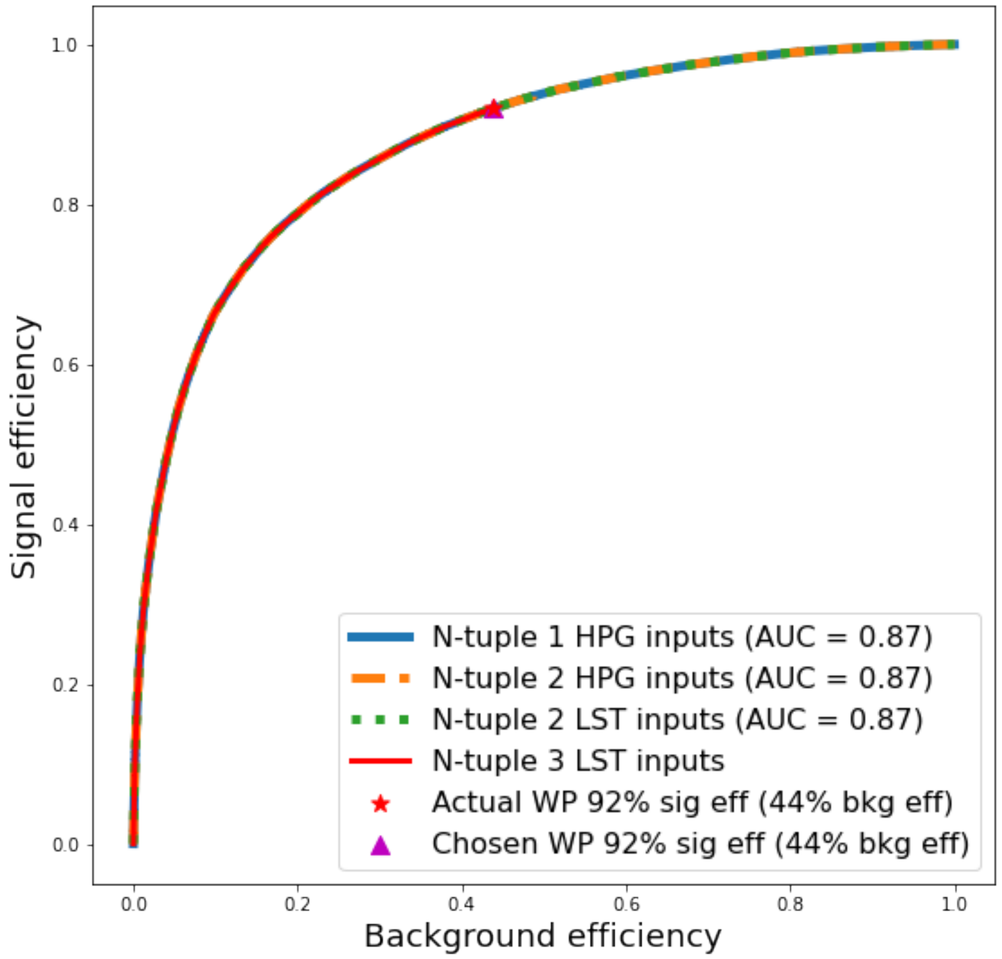


T5 DNN Results

It works…?

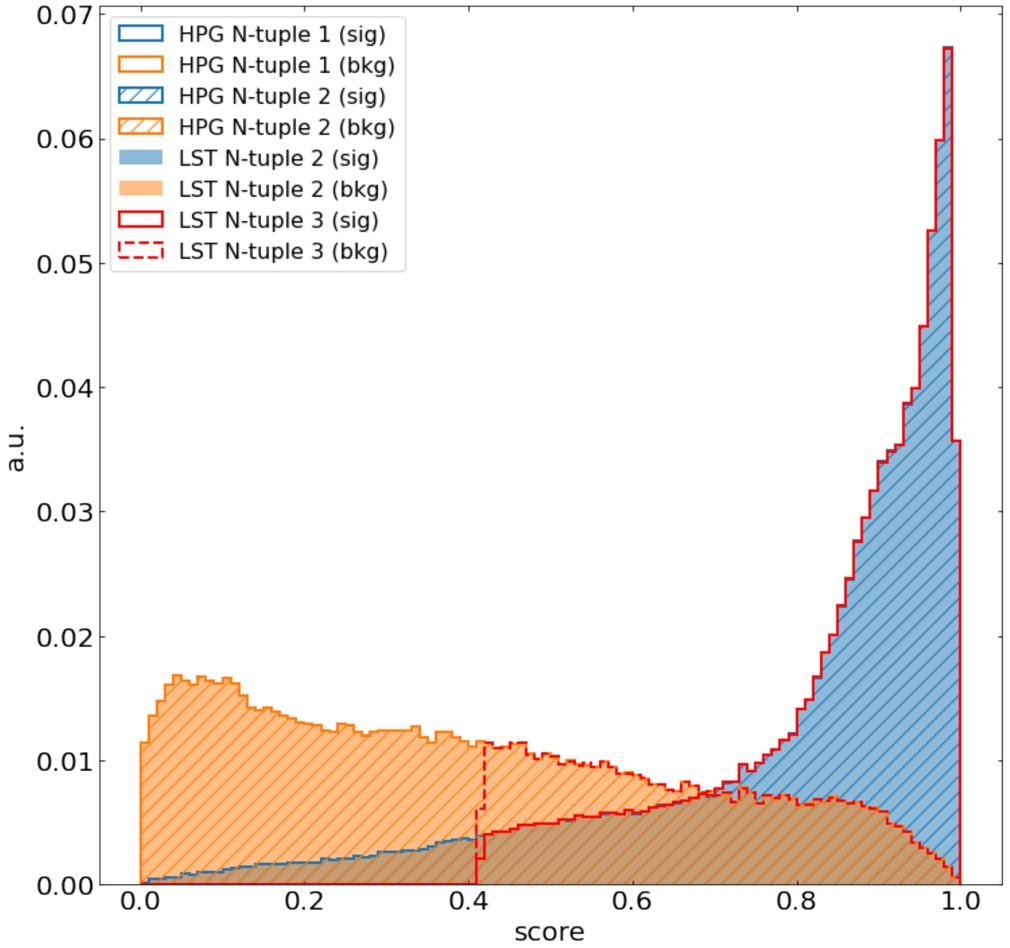
- — curve: $FPR_3 \times (N_3/N_2)$ vs. $TPR_3 \times (P_3/P_2)$
- ★ point: N_3/N_2 vs. P_3/P_2
- $FPR_i = FP_i/N_i$
  - FP_i = # false positives for DNN > X (N-tuple i)
  - N_i = # total negatives in N-tuple i
- $TPR_i = TP_i/P_i$
  - TP_i = # true positives for DNN > X (N-tuple i)
  - P_i = # total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

2 hidden layers + no δr + p_T fix

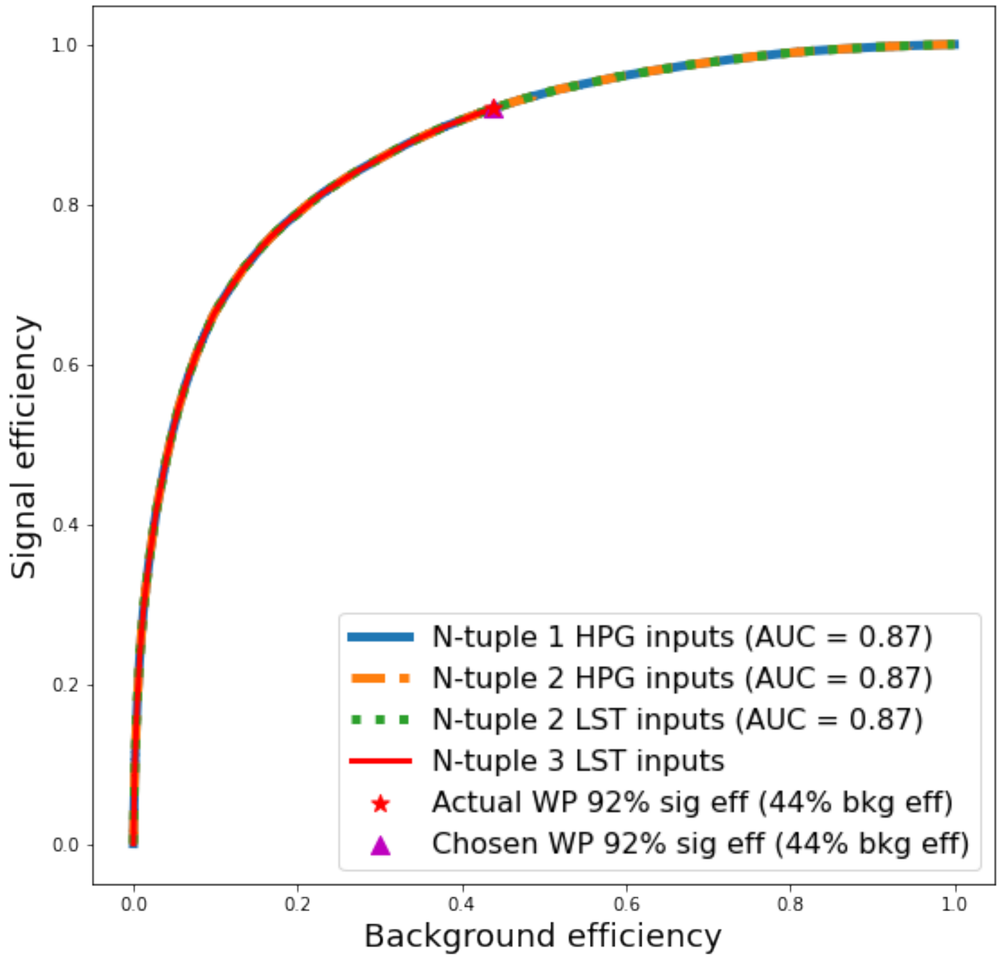


T5 DNN Results

It works!



2 hidden layers + no δr + p_T fix



T5 DNN Results

layers + no δr + p_T fix