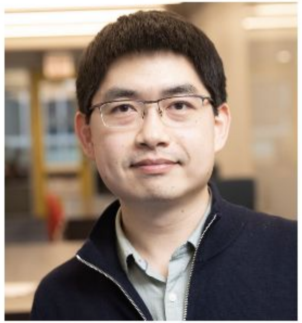
# RL-Driven Anomaly Detection for Adaptive Trigger Menus at the LHC

Zixin Ding
University of Chicago
on behalf of the team

Anomaly Detection
Workshop 2025
June 17
Columbia University

Authors involved in the Autonomous Trigger Project:

Yuxin Chen, Zixin Ding, Shaghayegh Emami, Abhijith Gandrakota, Christian Herwig, David Miller, Jennifer Ngadiuba, Giovanna Salvi, Cecilia Tosciri, Nhan Tran









# Single Trigger Setting

Using RL to adaptively adjust thresholds in both training (learning phase) and testing data (testing phase).
Stabilizing Background Rates

# Why Reinforcement Learning for Adaptive Thresholding

🚨 The Challenge:

- Trigger Menu is fixed menu and does not account for changing accelerator and detector conditions over time.
- Level-1 triggers use fixed thresholds (e.g. on $H_T$, jets $p_T$), rejecting >99% of events.
- These thresholds are manually tuned and static, even though:
  - Data rates fluctuate.
  - Physics signatures evolve.
  - Background noise varies across time and detector conditions.

⚠️ The Consequence:

- Risk of discarding rare/anomalous events, including sign of new physics.

# Why Reinforcement Learning?

🤖 RL adapts thresholds on-the-fly, optimizing for:

- Event retention with bandwidth limits
- Maintaining a desired background acceptance rate.
- Maximizing signal-like or anomalous events.

- Learns a **policy** to dynamically shift thresholds based on:
  - Observed acceptance rate
  - Prior threshold values
  - Event-level features (optional)

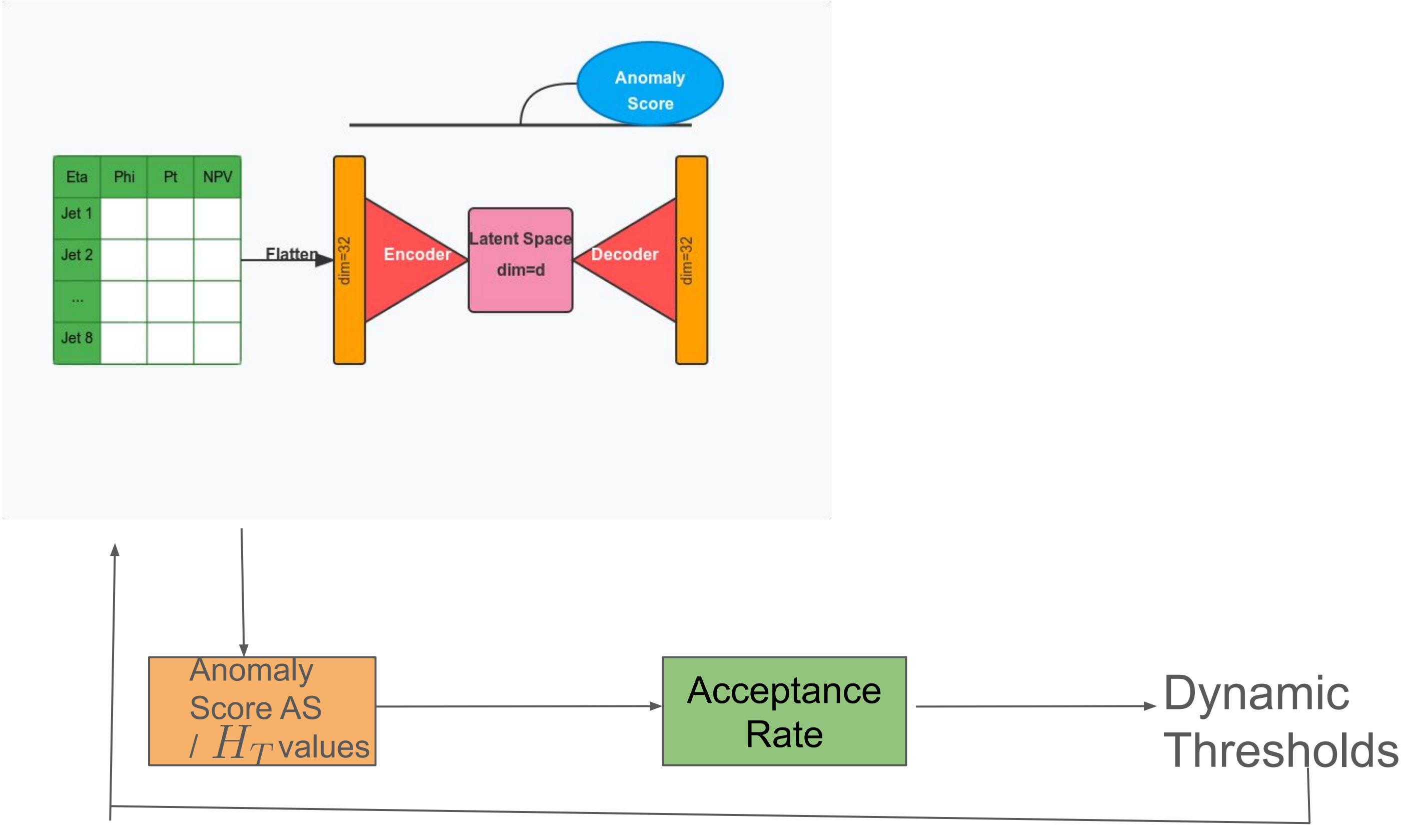
# Problem Setup

Goal: Ensuring a stable background rate across varying pileup conditions by dynamically adjusting trigger menu values. We use background rate as 5% as a proof-of-concept.

Solution:
Separate Training set into batches of events
State: Previous Batch Acceptance Rates, Previous Batch Event Level Features
Action: continuous value of adjusting thresholds
Value: Q(s,a): Estimates the long-term benefit of applying action a in state s, i.e., how this adjustment improves future background control and signal yield.

Deep-Q Learning learns a function approximator(usually a neural network) to estimate.

# RL for Anomaly Detection Pipeline



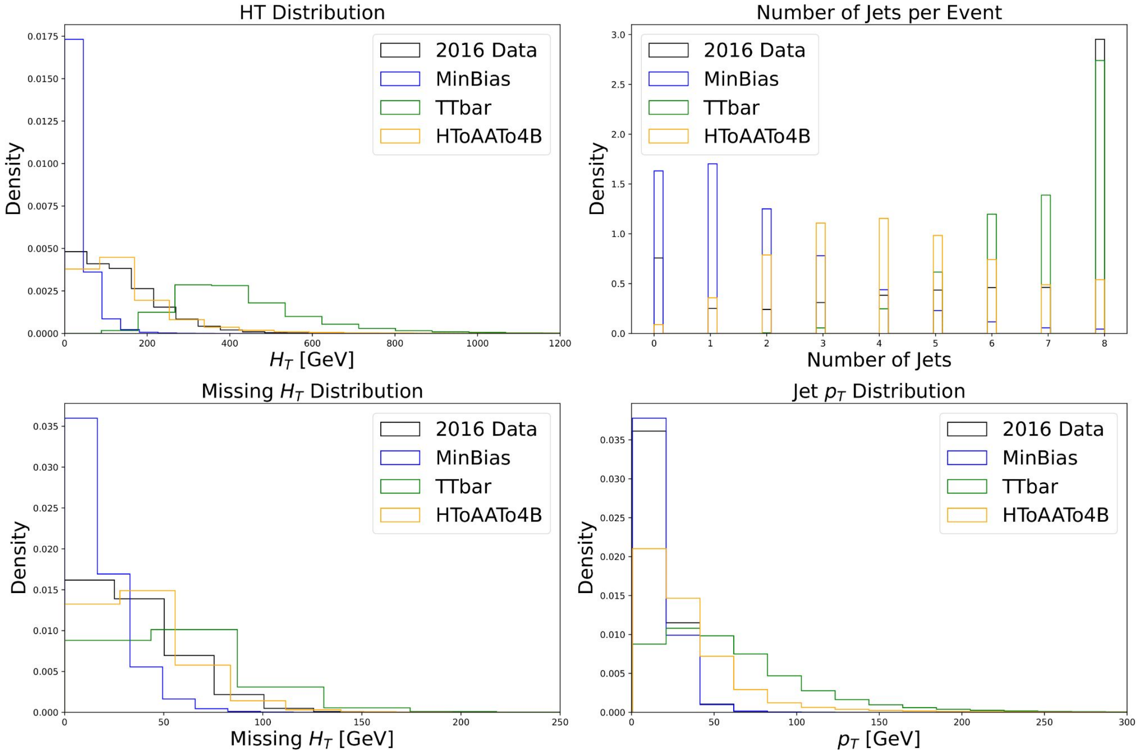
# Dataset and Trigger Setup

- CMS Open Data from Run2 (2016), including:
  - ZeroBias (data)
  - MiniBias (MC background)

- Two representative trigger paths:
  - Conventional $H_T$ trigger, with $H_T = \sum p_T^{jet}$: captures broad hadronic activity, sensitive to pileup conditions
  - Anomaly Detection trigger: targets rare or unexpected signatures, trained using MinimumBias training dataset.

- To emulate this behavior in MC, samples are sorted by NPV,
  used as a proxy of pile-up

# Exploratory Data Analysis



# Proportional-Derivative controller(PD controller)

- Proportional term (P): Reacts to the current error (difference between actual value and desired target). If error is large, it changes the output significantly.

$$P = K_p \cdot \mathrm{Error}$$

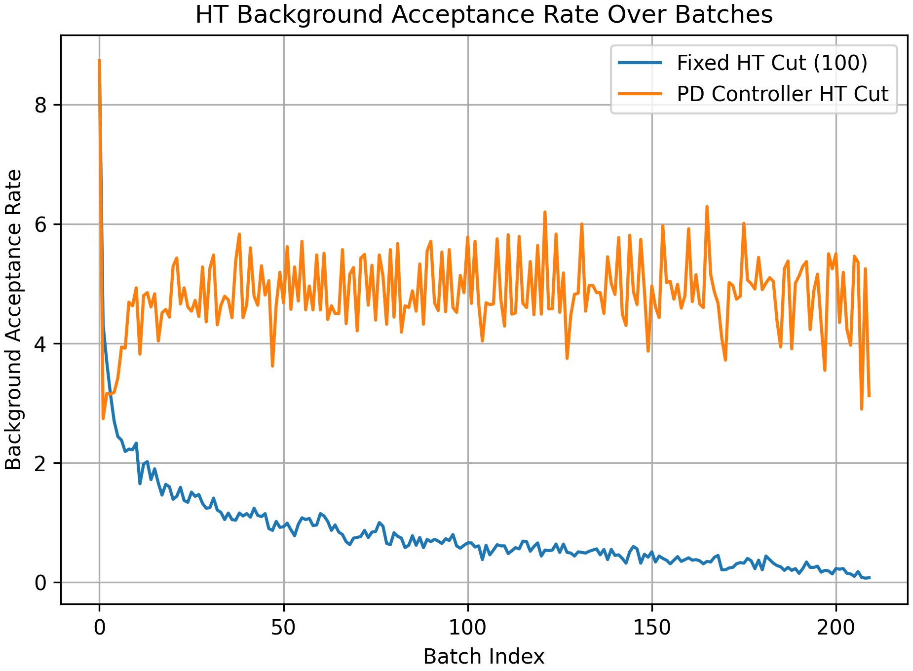
- Derivative term (D): Reacts to the rate of change of the error. Helps to anticipate the system's future behavior and dampen oscillations.

$$D = K_d \cdot \frac{d(\mathrm{Error})}{dt}$$

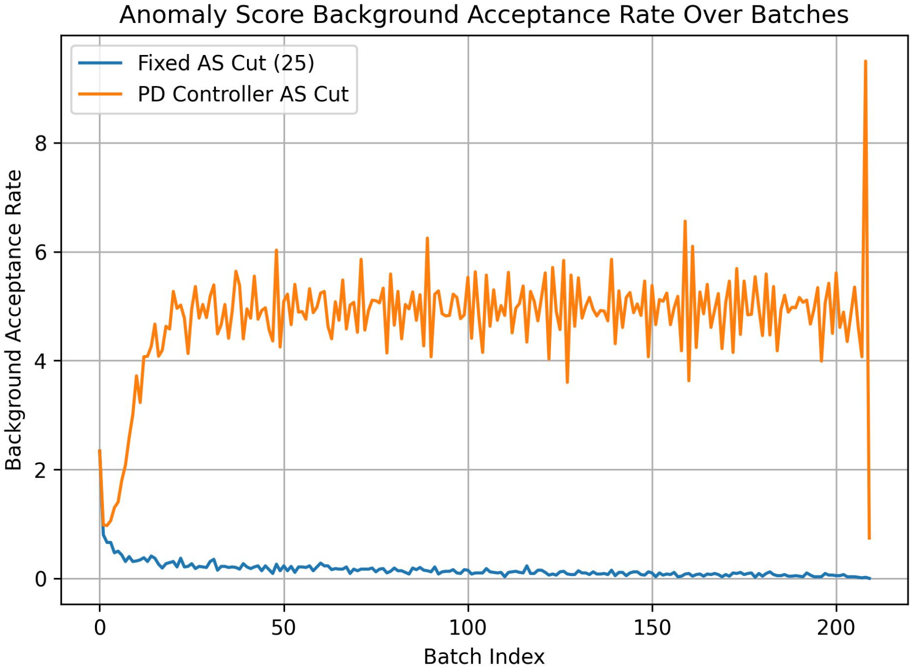
Drawbacks 🚨:

- $K_p, K_d$ are **hard to tune**. $K_p$ denotes Proportional-gain, how aggressively it reacts to the errors. $K_d$ denotes Derivative-gain, how much it dampens the based on the error trend. We show results that these two parameters are selected by **grid search over background samples MiniBias**.

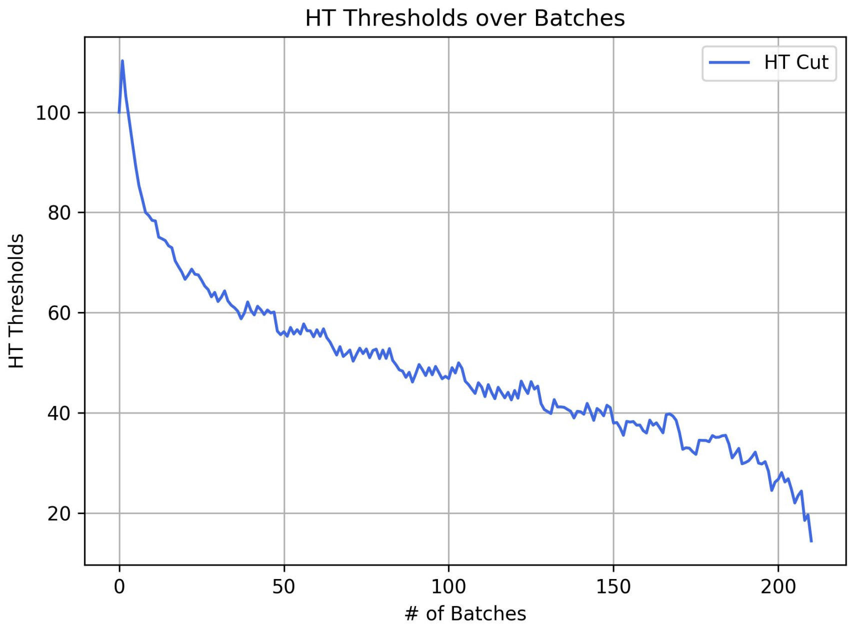
# Fixed Trigger Menu + PD Controller



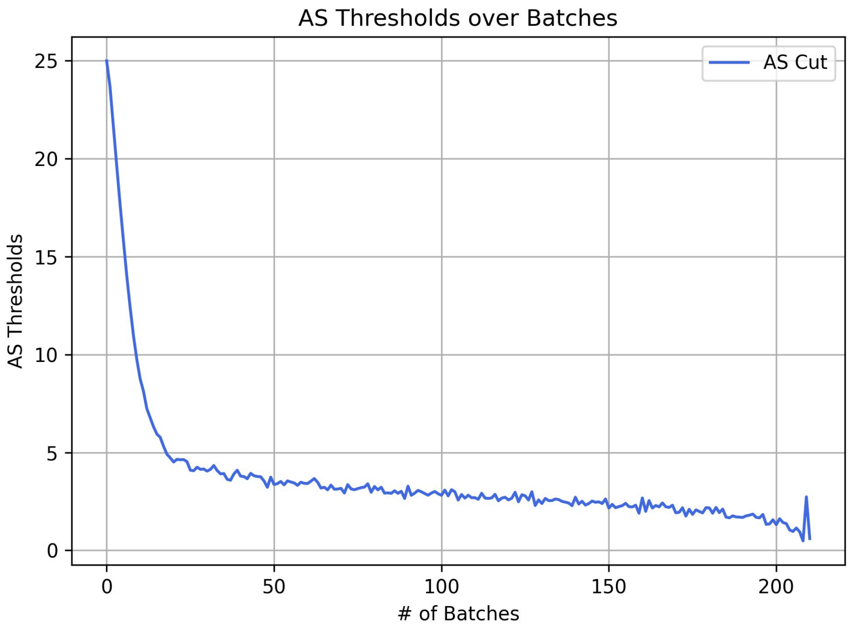

$K_p = 2.55 \; K_d = 0.2$




$K_p = 0.5 \; K_d = 0$



HT Thresholds over Batches
HT Cut
HT Thresholds
# of Batches
100
80
60
40
20
0
50
100
150
200

AS Thresholds over Batches
AS Cut
AS Thresholds
# of Batches
25
20
15
10
5
0
0
50
100
150
200

# Q-Learning

- **Deep Q-learning** is a model-free **reinforcement learning algorithm**.

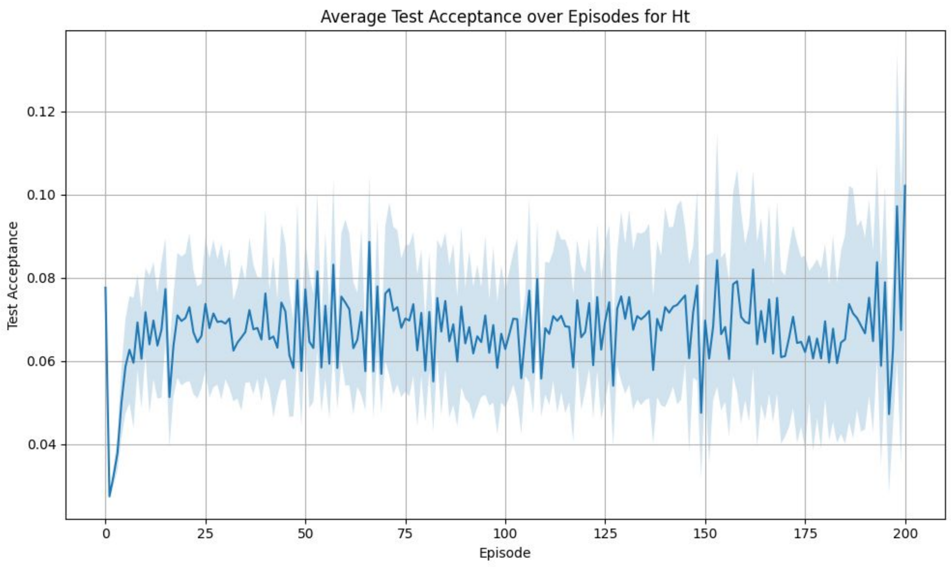
- It learns the optimal action-value function (Q-function) to maximize expected cumulative reward.

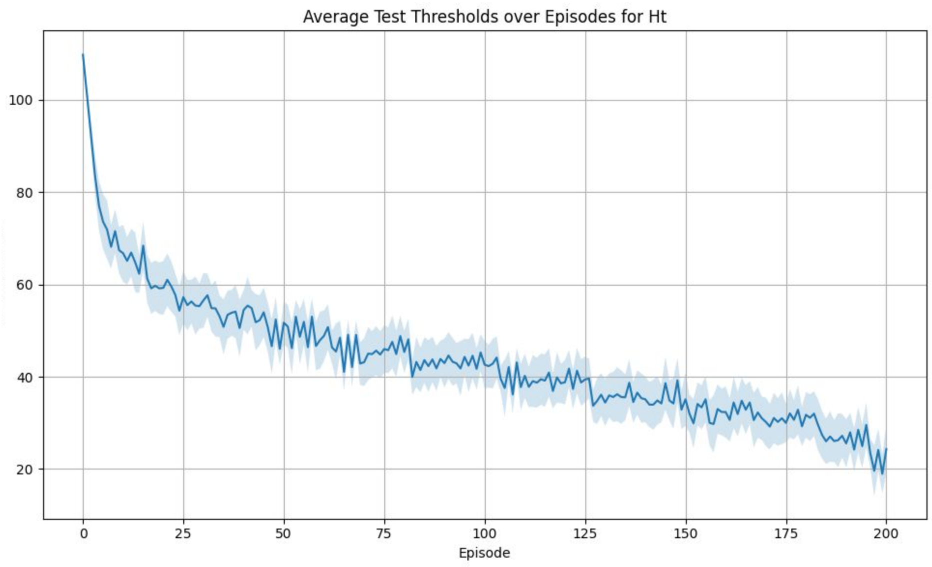
- Deep Q-learning automatically learns the values of functions in different states compared to manually tuning of $K_p$ and $K_d$.

- Deep Q-learning learns optimal behavior across varied distributions over time, and PD controller is poor if background distribution changes.

# Deep Q-Learning for Ht

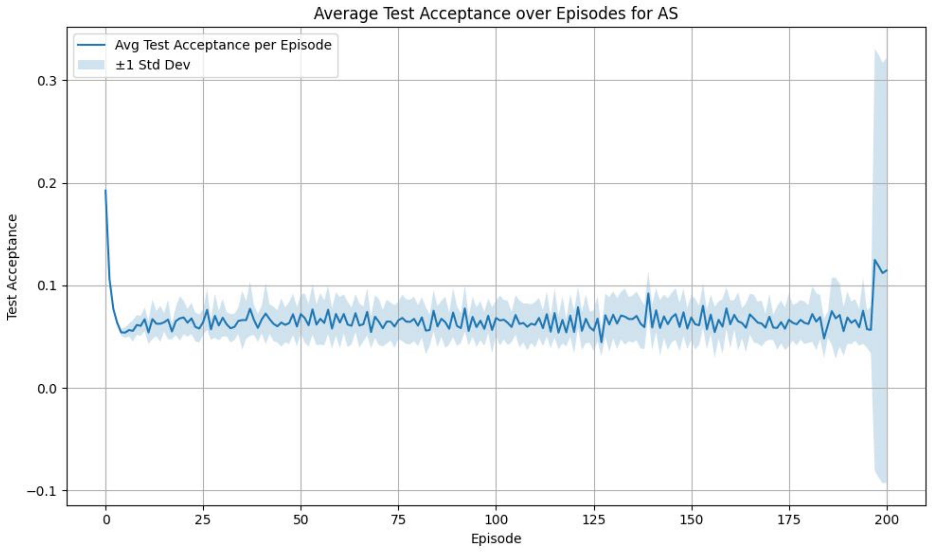
Test Phase: Repeated for 20 trials

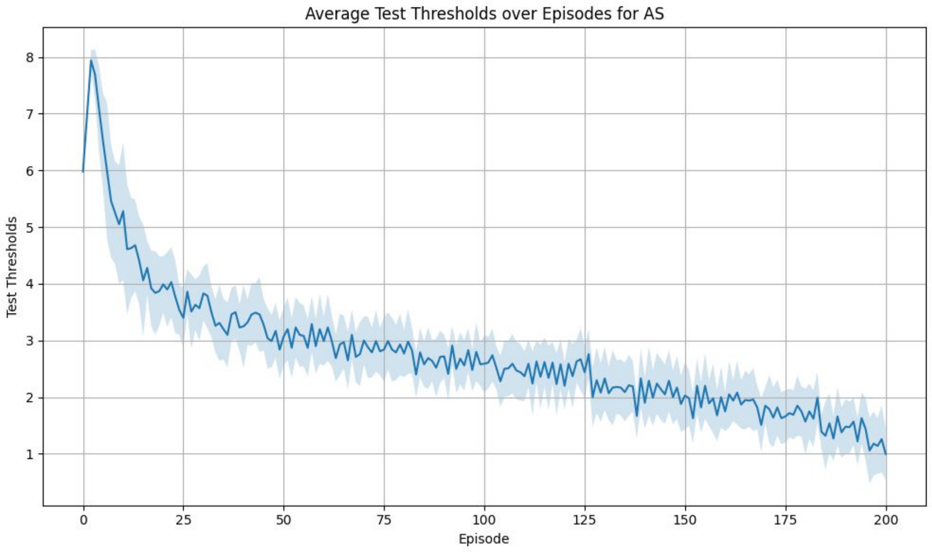


# Deep Q-Learning for AS

Test Phase: Repeated for 20 trials



Thank you!