AI Safety For High-Energy Physics

arXiv:1910.08606

Chase Shimmin & Ben Nachman

ML4Jets, January 17th 2020

What is AI Safety?

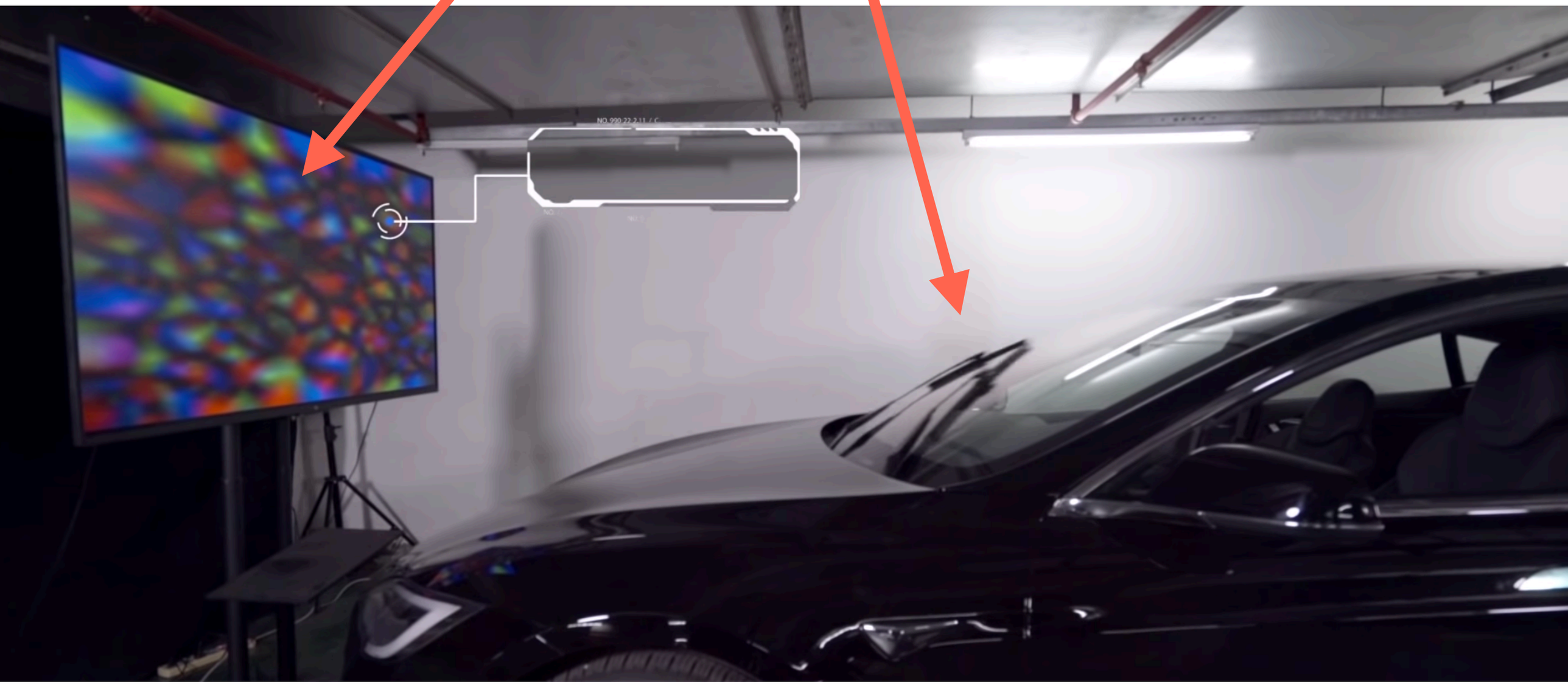
✓ **We can all agree, we should try not to get murdered by robots**

What is AI Safety?

For starters, let’s not get killed by this robot!

Tesla Adversary

Wipers engaged by carefully-chosen noise



Tencent https://www.youtube.com/watch?v=6QSsKy0I9LE

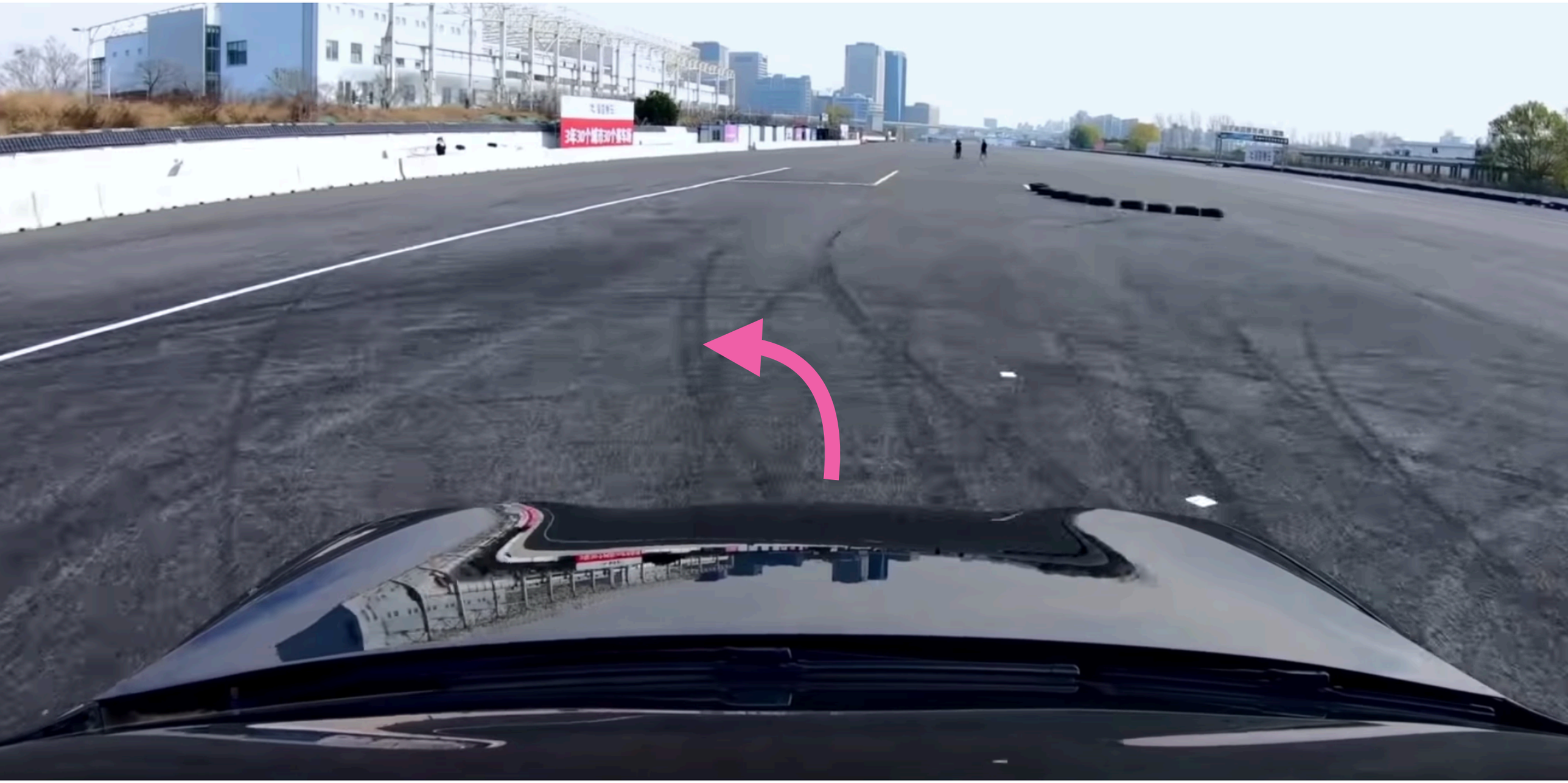
Tesla Adversary

Stickers placed on pavement at intersection

Tencent https://www.youtube.com/watch?v=6QSsKy0I9LE

Tesla Adversary

Autopilot veers into oncoming lane!!



Tencent https://www.youtube.com/watch?v=6QSsKy0I9LE

Tesla Autopilot Attack

Nearly imperceptible artifacts can create imaginary lanes, traffic signs, etc:

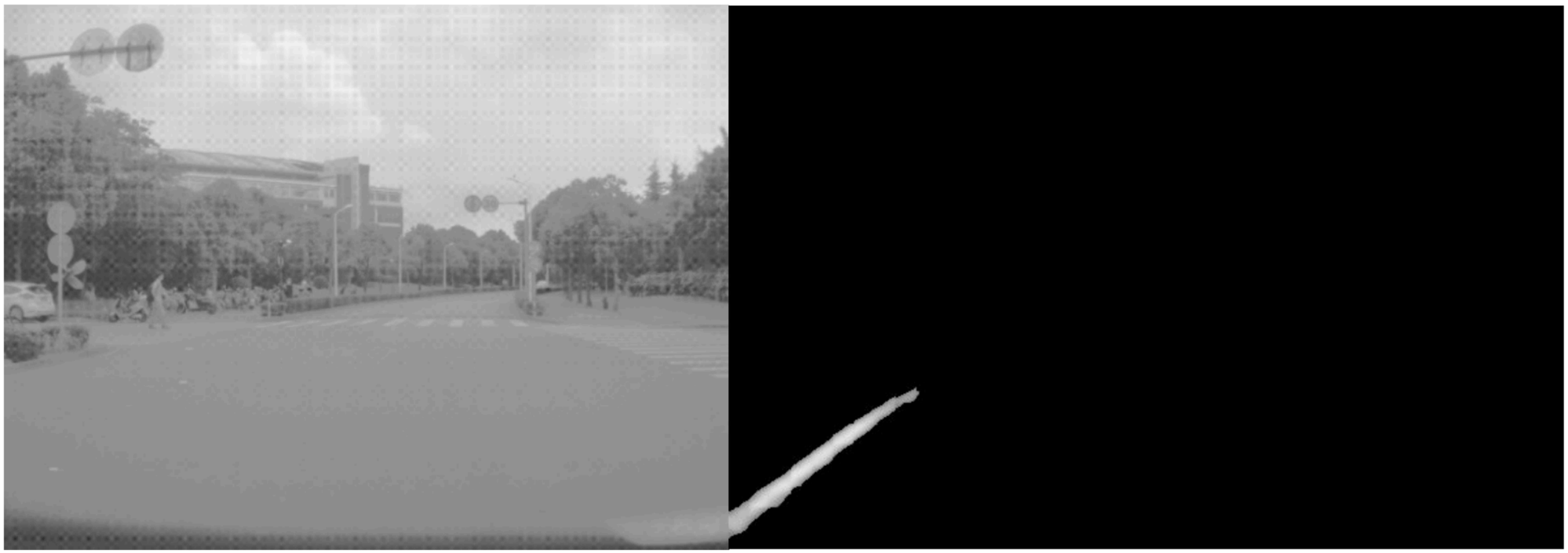
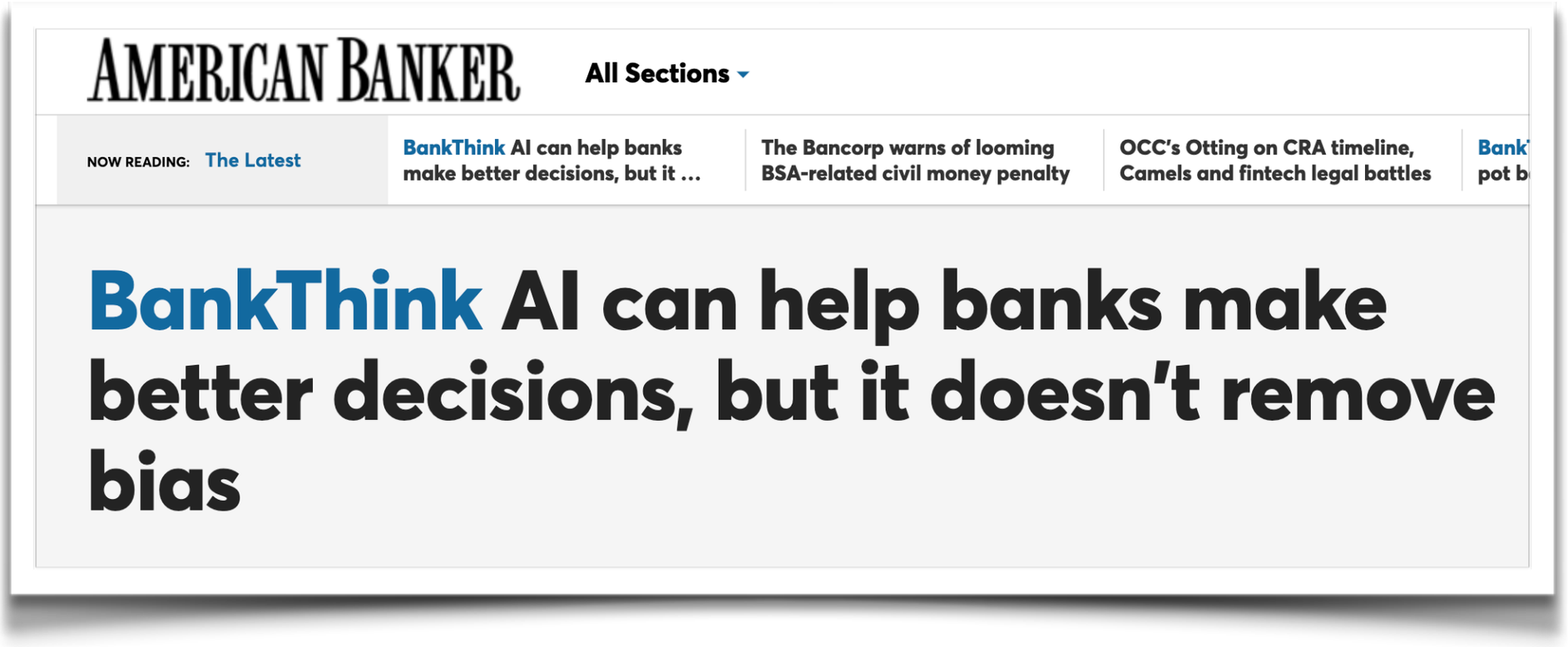


Fig 33. Fake lane in digital level

[Whitepaper by Tencent Security Lab]

AI Ethics

But it's not all just killer robots!



AI Ethics

But it's not all just killer robots!

AMERICAN BANKER All Secti

NOW READING: The Latest

BankThink AI can help banks make better decisions, but it ..

BankThink AI ca better decisions, bias

MIT Researcher Exposing Bias in Facial Recognition Tech Triggers Amazon's Wrath

By Matt O'Brien | April 8, 2019

AI Ethics

But it's not all just killer robots!

AMERICAN BANKER

All Sectio

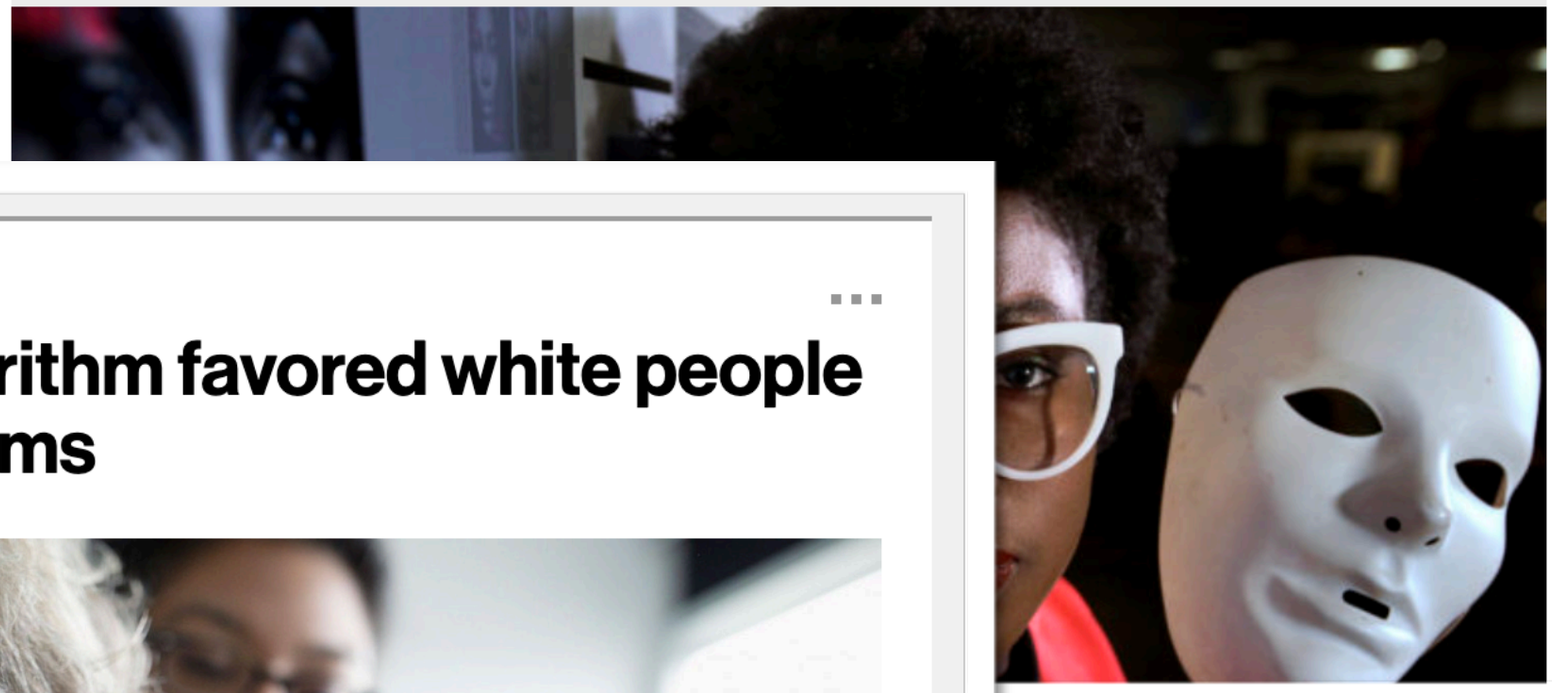
NOW READING: The Latest

BankThink AI can help banks make better decisions, but it ..

BankThink AI ca
better decisions,

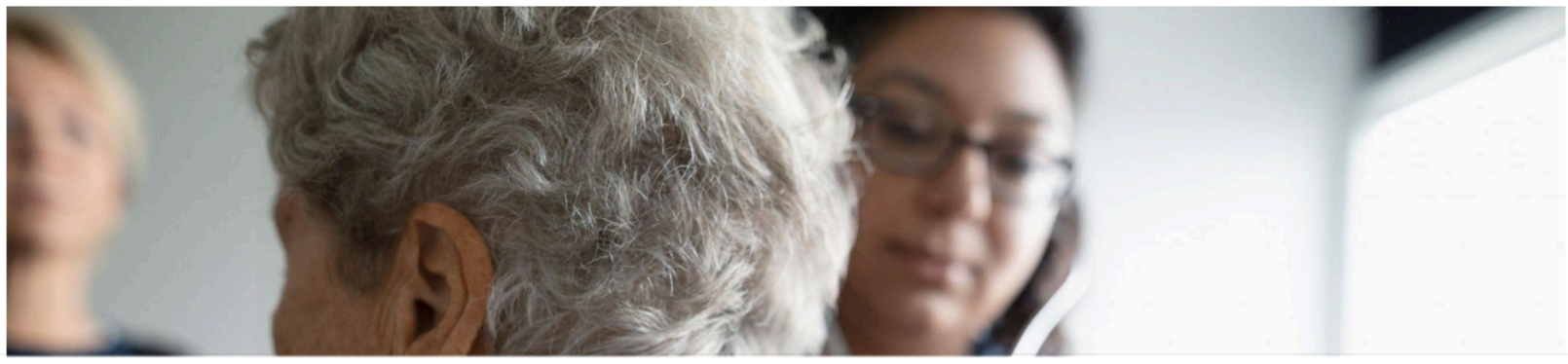
MIT Researcher Exposing Bias in Facial Recognition Tech Triggers Amazon's Wrath

By Matt O'Brien | April 8, 2019



Artificial Intelligence Oct 25

A biased medical algorithm favored white people for health-care programs



AI Ethics

But it's not all just killer robots!

AMERICAN BANKER

All Sectio

NOW READING: The Latest

BankThink AI can help banks make better decisions, but it ..

BankThink AI ca
better decisio

Artificial Intelligence Oct 25

A biased medic
for health-care

MIT Researcher Exposing Bias in Facial Recognition Tech Triggers Amazon's Wrath

By Matt O'Brien | April 8, 2019

Tech Policy / AI Ethics

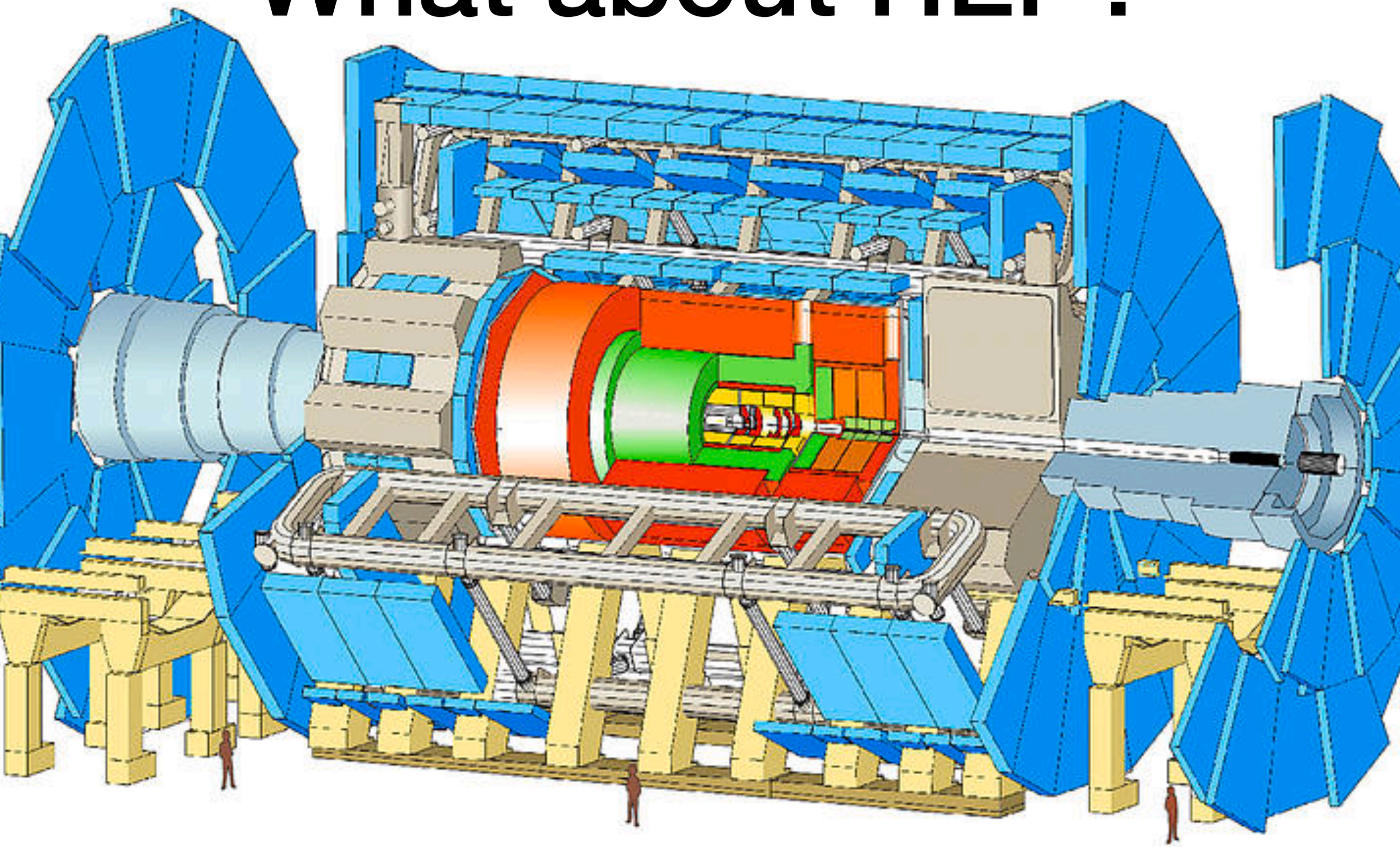
AI is sending people to jail —and getting it wrong

Using historical data to train risk assessment tools could mean that machines are copying the mistakes of the past.

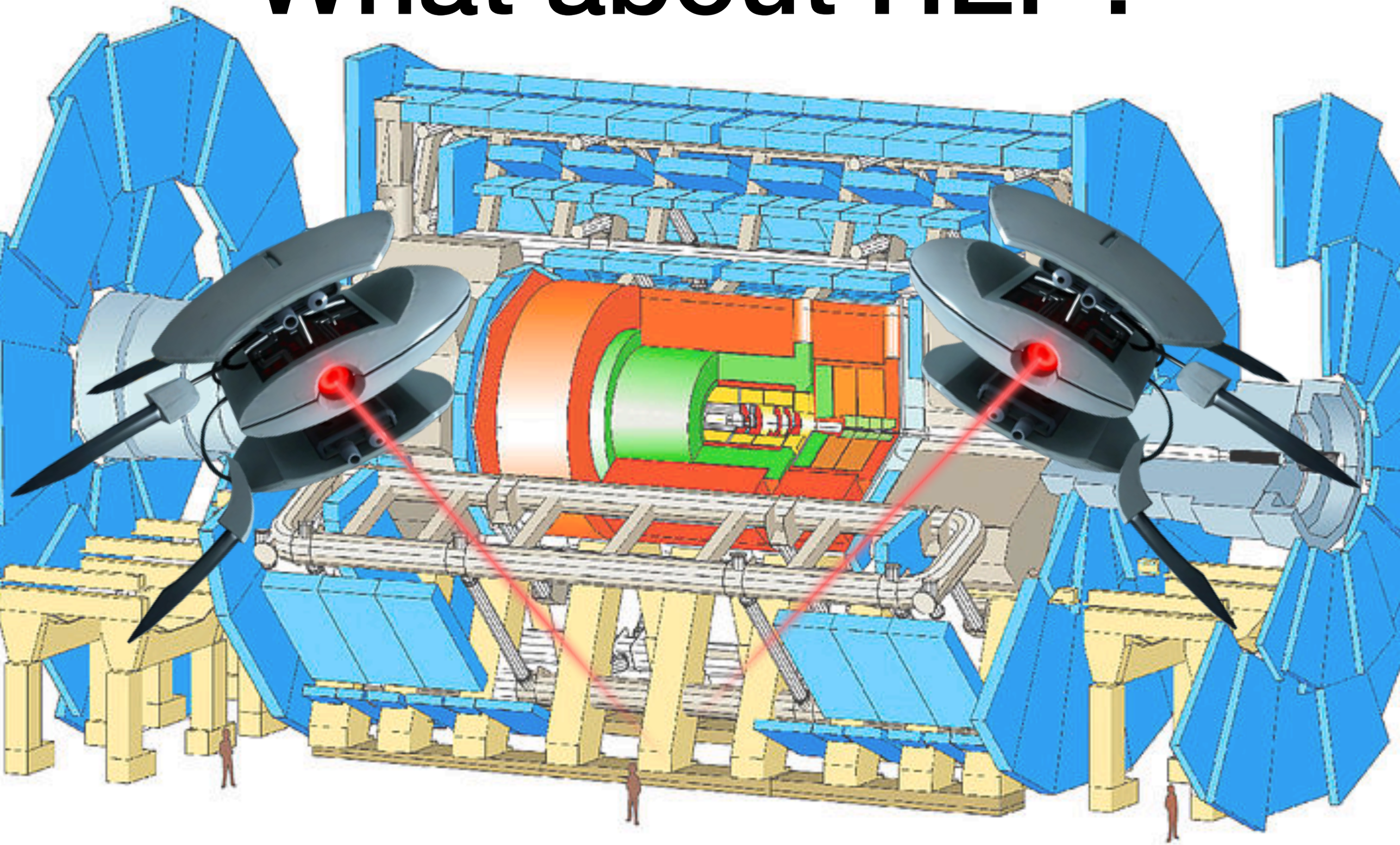
by **Karen Hao**

Jan 21, 2019

What about HEP?



What about HEP?

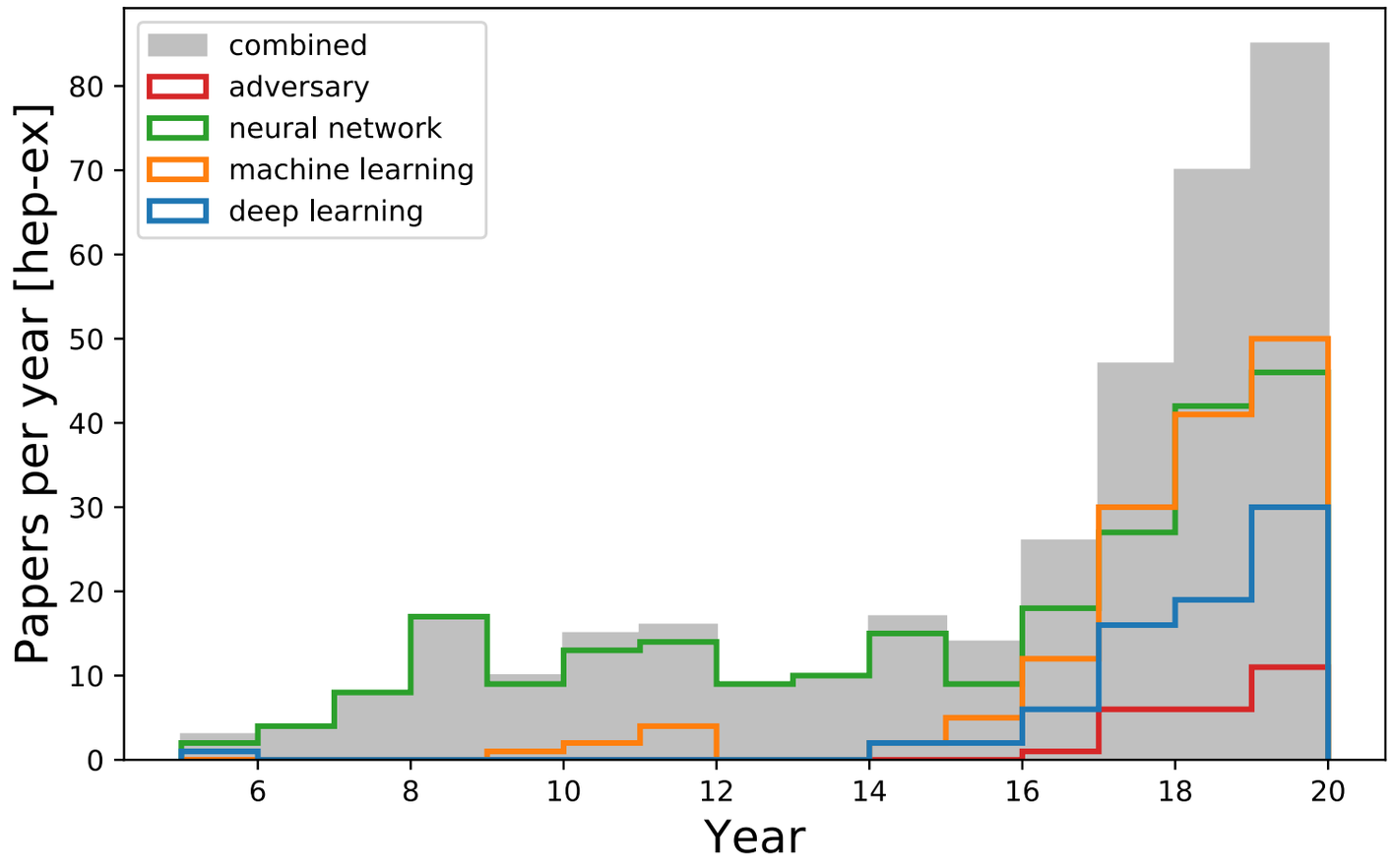


AI & HEP

- Deep learning is becoming increasingly pervasive in our field

- We should at least *consider* whether some of the issues identified in other fields applies to us

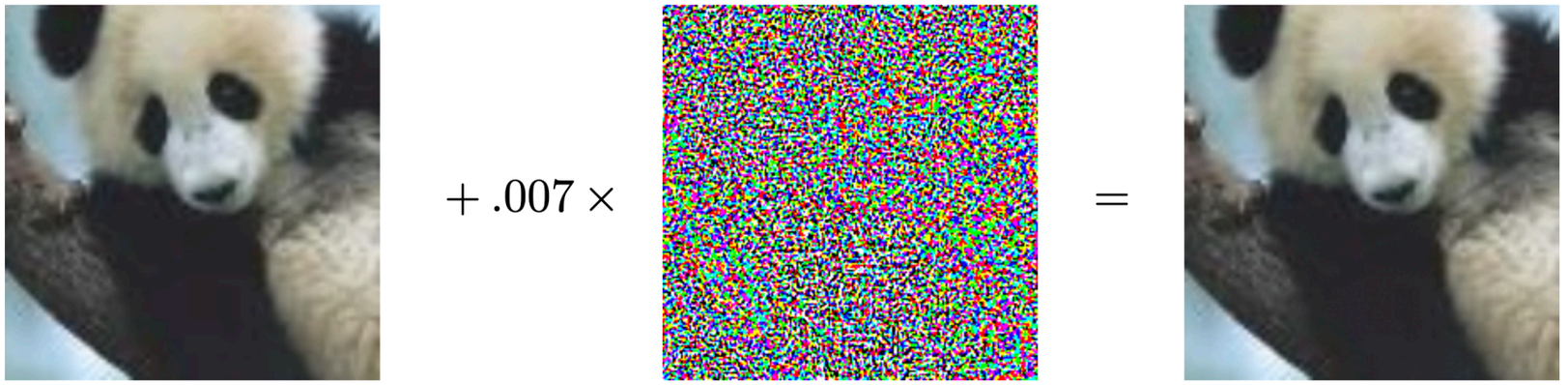
- In particular, we investigate whether there exists some kind of ***systematic bias*** that could affect ***science results***



[*data I scraped from arXiv in November*]

Adversarial Examples

Fast Gradient Sign Method: $\boldsymbol{\eta} = \epsilon \text{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right)$



$\boldsymbol{x}$

"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

- Goodfellow *et al.* show formally that the sensitivity of a network to these types of attack **scales with input dimensionality**

- Empirical evidence also suggests that **deep networks** are more susceptible

arXiv:1412.6572

Adversarial Examples

What happens if we do this to, say, a jet, when shown to a QCD/Boson tagger?

$$J = \{(p_{\mathrm{T}}^k, \eta^k, \phi^k) : k = 1 \ldots N_J\}$$

Adversarial Examples

What happens if we do this to, say, a jet, when shown to a QCD/Boson tagger?

$$J = \{(p_{\mathrm{T}}^k, \eta^k, \phi^k) : k = 1 \ldots N_J\}$$

$$\delta J = \mathrm{sign}[\nabla_J \mathcal{L}_{\mathrm{XE}}(f(J), y_{\mathrm{target}})]$$

** Here, f(J) is the classifier network*

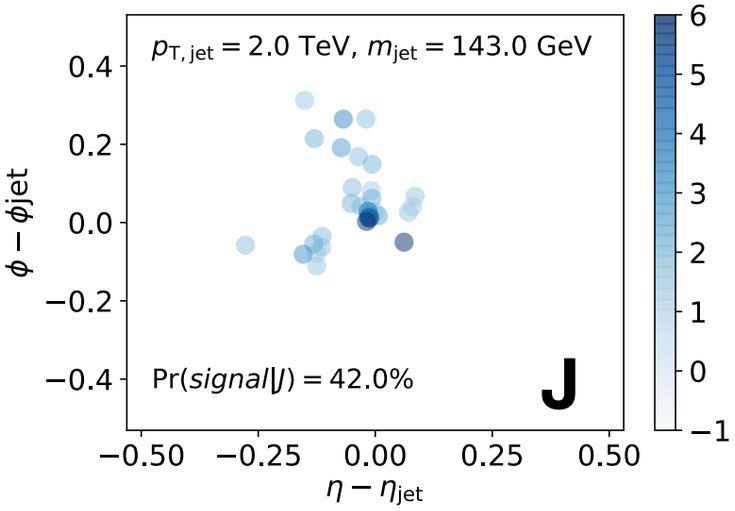
Adversarial Examples

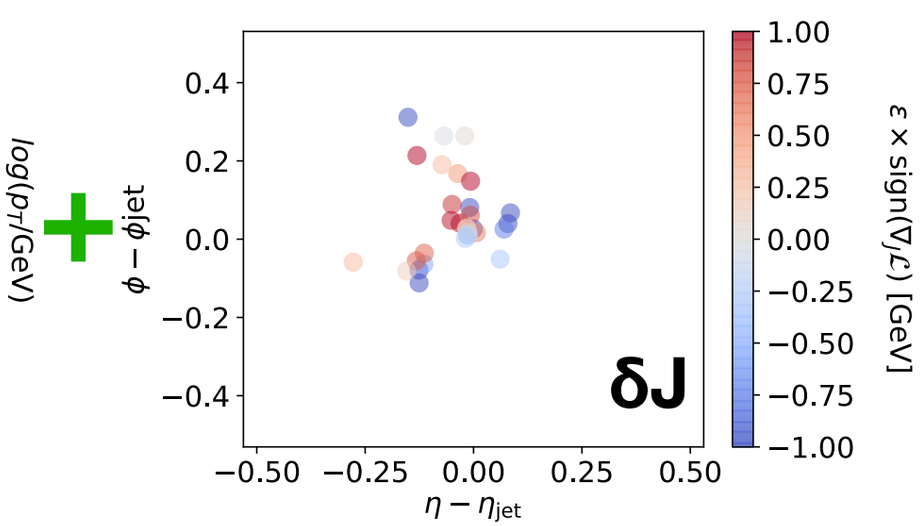
What happens if we do this to, say, a jet, when shown to a QCD/Boson tagger?

$$J = \{(p_{\mathrm{T}}^k, \eta^k, \phi^k) : k = 1 \ldots N_J\}$$

$$\delta J = \mathrm{sign}[\nabla_J \mathcal{L}_{\mathrm{XE}}(f(J), y_{\mathrm{target}})]$$

** Here, f(J) is the classifier network*





p_T = 2.0 TeV
m = 143.0 GeV

Pr(signal | J) = 42%

Perturb pT/eta/phi of constituents **(Scale 0-1 GeV)**

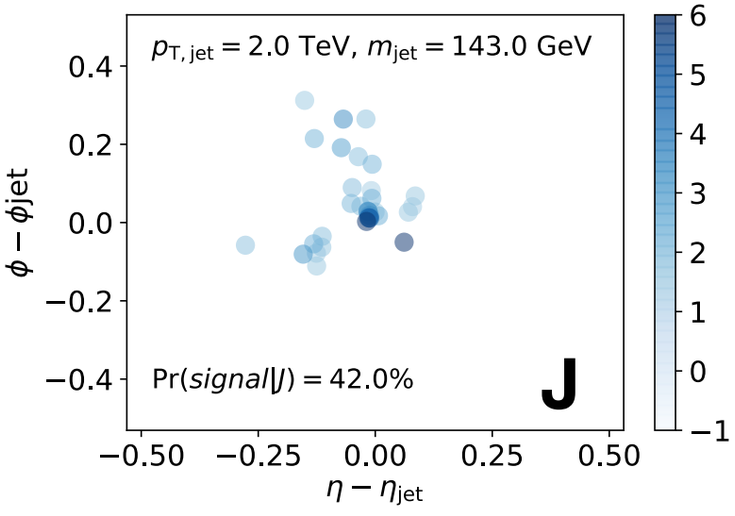
Adversarial Examples

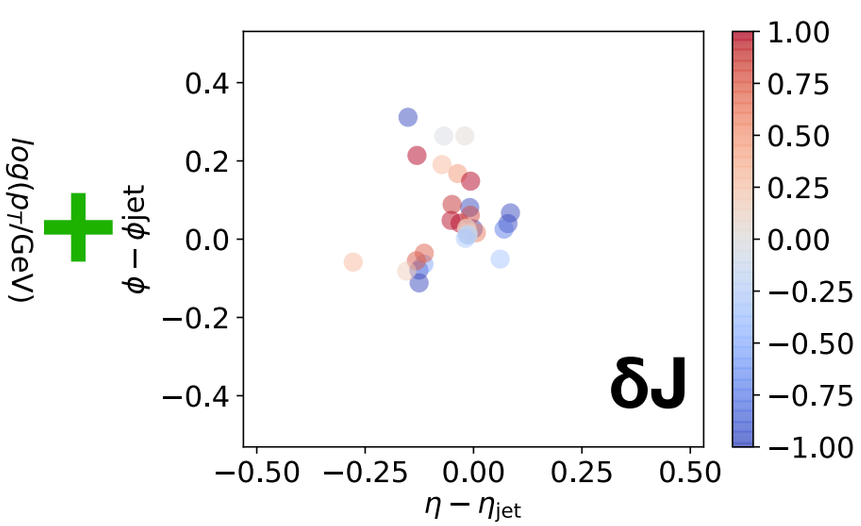
What happens if we do this to, say, a jet, when shown to a QCD/Boson tagger?

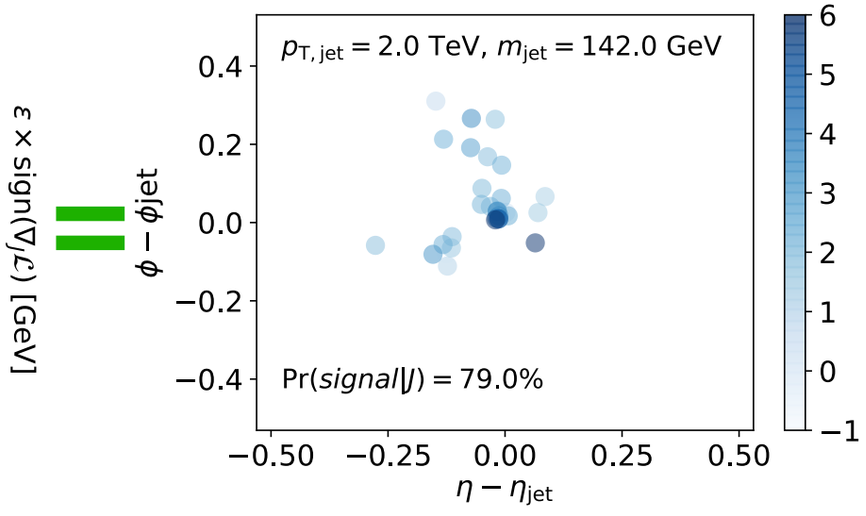
$$J = \{(p_{\mathrm{T}}^k, \eta^k, \phi^k) : k = 1 \ldots N_J\}$$

$$\delta J = \mathrm{sign}[\nabla_J \mathcal{L}_{\mathrm{XE}}(f(J), y_{\mathrm{target}})]$$

** Here, f(J) is the classifier network*







p_T = 2.0 TeV
m = 143.0 GeV

Pr(signal | J) = 42%

Perturb pT/eta/phi of constituents **(Scale 0-1 GeV)**

p_T = 2.0 TeV
m = 142.0 GeV

Pr(signal | J) = 79%

Detector Systematics

- Taking this further, imagine an arbitrary “particle-level” jet *j*
  - Simulation maps these to an observed object $J = D_{\mathrm{sim}}(j)$
  - If this jet is from an LHC collision, the physical ATLAS detector will map it to a different object $J' = D_{\mathrm{det}}(j)$

Detector Systematics

- We often train ML models, and design analyses using *simulated* events
  - Therefore the ***expected*** network response:

$$y_{exp} = f[D_{\mathrm{sim}}(j)]\big|_{j\sim\mathrm{physics}}$$

    is a key part of our statistical models

- However, what is ***observed*** in an analysis signal region is:

$$y_{obs} = f[D_{\mathrm{det}}(j)]\big|_{j\sim\mathrm{physics}}$$

Detector Systematics

- Obviously, we do our best to minimize the difference between D_{sim} and D_{det}

- But what if we tried to **do our worst**?

Detector Systematics

- Obviously, we do our best to minimize the difference between D_{sim} and D_{det}

- But what if we tried to **do our worst**?

Given a jet J and a classifier f(J):

$$J = \{(p_{\mathrm{T}}^k, \eta^k, \phi^k) : k = 1 \ldots N_J\}$$

** Our samples are dijets (BG) and y+Z(qq) (signal), simulated With MG+pythia8+Delphes*

Detector Systematics

- Obviously, we do our best to minimize the difference between D_{sim} and D_{det}

- But what if we tried to **do our worst**?

Given a jet J and a classifier f(J):

$$J = \{(p_{\mathrm{T}}^k, \eta^k, \phi^k) : k = 1 \ldots N_J\}$$

** Our samples are dijets (BG) and y+Z(qq) (signal), simulated With MG+pythia8+Delphes*

We want to train an adversary g(J):

$$g(J) \mapsto J'$$

*** g(J) is constrained so that it can only modify pT's by 2%, and eta/phi by 0.02*

Such that the classifier network sees everything as background:

$$f(J') = f(g(J)) = y_{bg}$$

Detector Systematics

- Obviously, we do our best to minimize the difference between D_{sim} and D_{det}

- But what if we tried to **do our worst**?

Given a jet J and a classifier f(J):

$$J = \{(p_{\mathrm{T}}^k, \eta^k, \phi^k) : k = 1 \ldots N_J\}$$

** Our samples are dijets (BG) and y+Z(qq) (signal), simulated With MG+pythia8+Delphes*

We want to train an adversary g(J):

$$g(J) \mapsto J'$$

*** g(J) is constrained so that it can only modify pT's by 2%, and eta/phi by 0.02*

Such that the classifier network sees everything as background:

$$f(J') = f(g(J)) = y_{bg}$$

But! We also require the adversary *g*, to be sneaky…

Adversarial Mismodelling

To train such an adversary, we write two loss functions:

Applied only to signal events →

$$\mathcal{L}_{\text{sig}} = \log(1 - f(g(J)))$$

Applied only to background events →

$$\mathcal{L}_{\text{bg}} = \lambda_{\text{cls}} \left(f(J) - f(g(J))\right)^2 + \sum_i \lambda_{\text{obs}}^{(i)} (\mathcal{O}^{(i)}(J) - \mathcal{O}^{(i)}(g(J)))^2$$

Adversarial Mismodelling

To train such an adversary, we write two loss functions:

$$\mathcal{L}_{\text{sig}} = \log(1 - f(g(J)))$$

Binary crossentropy perturbed signal jets to be labeled as background

$$\mathcal{L}_{\text{bg}} = \lambda_{\text{cls}} \left(f(J) - f(g(J))\right)^2 + \sum_i \lambda_{\text{obs}}^{(i)} (\mathcal{O}^{(i)}(J) - \mathcal{O}^{(i)}(g(J)))^2$$

Adversarial Mismodelling

To train such an adversary, we write two loss functions:

$$\mathcal{L}_{\text{sig}} = \log(1 - f(g(J)))$$

$$\mathcal{L}_{\text{bg}} = \lambda_{\text{cls}} \left(f(J) - f(g(J))\right)^2 + \sum_i \lambda_{\text{obs}}^{(i)} (\mathcal{O}^{(i)}(J) - \mathcal{O}^{(i)}(g(J)))^2$$

Mean squared error to minimize change in the classifier's response to background jets

Adversarial Mismodelling

To train such an adversary, we write two loss functions:

$$\mathcal{L}_{\text{sig}} = \log(1 - f(g(J)))$$

$$\mathcal{L}_{\text{bg}} = \lambda_{\text{cls}} \left(f(J) - f(g(J))\right)^2 + \sum_i \lambda_{\text{obs}}^{(i)} (\mathcal{O}^{(i)}(J) - \mathcal{O}^{(i)}(g(J)))^2$$

Mean squared error(s) to minimize change in any other observables that we wish to remain unchanged

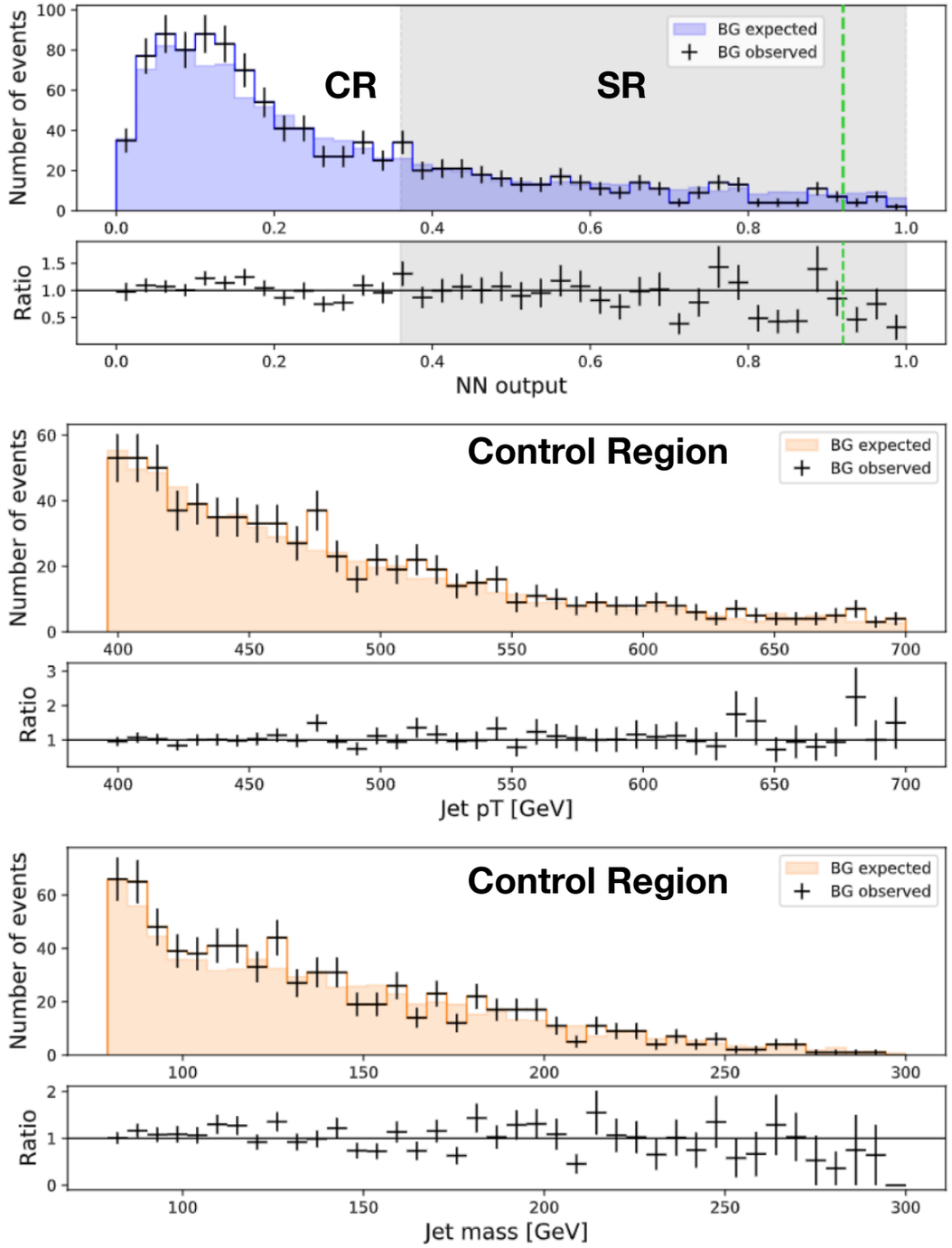
Implementing the Attack

- For auxiliary observables, we selected the jet p_T and mass

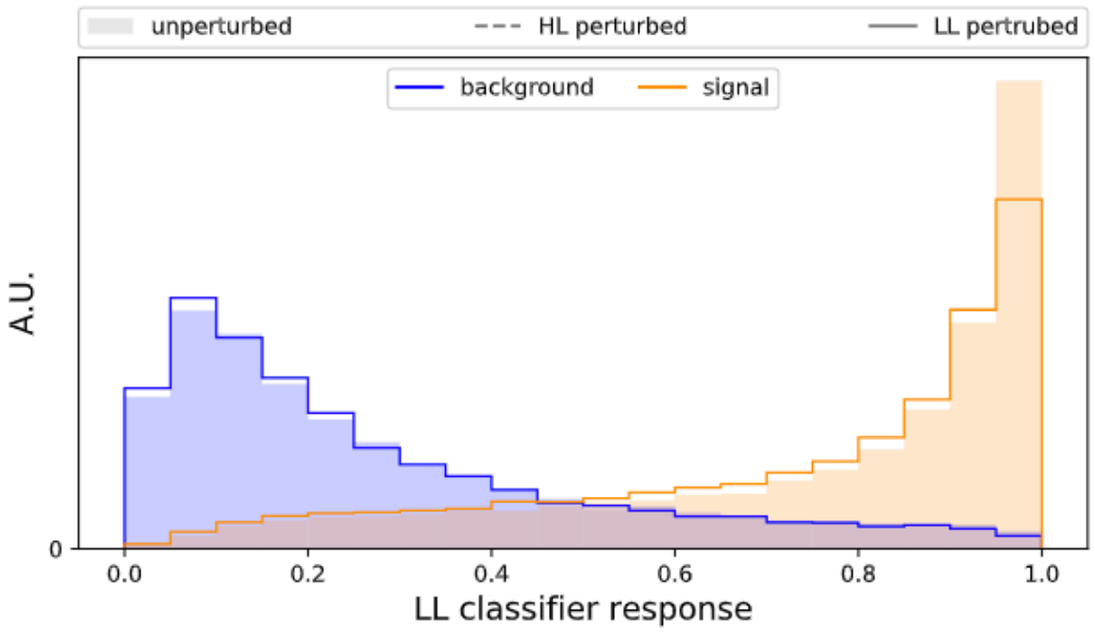
- Tune loss terms by maximizing the value of L_{sig}

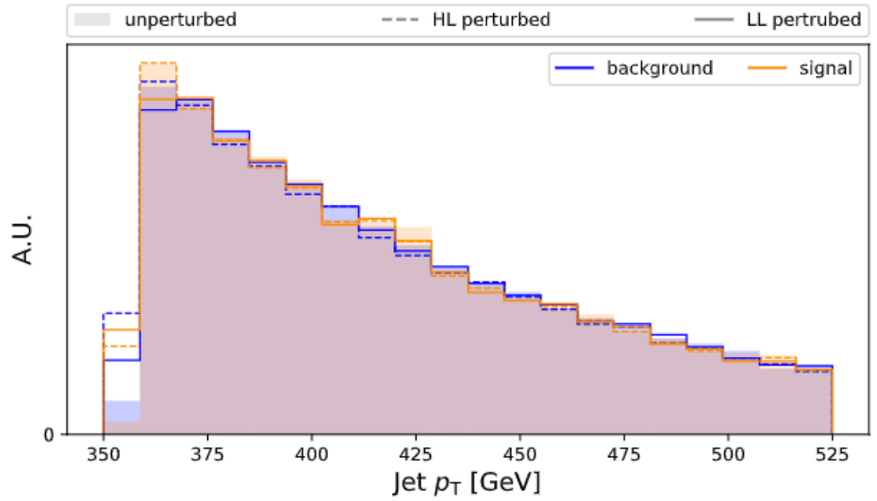
  - Stop training when goodness-of-fit of the auxiliary observables becomes unacceptable.

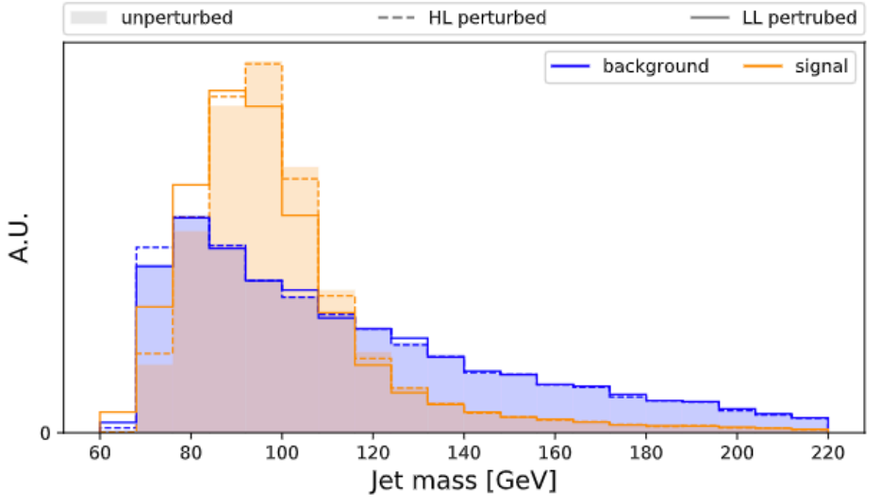
- This gives worst-case mismodelling while remaining “unnoticeable”

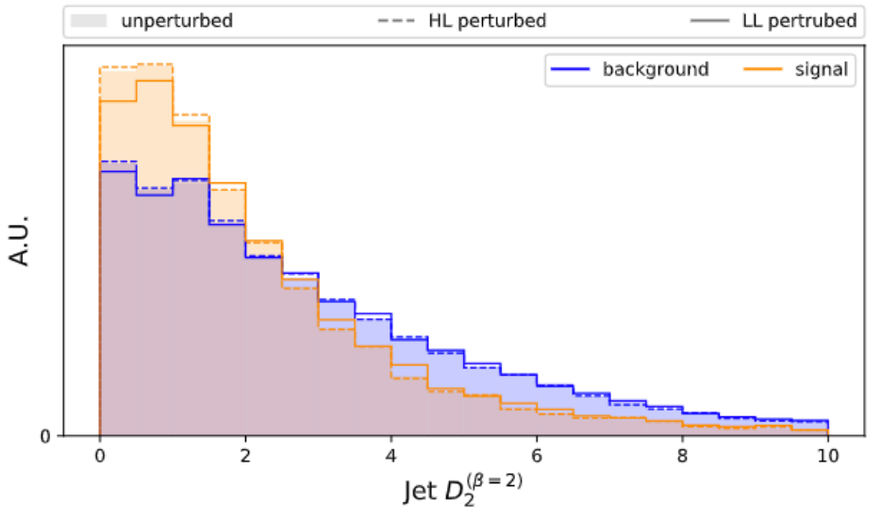


Perturbed Distributions

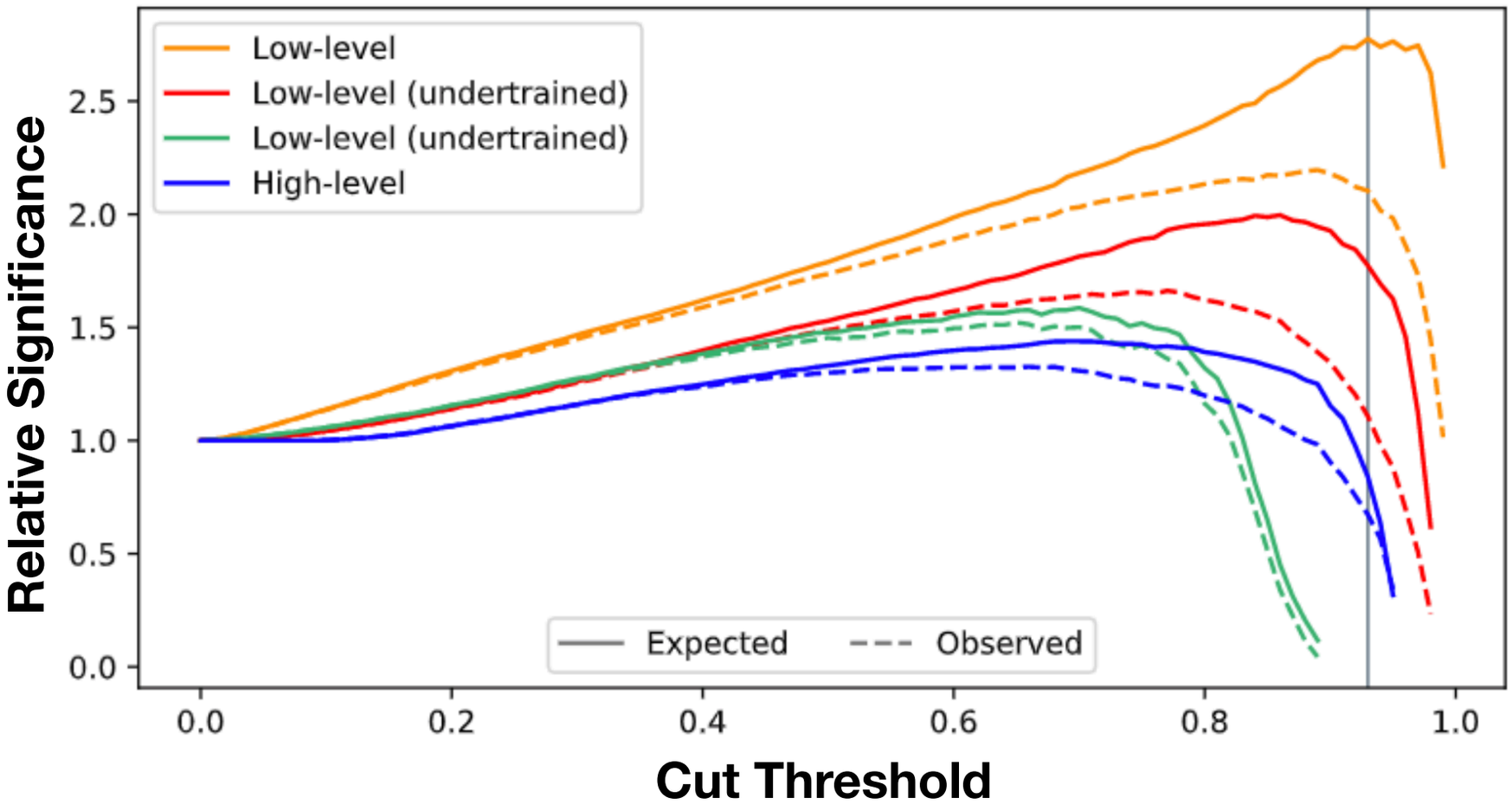








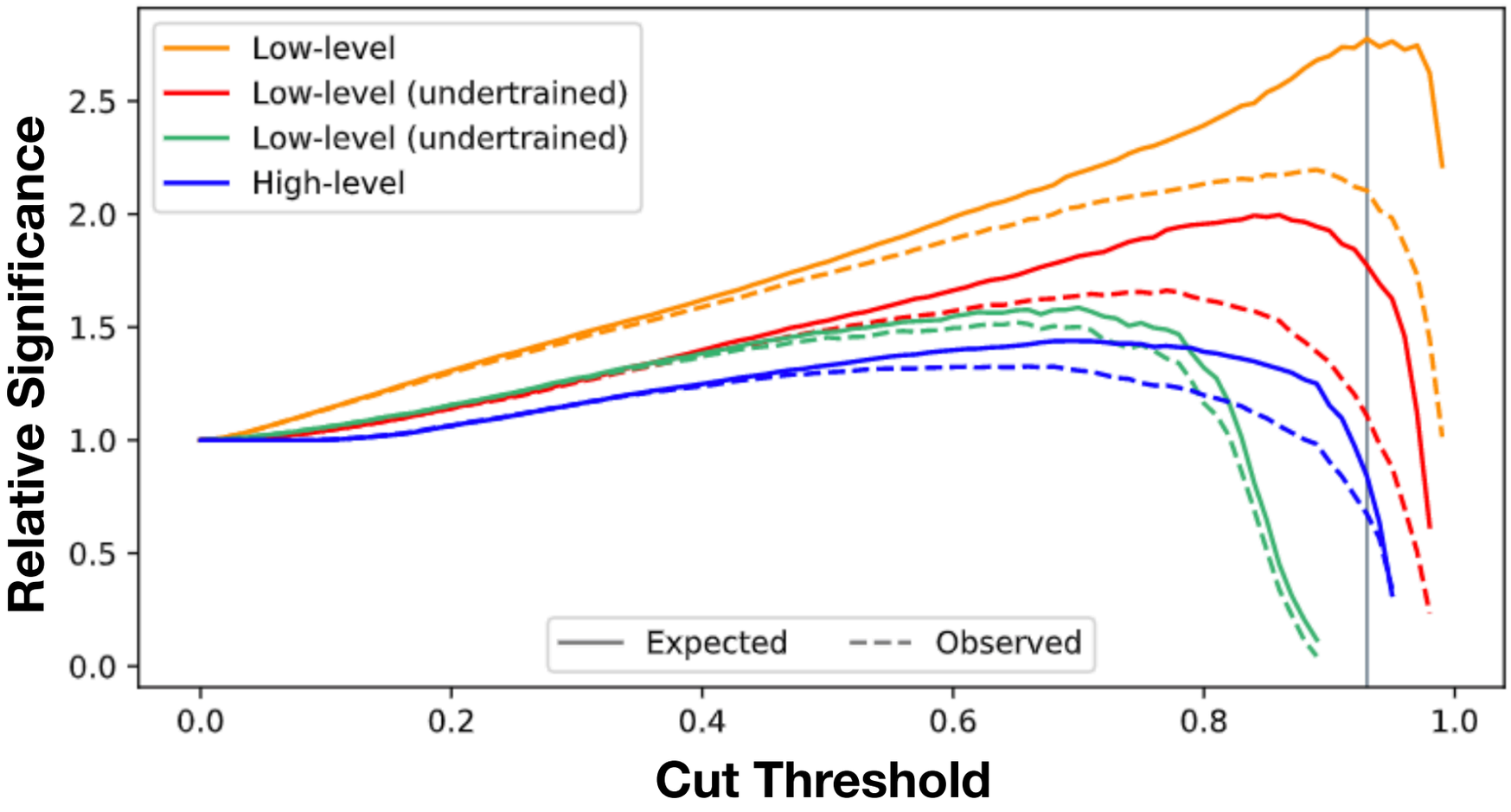
Results



We compared a simple cut&count sensitivity for a boosted resonance with:

- ParticleFlow Network (operating on **low-level** jet constituent 4-vectors)
- HL Network (operating on **high-level** jet p_T/mass/η/D_2)

Results



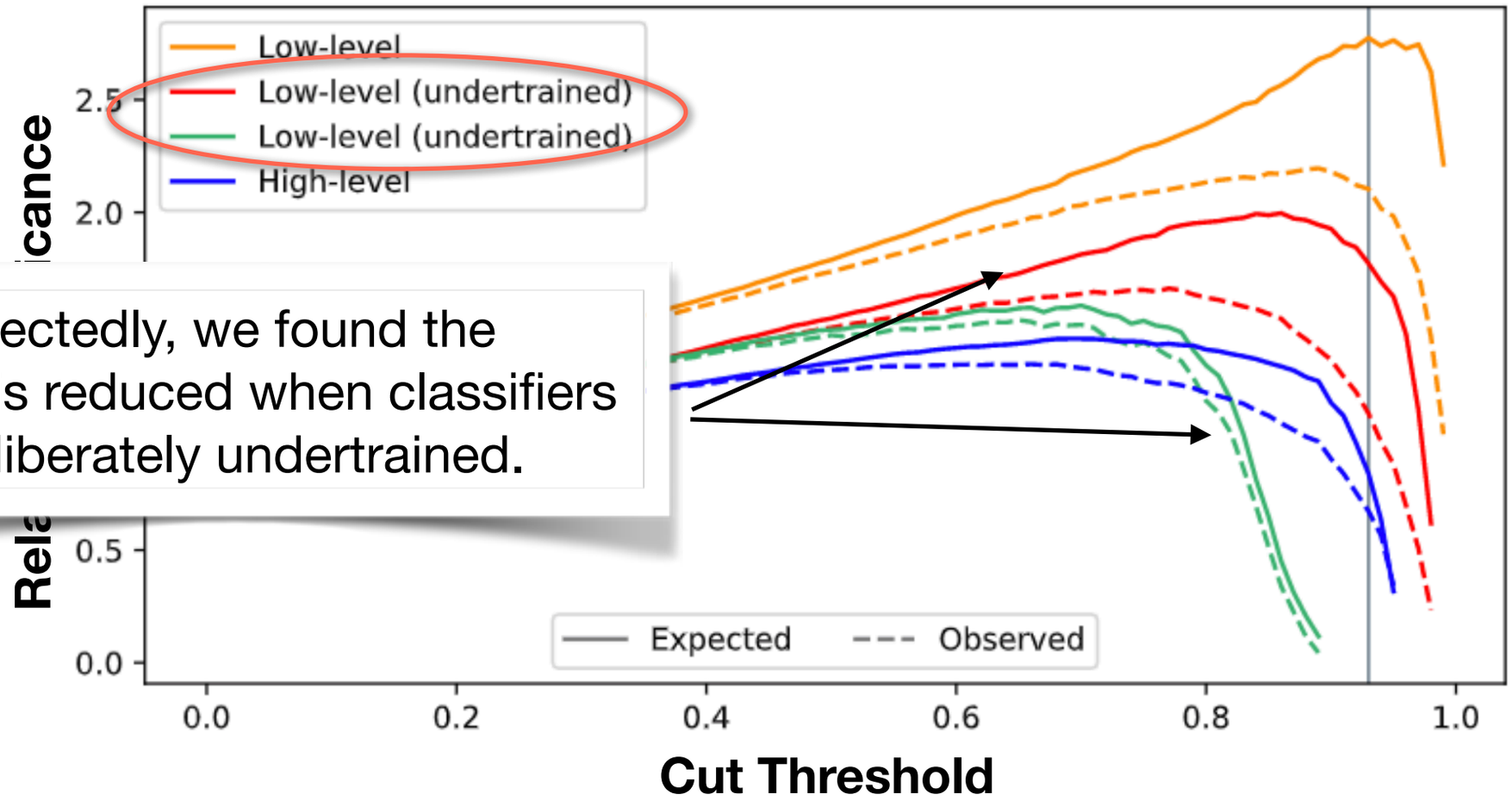
~25% effect for PFN

~8% effect for HL net

We compared a simple cut&count sensitivity for a boosted resonance with:

- ParticleFlow Network (operating on **low-level** jet constituent 4-vectors)
- HL Network (operating on **high-level** jet p_T/mass/η/D_2)

Results



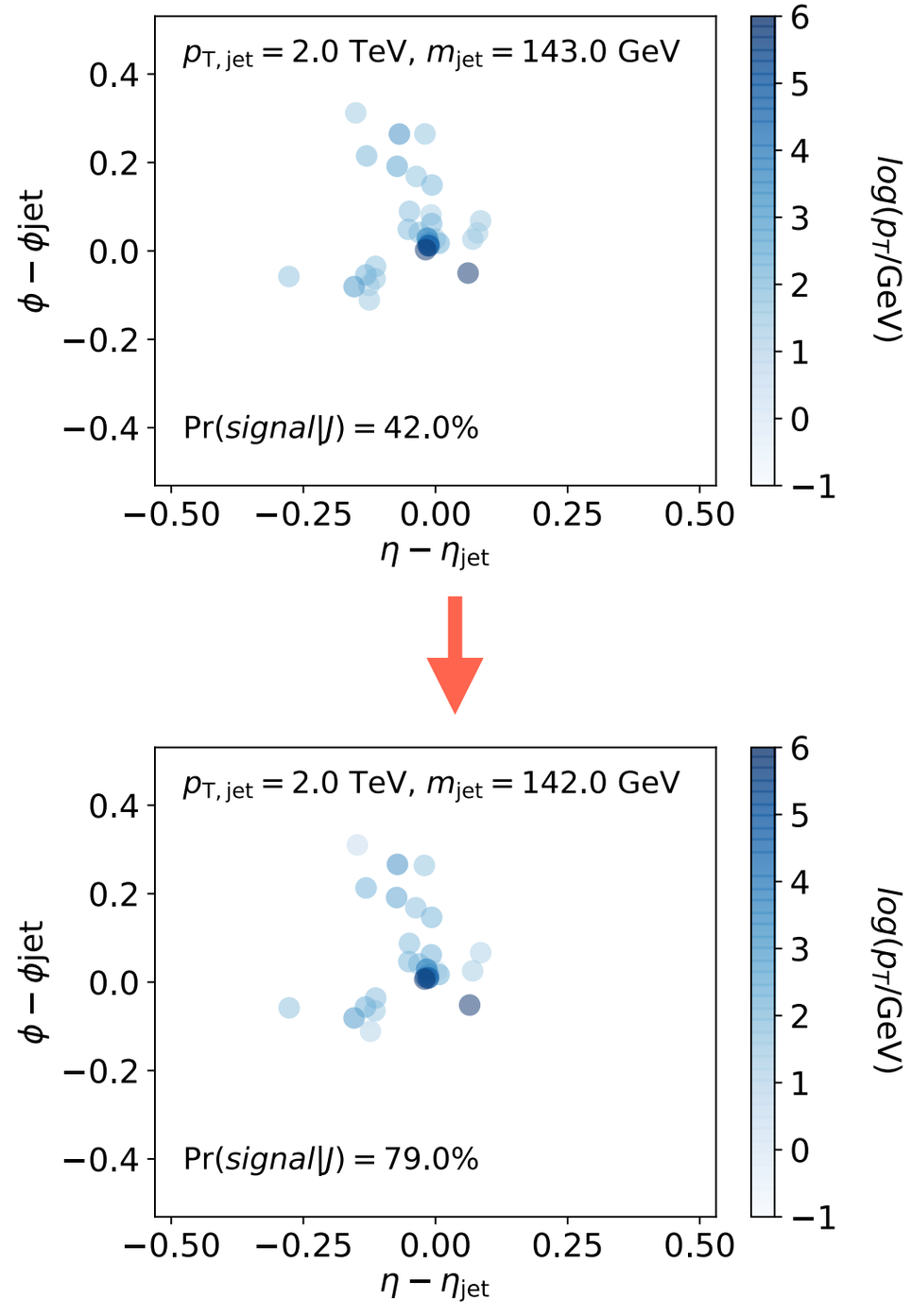
Unexpectedly, we found the effect is reduced when classifiers are deliberately undertrained.

We compared a simple cut&count sensitivity for a boosted resonance with:

- ParticleFlow Network (operating on **low-level** jet constituent 4-vectors)
- HL Network (operating on **high-level** jet p_T/mass/η/D_2)

Conclusions

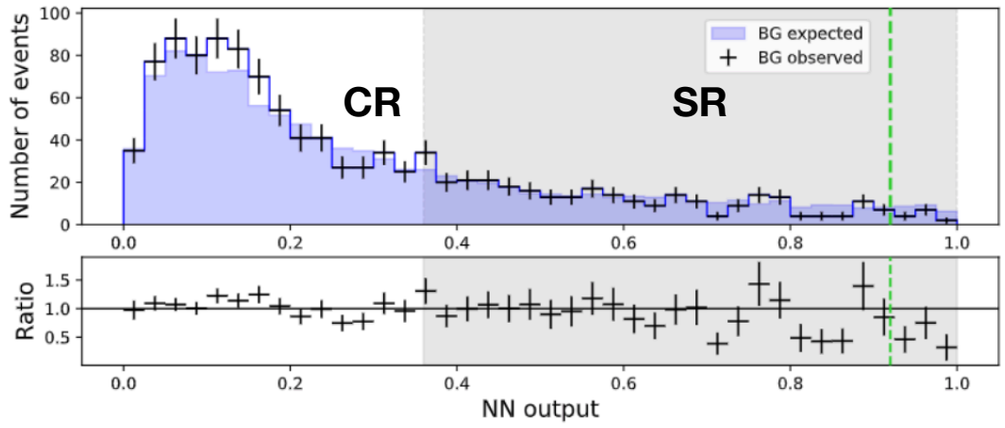
We show that is possible to find systematic mismodellings $g(J) \mapsto J'$, that confuse NN classifiers

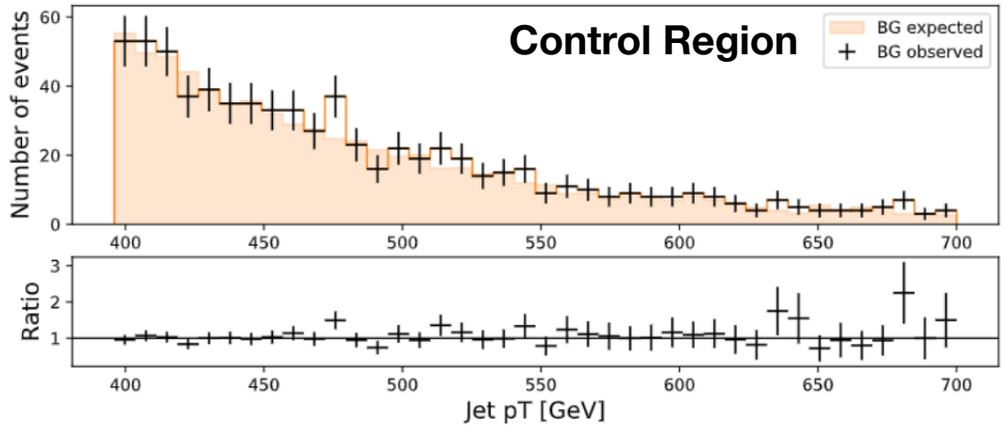


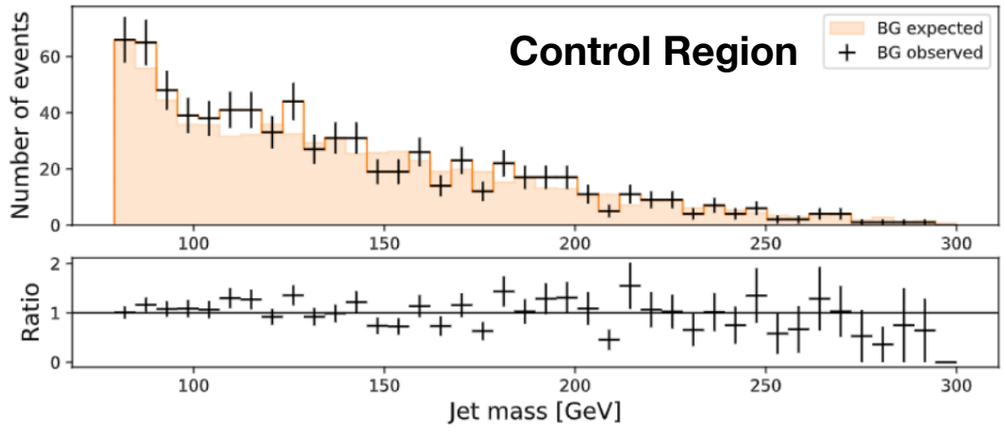
Conclusions

We show that is possible to find systematic mismodellings $g(J) \mapsto J'$, that confuse NN classifiers

- These effects are subtle, remaining undetected in control/validation regions





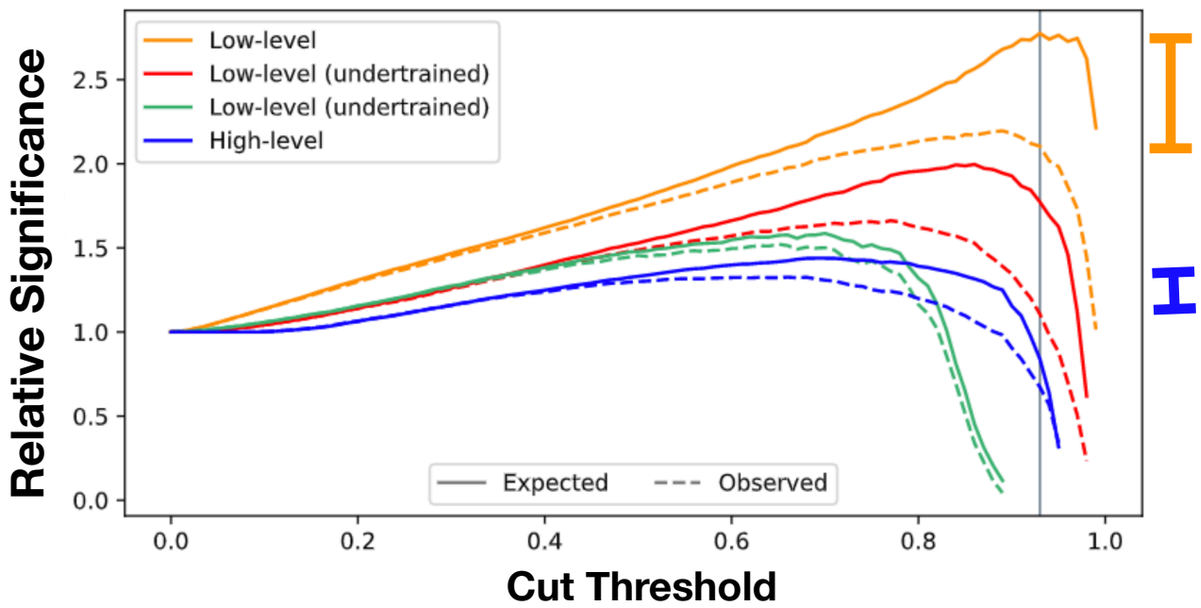


Conclusions

We show that is possible to find systematic mismodellings $g(J) \mapsto J'$, that confuse NN classifiers

- These effects are subtle, remaining undetected in control/validation regions

- Susceptibility is reduced, but not entirely, when using fewer and higher-level inputs



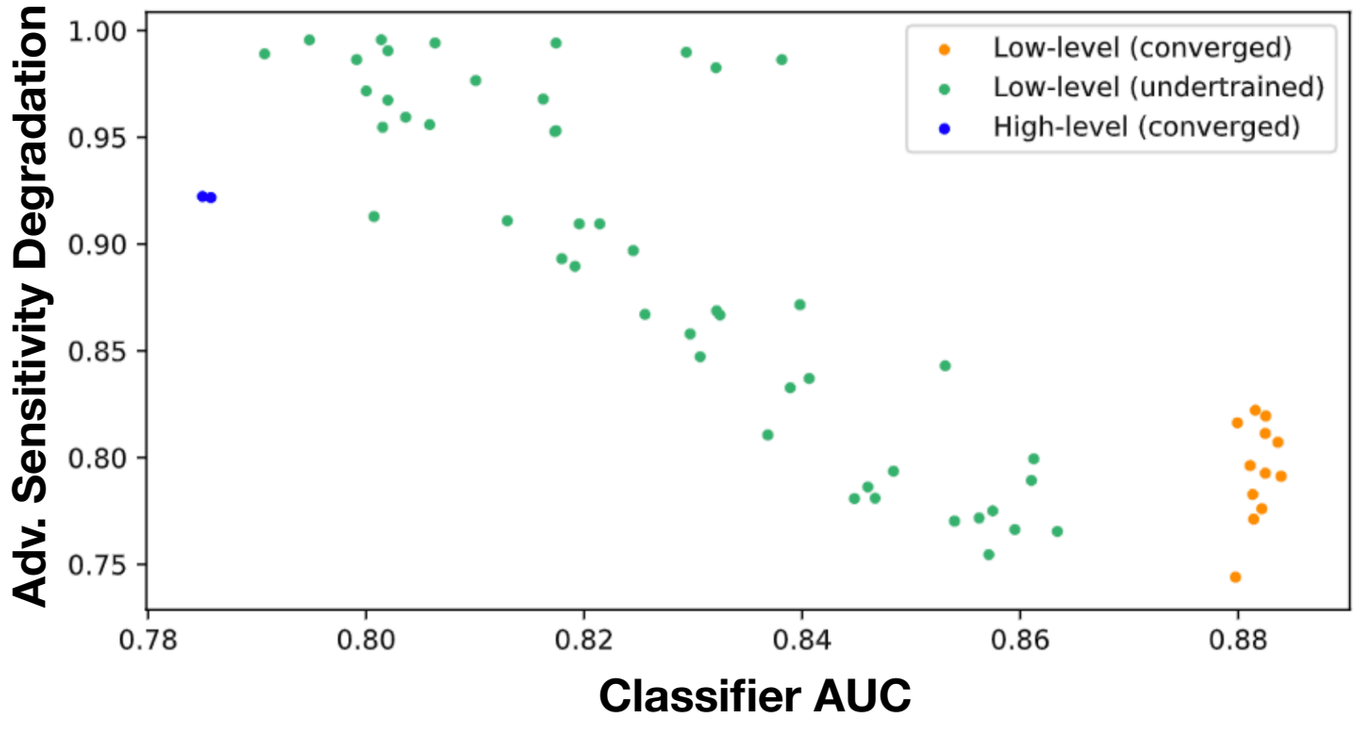
Conclusions

We show that is possible to find systematic mismodellings $g(J) \mapsto J'$, that confuse NN classifiers

- These effects are subtle, remaining undetected in control/validation regions

- Susceptibility is reduced, but not entirely, when using fewer and higher-level inputs

- We also find evidence that undertraining nets may help reduce susceptibility



Future Work

- **BY NO MEANS** does this study give any sense of a “systematic” that can be assessed for techniques used today

    - It is more like an upper bound for systematic exposure due to mismodelling

- However, this method could be used to **identify or design** ML architectures which are inherently robust, reducing the need to worry

Future Work

- **BY NO MEANS** does this study give any sense of a “systematic” that can be assessed for techniques used today

  - It is more like an upper bound for systematic exposure due to mismodelling

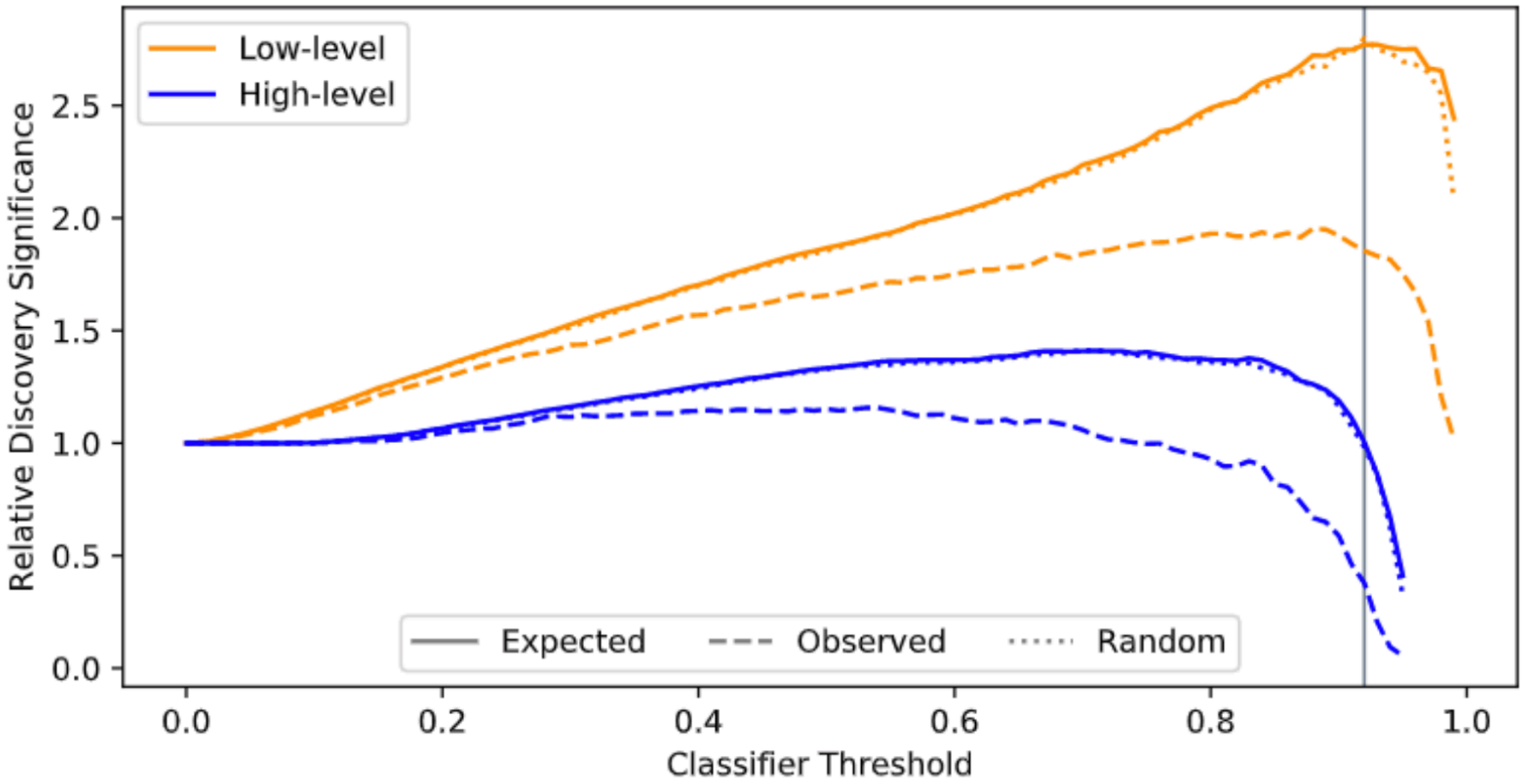
- However, this method could be used to **identify or design** ML architectures which are inherently robust, reducing the need to worry

- The adversarial network g(J) probes the “space of mismodellings”, but tells us nothing about **statistics**: i.e. how *likely* is a given mismodelling

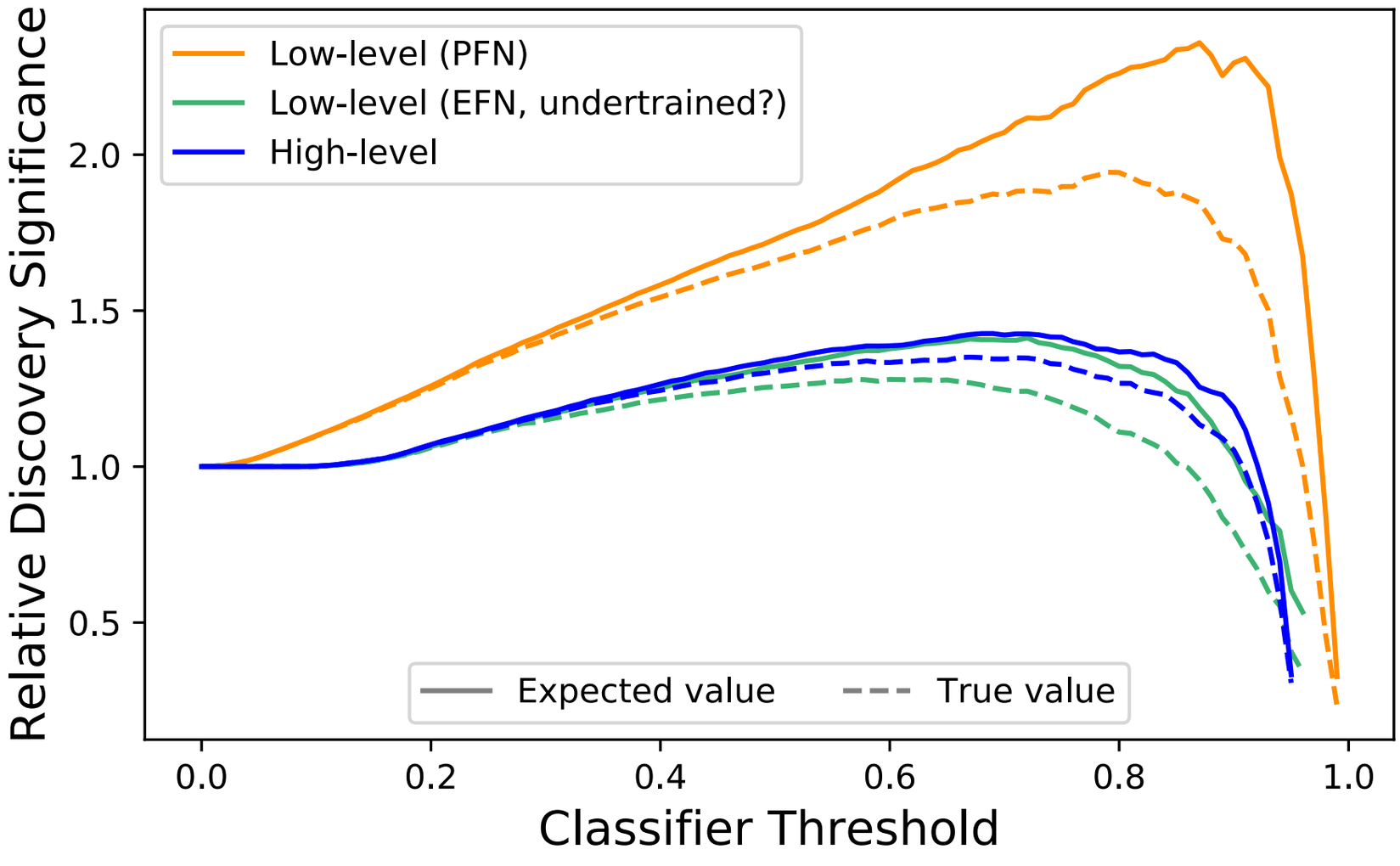
  - However, the mathematical apparatus gives us access to explore this large function space e.g. using optimization.

  - We hope this could be a starting point to a more robust understanding of high-dimensional, nonlinear systematic effects in HEP data!

Backup


Low-level
High-level
Expected
Observed
Random
Relative Discovery Significance
Classifier Threshold
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0

Backup


Low-level (PFN)
Low-level (EFN, undertrained?)
High-level
Expected value
True value
Relative Discovery Significance
Classifier Threshold
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0