Strategy Classification and Mitigation in Adversarial Gameplay Modeling

Lockheed Martin Advanced Technology Labs

Jessica Kinnevan & Toral Chauhan
Team G_HF_1
4/24/2026

IOWA STATE UNIVERSITY

The Problem

Most game AI adversary models are trained with the objective being to win.

How can a model be trained to mitigate specific player strategies?

Who it affects: Defense, Financial, Gaming, Education

Granular Problems

1. How is “strategy” defined in the context of a particular game?
2. What differentiates one strategy from another?
3. What data do we use to train a model?
4. How can a model be trained to mitigate multiple strategies in a way that can be measured?

Models 1, 2, and 3 - Key Findings (Jessica Kinnevan)

1. Model 2 - A Vector Quantized Variational Autoencoder (VQ-VAE) codebook clusters game data in a way that is indirectly influenced by the VQ-VAE decoder's predicted output.

2. Model 3 - A multi-head Proximal Policy Optimization (PPO) adversarial model generally performed better when training and inference was routed to heads using the VQ-VAE as the strategy classifier, selecting the head by strategy index (S0, S1, S2, S3, S4)

3. Game and Model 1 - changing the game environment has cascading effect Game environment must enable varieties of strategies, and Model 1 must exploit them for VQ-VAE (Model 2) to differentiate them.

Key Findings (Toral Chauhan)

F1 **Bimodal Performance Gap**
GridPower (79%) and Bayesian (57%) sit 33+ pts above all others.

F2 **Exploitation ≠ Adaptation**
GridPower wins via a game flaw (targets buildings). High wins, zero SSR.

F3 **Model 2 Boosts Every Adversary 3.2×**
Avg wins jump from 4.7 → 15.0 when Model 2 is integrated. Strategy-awareness is the single biggest performance lever.

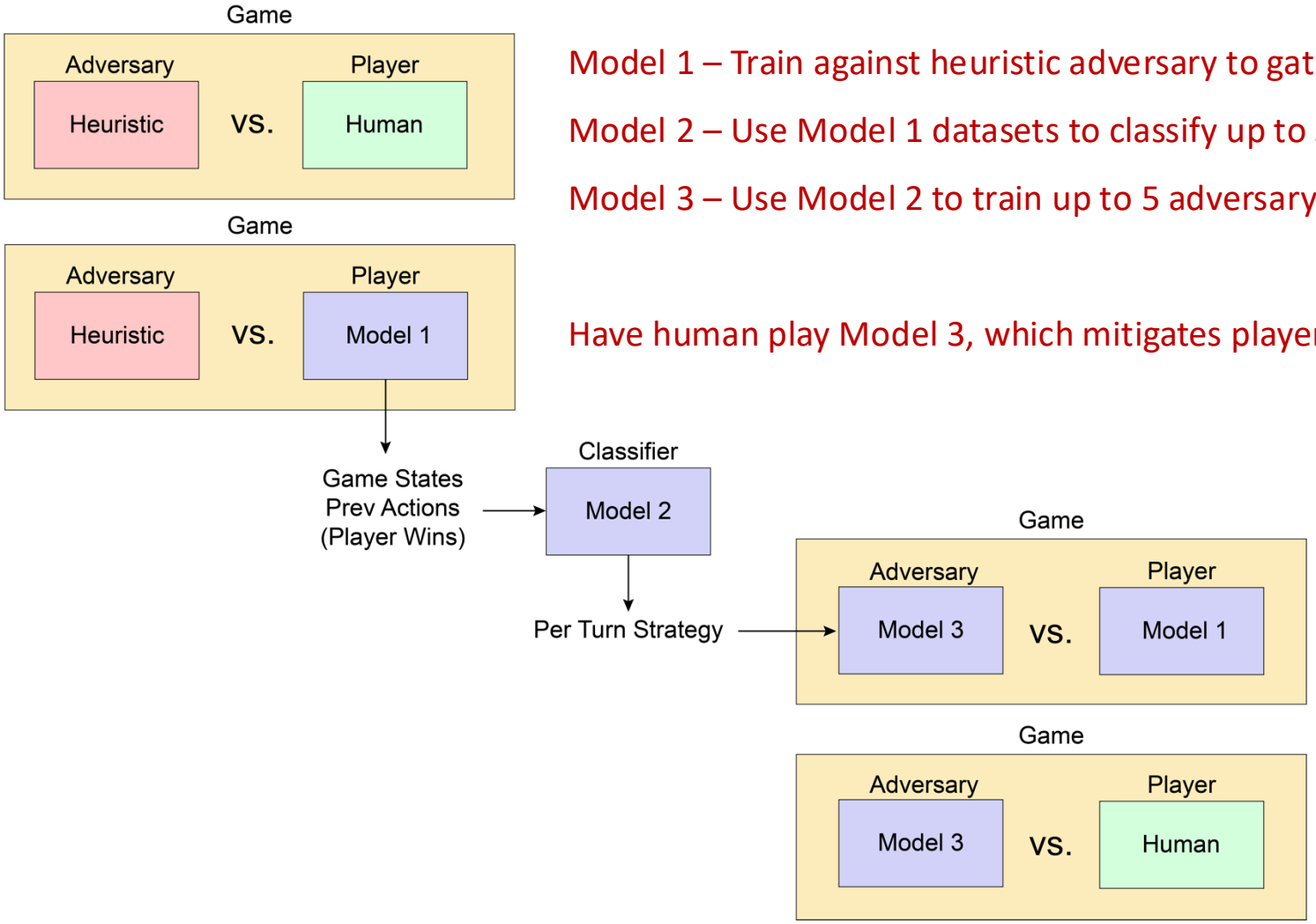
F4 **SSR = 0 Without Model 2**
All baseline adversaries (PPO, DQN, MCTS, Minimax, A*) show SSR = 0.0. Strategy-blind agents never force the player to adapt.

F5 **Planning Fails in 5-Turn Environments**
MCTS, Minimax, and A* perform at or below Random (≤2.8 avg wins). Short horizon eliminates look-ahead advantage entirely.

F6 **Building Damage = Key Differentiator**
Only GridPower and Bayesian reduce avg power left below 1.0. All others leave the player with comfortable grid margins.

F7 **VBRS Architecture Ceiling Not Reached**
VBRS results reflect early low-β stage. Designed to improve continuously current results are a floor, not a ceiling.

3 Model Approach



Model 1 – Train against heuristic adversary to gather datasets

Model 2 – Use Model 1 datasets to classify up to 5 player strategies

Model 3 – Use Model 2 to train up to 5 adversary policies

Have human play Model 3, which mitigates player strategies

3 Model Approach

Model 1 - Player Model

- Plays the game well enough to use a variety of strategies
- Saves winning game sequences as datasets for Model 2

Model 2 - Classifier Model

- Classifies each turn (set of up to 6 actions) from Model 1 datasets as **one of N strategies**
- Informs Model 3 in real time with strategies detected as player makes actions

Model 3 - Adversary Model

- Trains the adversary to mitigate strategies from Model 2
- Trains N expert heads, one for each strategy

Team HiFi Contributions - Evolved Due to Time Constraints

Game – Game environment and UI by undergraduate team (Team LoFi)

Team HiFi additions:

- Adversary heuristic – 75%/25% preference to attack building/mech
- Grid Power trade-off mechanic
- Requirement for one of each mech at start of game
- Ability to destroy mountains

Model 1 – Developed PPO with 5 unique player policies (different from UG model)

Model 2 – Developed VQ-VAE with 3 trained versions (2.1, 2.2, 2.3)

Model 3 – Developed/tested multiple models with vs without Model 2.2

The Game

Player vs Heuristic Adversary

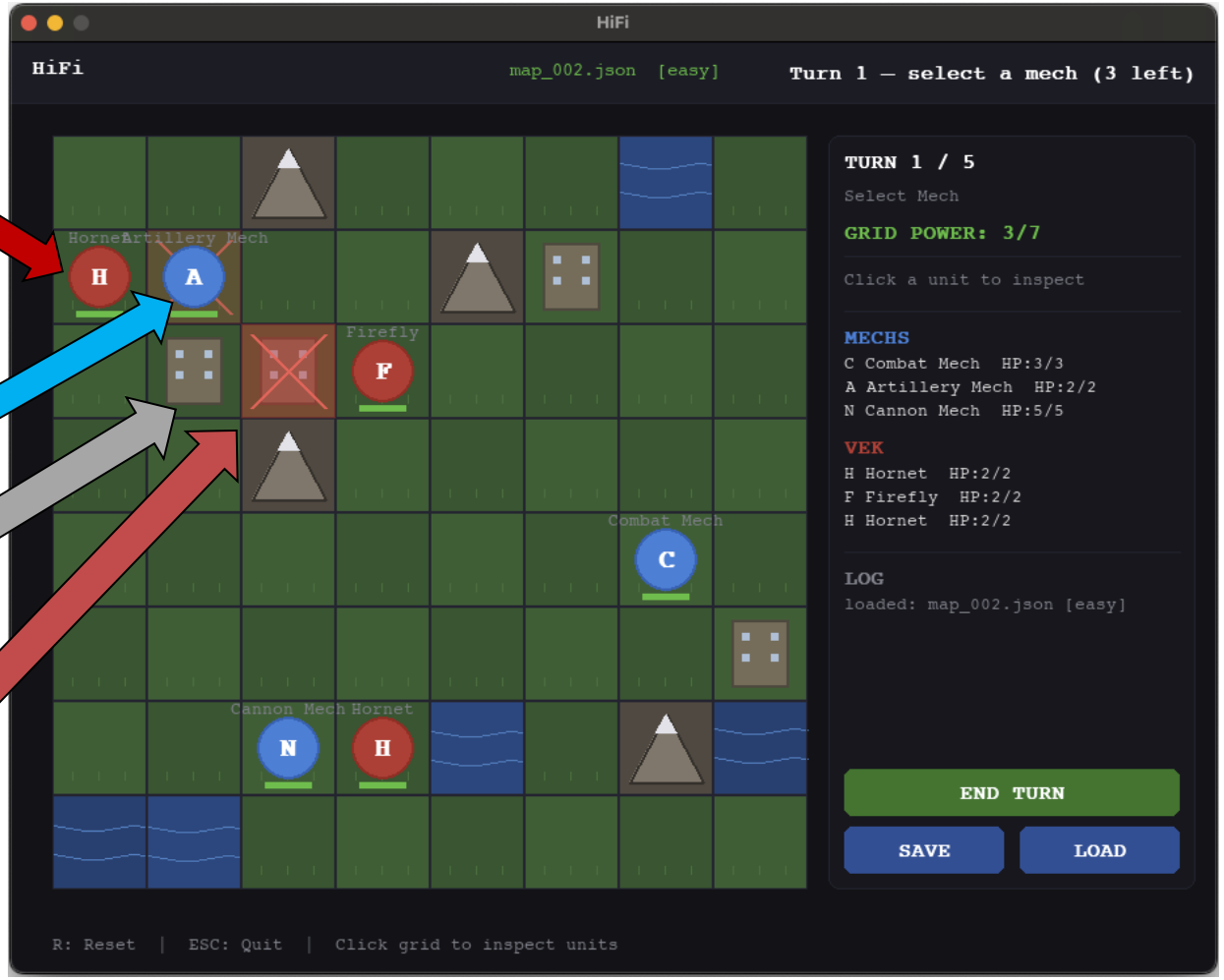
The Game - Inspired by *Into the Breach* by Subset Games

Vek in Red (adversary)

Mech in Blue (player)

Building

Building (targeted by Vek)



Starting Grid Power is always 3

Destroyed building = -1 Grid Power

The Game

- Turn-based puzzle game
- Player: 3 “Mech” robots
- Adversary: 3 “Vek” aliens
- 5 turns for each side – moves/attacks with all 3 units per turn
- 8 x 8 grid with 3 types of terrain and buildings (grid power) that Mech must protect
- Units spawned randomly at start on valid tiles

- **Player Objective - to survive a mission (5 turns)**
  - Have at least 1 grid power AND 1 mech with at least 1 hit point (HP) remaining

- **Adversary Objective - to destroy 3 buildings OR kill all mech within 5 turns**

The Game

- **Player** – 3 “Mech” robots per mission (blue circles)
  - Multi-action turns - 1 move and/or attack per mech per turn
  - Move must occur before attack – cannot move after attack
  - Mechs each have unique HP (health) meters, damage points, movement ranges, attack patterns

- **Adversary** – 3 “Vek” alien insects (red circles)
  - Multi-action turns - 1 attack and/or move per Vek per turn
  - Attack must occur at target position before move and new target chosen
  - Heuristic
    - 75% chance Vek targets nearest building (if in range) to reduce grid power
    - 25% change Vek targets a Mech (if in range) to reduce mech HP

The Game – Inspired by Into the Breach by Subset Games

Mech Move Ranges

Combat Mech: 3

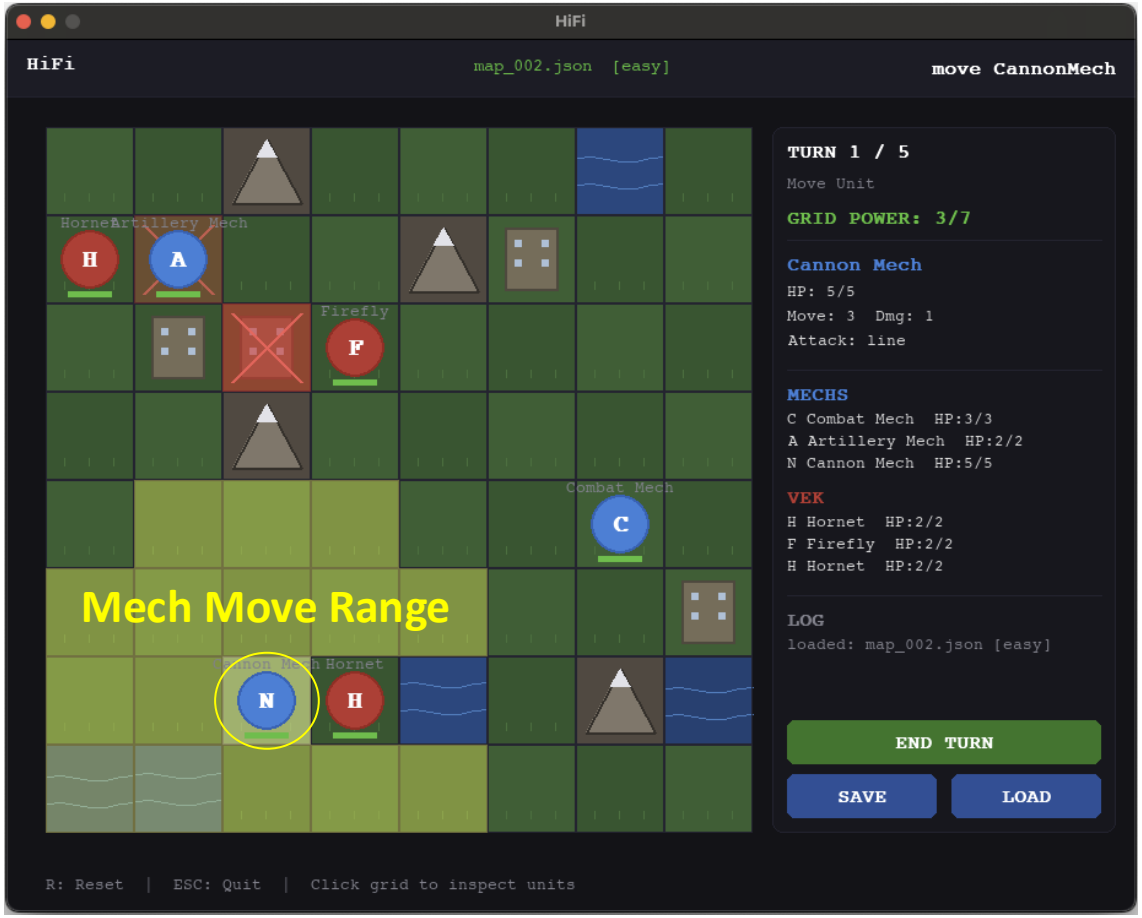
Artillery Mech: 3

Cannon Mech: 3

Vek Move Ranges

Hornet: 2

Firefly: 5



Water kills Hornet instantly

Mountains are obstacles to all but can be converted to grass by Mech with 1 attack damage

The Game – Inspired by Into the Breach by Subset Games

Combat Mech Attack

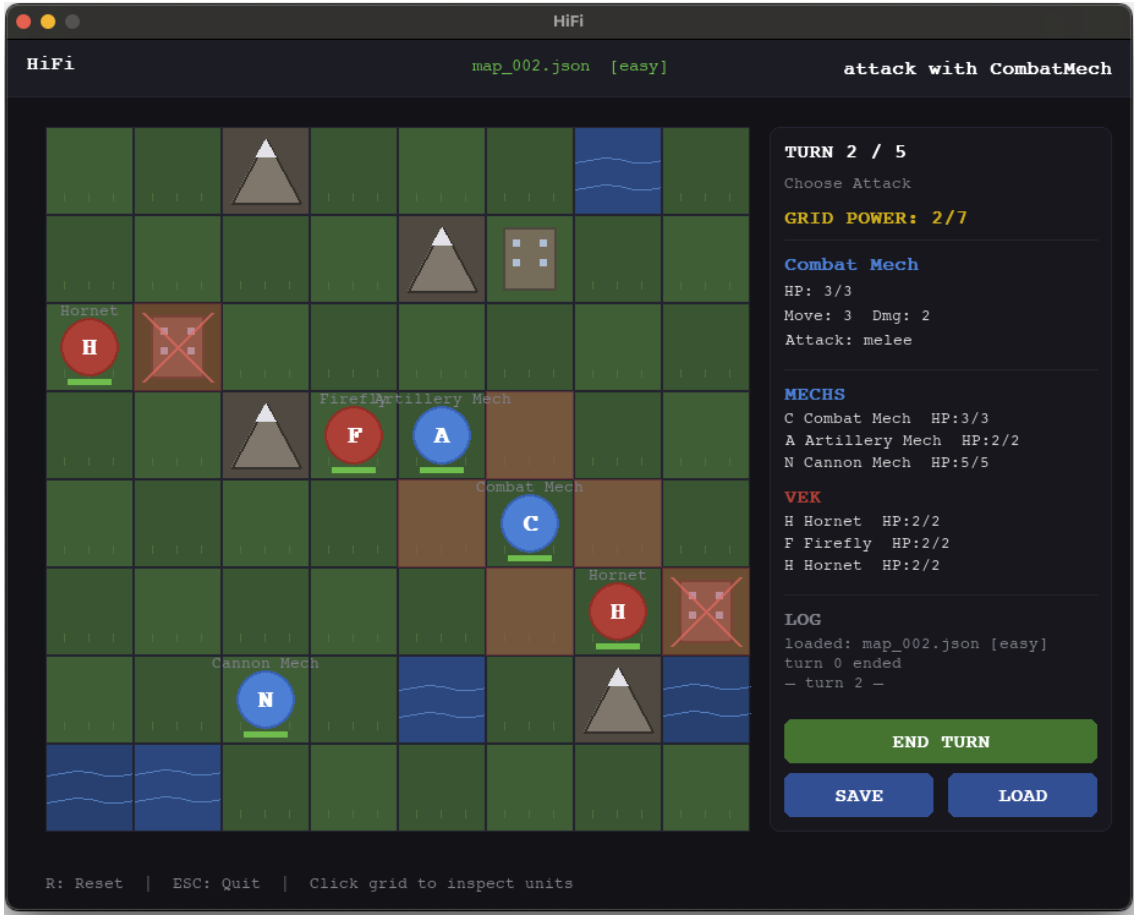
1 space from mech

in one of 4 cardinal directions

Deals 2 HP damage

1 Knockback:

Adjacent unit moves 1 space away from attacked tile, or deals 1 damage if tile occupied or off grid



The Game

Artillery Mech Attack

2 or 3 spaces from mech

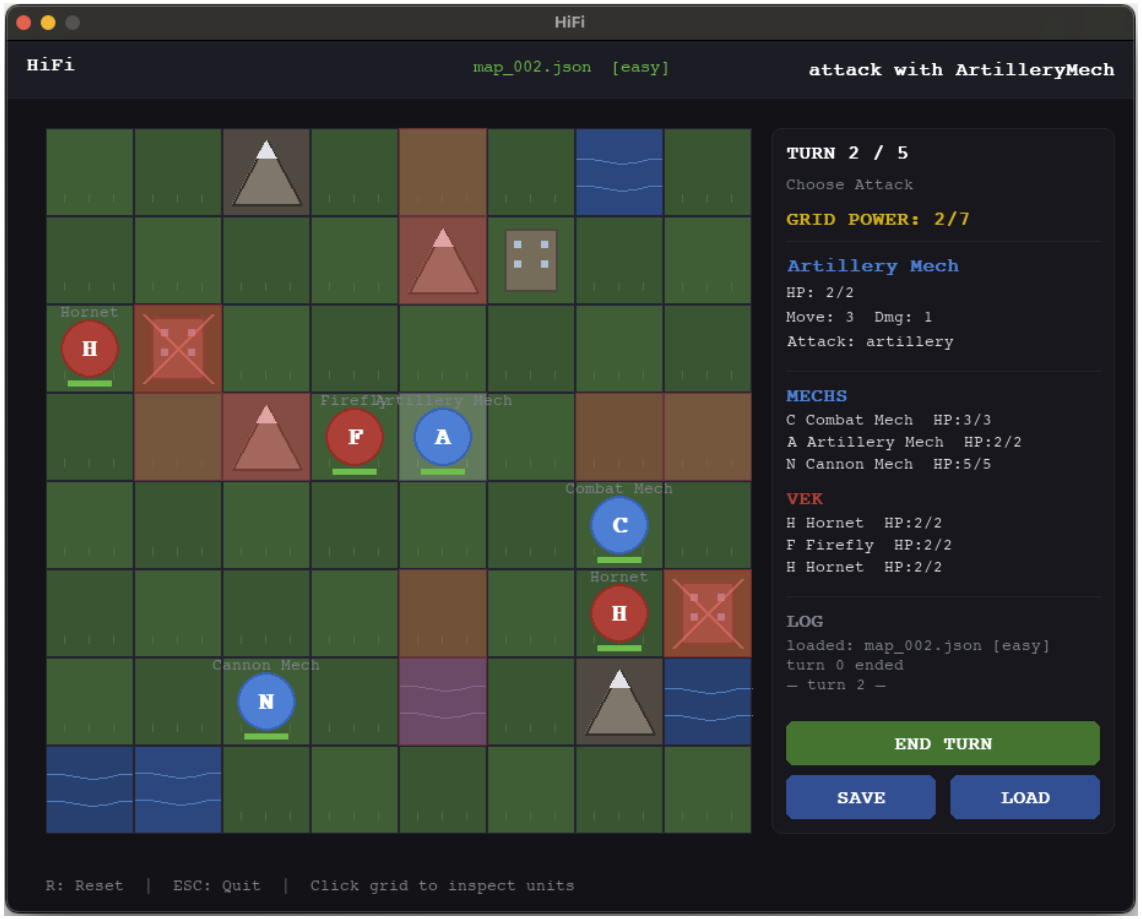
in one of 4 cardinal directions

Can shoot over obstacles

Deals 1 HP damage

1 Knockback:

All adjacent units move 1 space radially from tile, or deals 1 damage if tile occupied/off grid



The Game

Cannon Mech Attack

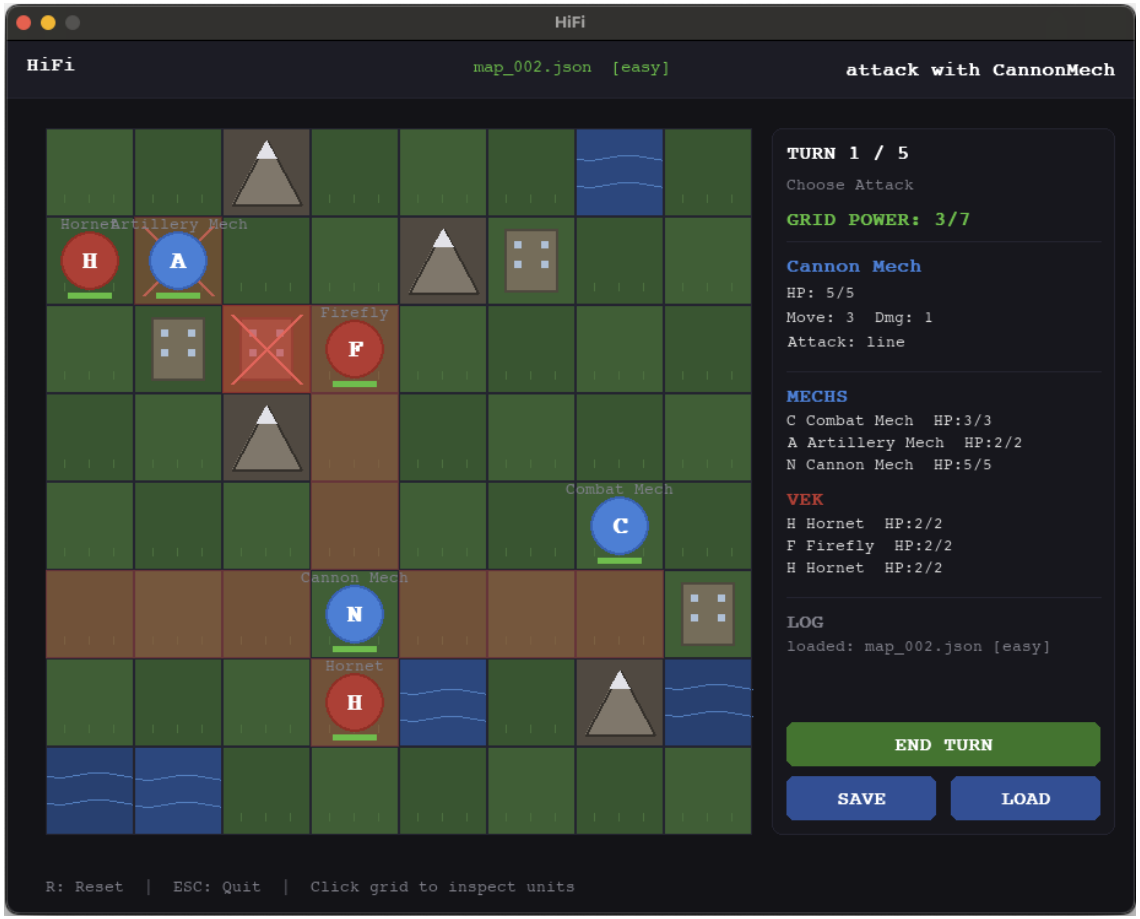
1, 2 and 3 spaces from mech in 1 of 4 cardinal directions

Hits first object in path

Deals 1 HP damage

1 Knockback:

Unit hit moves 1 space along vector of attack, or deals 1 damage if tile occupied/off grid



The Game

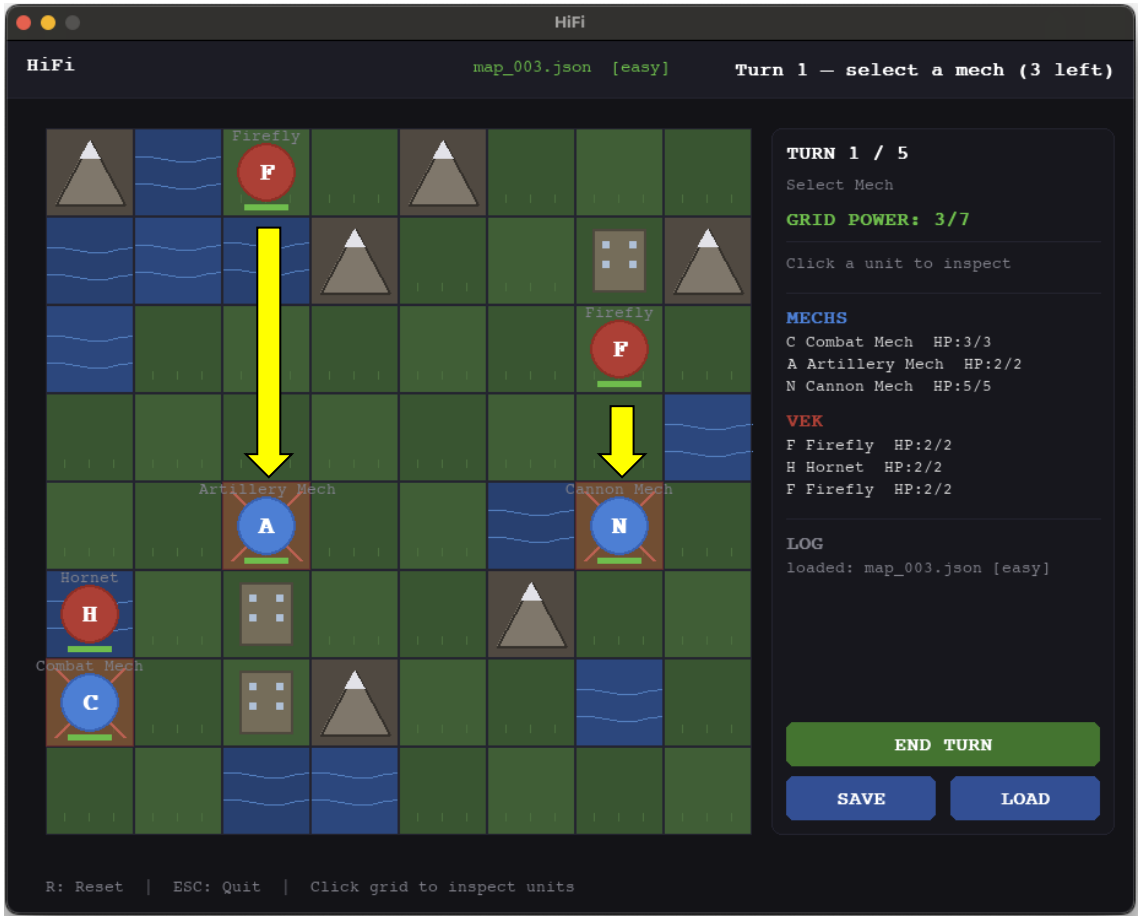
Firefly Vek Attack Range

Up to 7 tiles along one of 4 cardinal directions

Hits first object in path

Deals 1 HP damage

Instant death in water



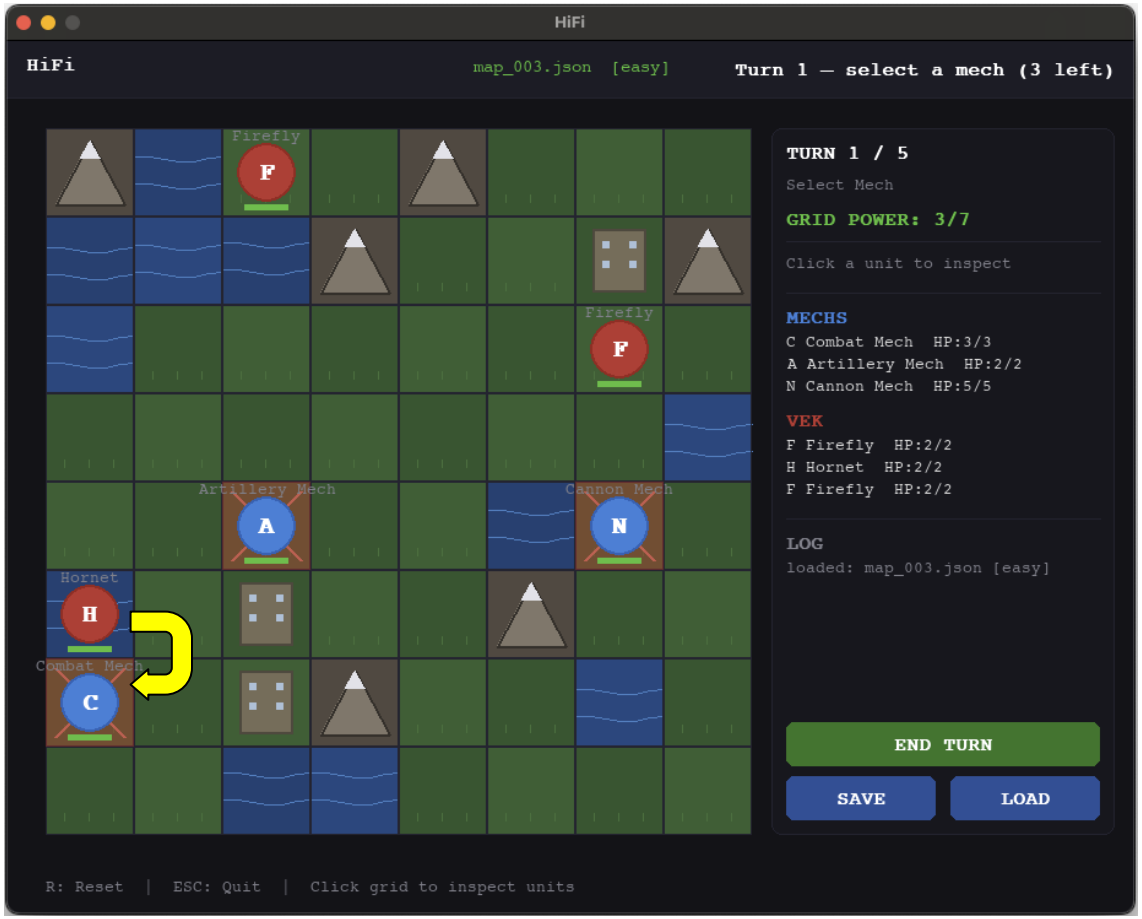
The Game

Hornet Vek Attack Range

1 of 4 adjacent tiles

Deals 1 HP damage

Can attack from water

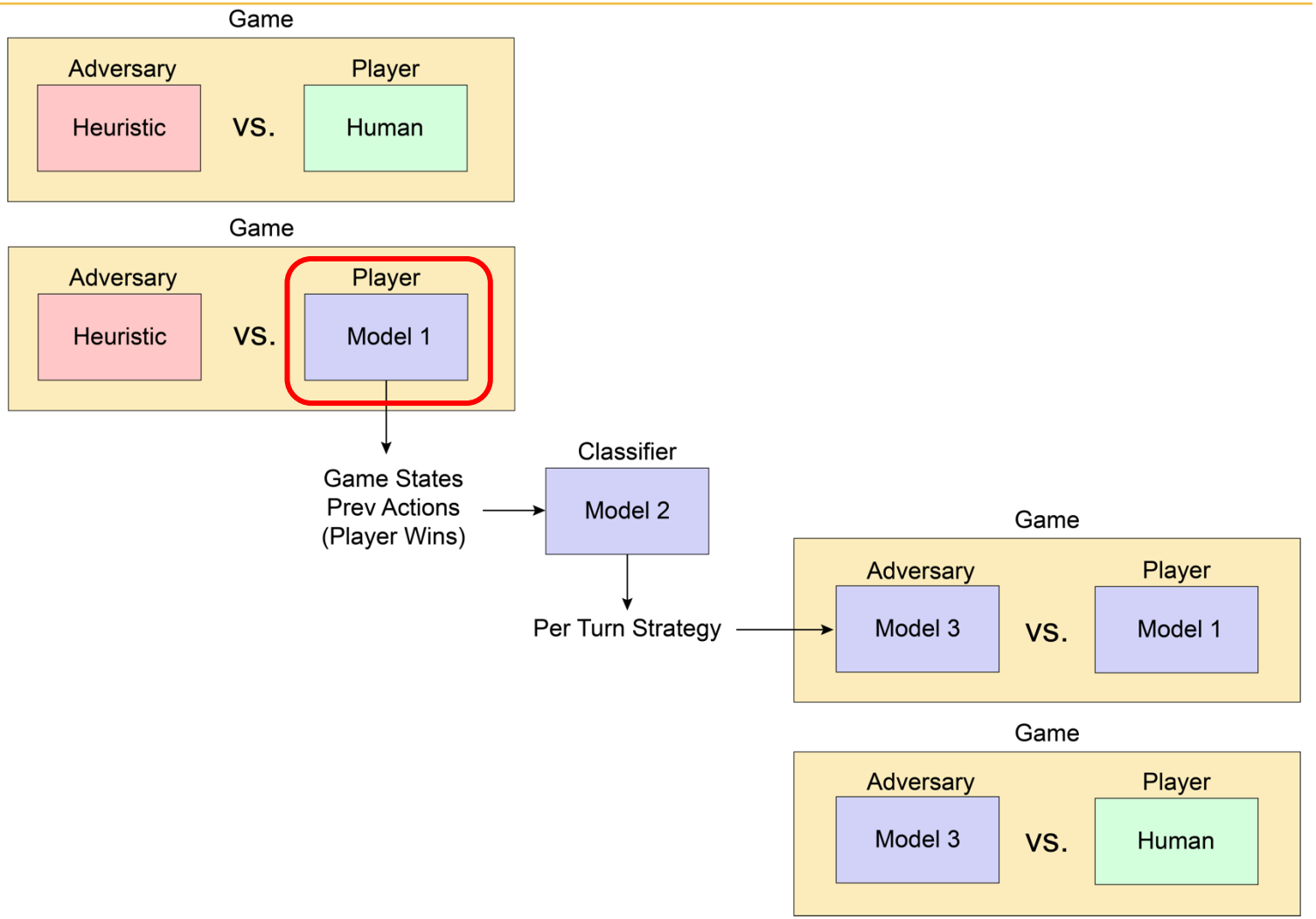


Model 1

Player Agent

(Dataset Generator)

Model 1 – Player Model for Dataset Generation



Model 1 – Proximal Policy Optimization (PPO) Model – Why?

Seminal paper – *Proximal Policy Optimization Algorithms* (Schulman, et al., 2017)

PPO vs Deep Q Network (DQN)

Stochastic policy by design

- PPO samples from a probability distribution - running the same policy with different seeds produces genuinely different action sequences
- DQN is deterministic (argmax Q-value) - running the same DQN on the same map always produces the same game, giving no diversity in the dataset

Reward shaping produces distinct behaviors

- PPO policy directly optimizes the shaped reward, so aggressive vs guardian reward configs produce genuinely different approach to how the agent plays, not just different outcomes
- DQN optimizes Q-values, which are less directly responsive to fine-grained reward shaping

Model 1 – Proximal Policy Optimization (PPO) Model

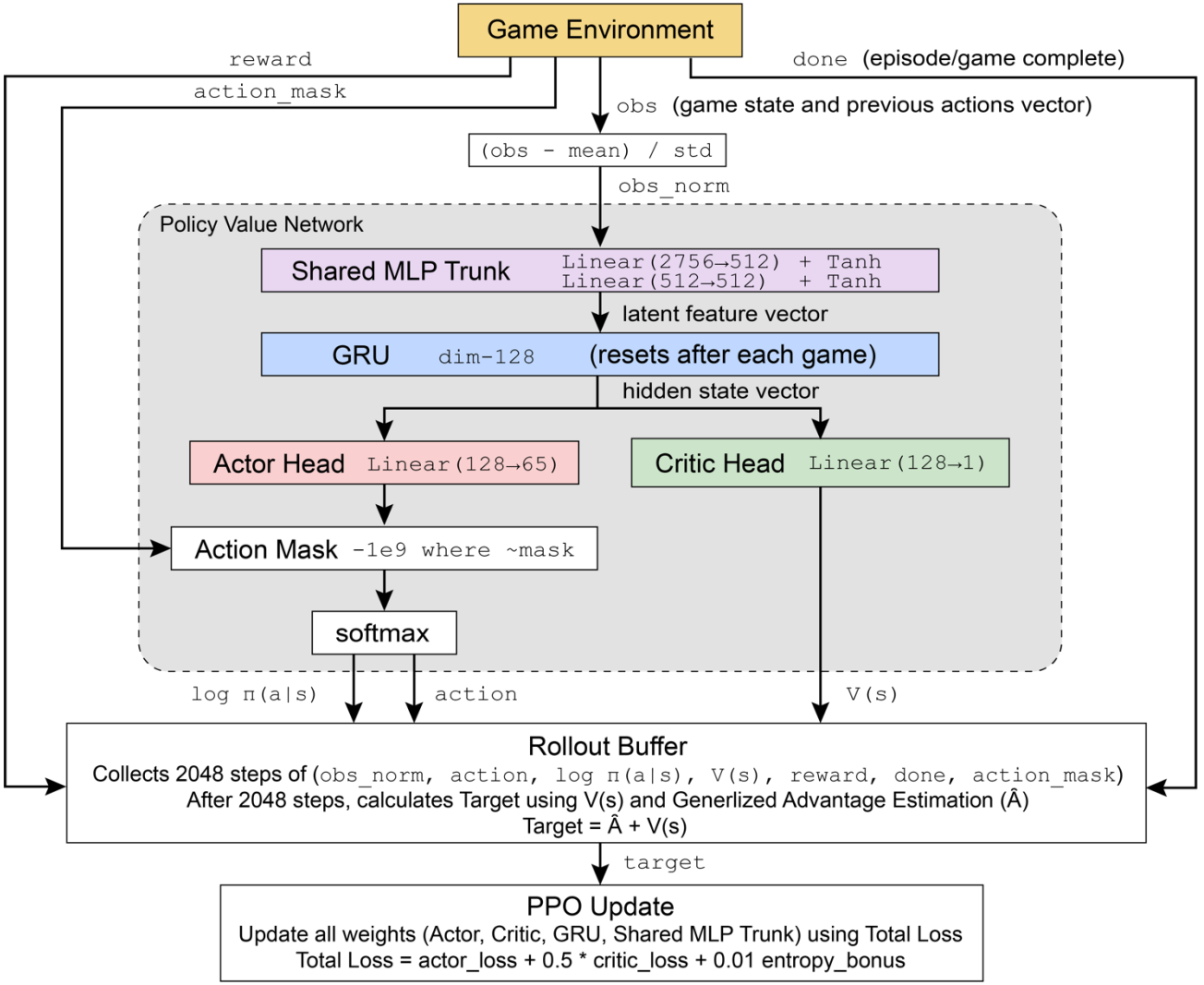
Gated Recurrent Unit (GRU) captures within-turn coherence

- The GRU gives the policy episodic memory
- Decisions within a turn are contextually linked
- Produces trajectories with more coherent tactical sequences
- Richer for the transformer to learn strategy patterns

Entropy bonus maintains exploration

- The entropy bonus keeps the policy from fully collapsing to one action
- Sustains diversity across the 10 collected wins per map

Model 1 – Proximal Policy Optimization (PPO) Model

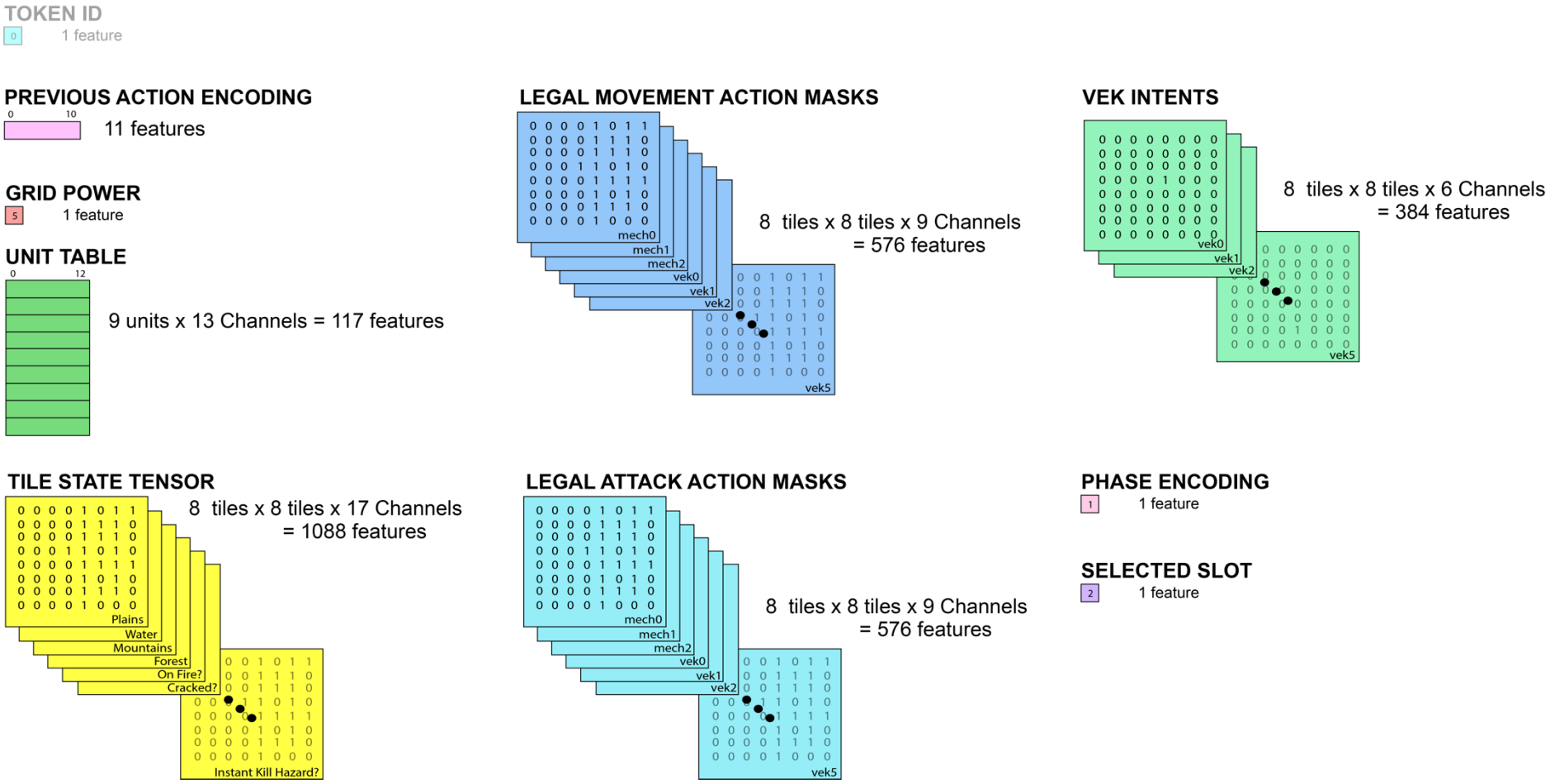


Model 1 – Observation Vector (Game State and Previous Action)

token_id - not used until dataset creation step

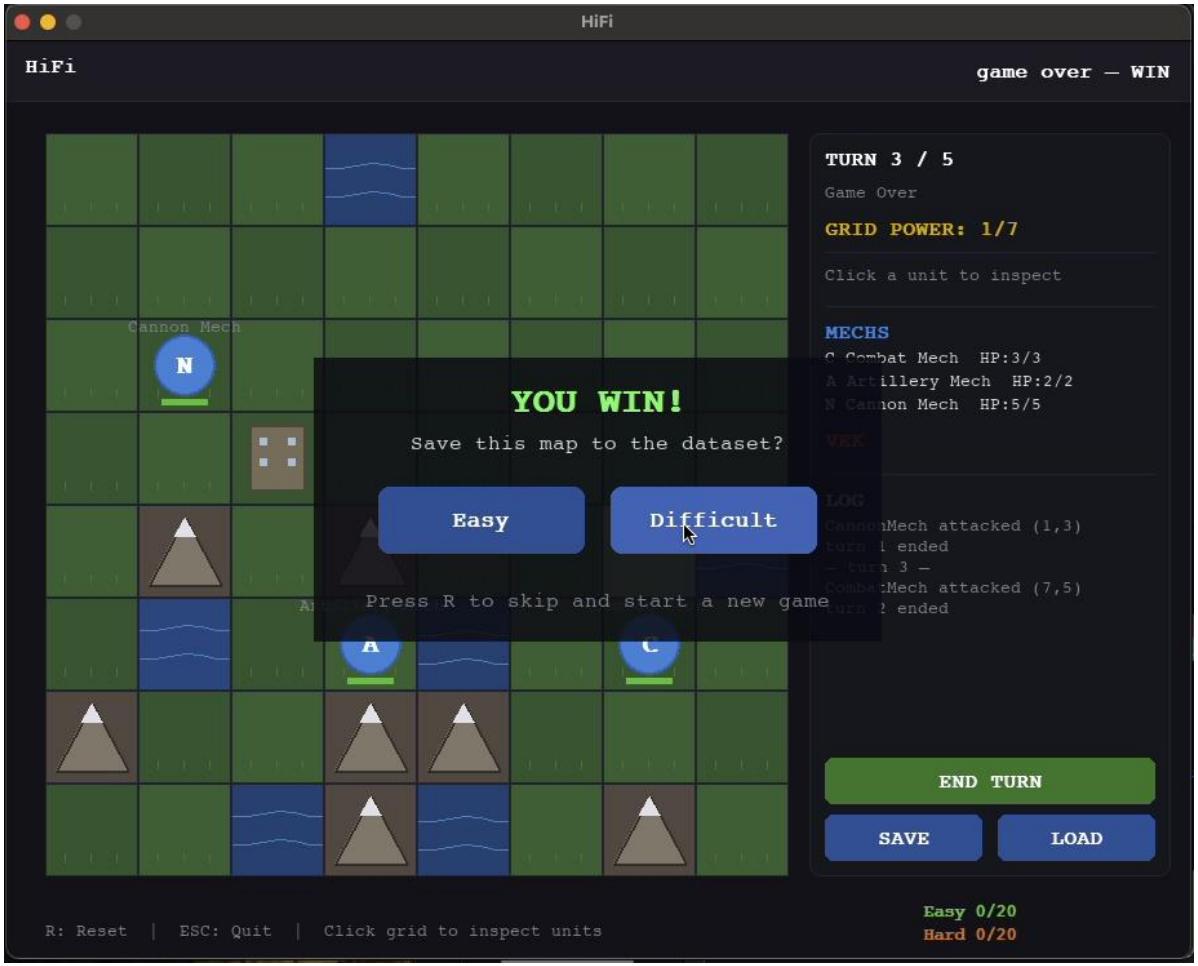
Component	Dims	Description
token_id (fixed 0.0)	1	Placeholder hardcoded to 0; compatibility with data collection format
prev_action vector	11	Encodes the last action taken (kind, unit slot, move tile, attack tile, weapon, kills)
grid_power (scalar)	1	Current number of powered buildings remaining on the map
unit_table (9×13)	117	Per-unit features for all 9 units (position, health, type, team, etc.)
tile_sensor (17×8×8)	1088	Per-tile board features across 17 channels (terrain, units, hazards, etc.)
move_masks (9×8×8)	576	Legal move destinations on the 8×8 grid for each of the 9 units
attack_masks (9×8×8)	576	Legal attack targets on the 8×8 grid for each of the 9 units
vek_intents (6×8×8)	384	Predicted enemy movement/attack intentions for each of the 6 Vek units
phase encoding	1	Current sub-phase: 0=mech_select, 1=mech_move, 2=mech_attack
selected_slot	1	Which mech slot (0/1/2) is currently selected; -1 during mech_select
Total	2756	

Model 1 – Observation Vector (visually represented)



Model 1 – Use only Winnable Map Configuration

- 20 easy maps, 20 difficult maps judged subjectively by human - later combined to one set



Model 1 – How do we know if strategies are being classified?

- Purposefully overtrain Model 1 player agent with N=5 distinct policies on 40 “winnable” maps
- “winnable” means manually won by a human player – initial state stored to JSON file

N = 5 Distinct Reward Systems

Reward	When	Balanced	Aggressive	Defensive	Guardian	Efficient
Vek killed (player phase)	per kill	+0.3	+0.6	+0.15	+0.2	+0.3
Vek killed (enemy phase)	per kill	+0.3	+0.6	+0.15	+0.2	+0.3
Building destroyed (player phase)	per building	-0.2	-0.05	-0.15	-0.6	-0.2
Building destroyed (enemy phase)	per building	-0.2	-0.05	-0.15	-0.6	-0.2
Mech lost (enemy phase)	per mech	-0.3	-0.1	-0.8	-0.2	-0.2
Survived Enemy Phase	per turn	+0.1	+0.05	+0.2	+0.1	+0.02
All mechs acted	per turn	+0.15	+0.2	+0.1	+0.15	+0.3
Idle mech at end turn	per mech	-0.1	-0.15	-0.05	-0.1	-0.2
Win	terminal	+1.0	+1.0	+1.0	+1.0	+1.0
Early win bonus	per turn	+0.0	+0.0	+0.0	+0.0	+0.25
Lose	terminal	-1.0	-0.5	-1.5	-1.0	-1.0

Model 1 – Policies in Words

- Balanced - treats kills, mech survival, and building protection as roughly equal priorities.
- Aggressive - prioritizes killing Vek above everything else. Actively move and attack every turn.
- Defensive - keeps mechs alive at nearly all costs. Heavily penalizes losing the game.
- Guardian - focuses on protecting buildings and grid power above everything else.
- Efficient - tries to win as quickly as possible. Strong pressure to use all mechs every turn.

Model 1 – Training

Training

- Total timesteps = 3,000,000
- Rollout steps = 2,048 (buffer fills before each update)
- Updates per rollout = 4 epochs over the buffer (update_epochs)
- Minibatch size = 256 transitions per gradient step
- Total PPO updates ≈ 3,000,000 / 2,048 × 4 ≈ 5,859 update passes

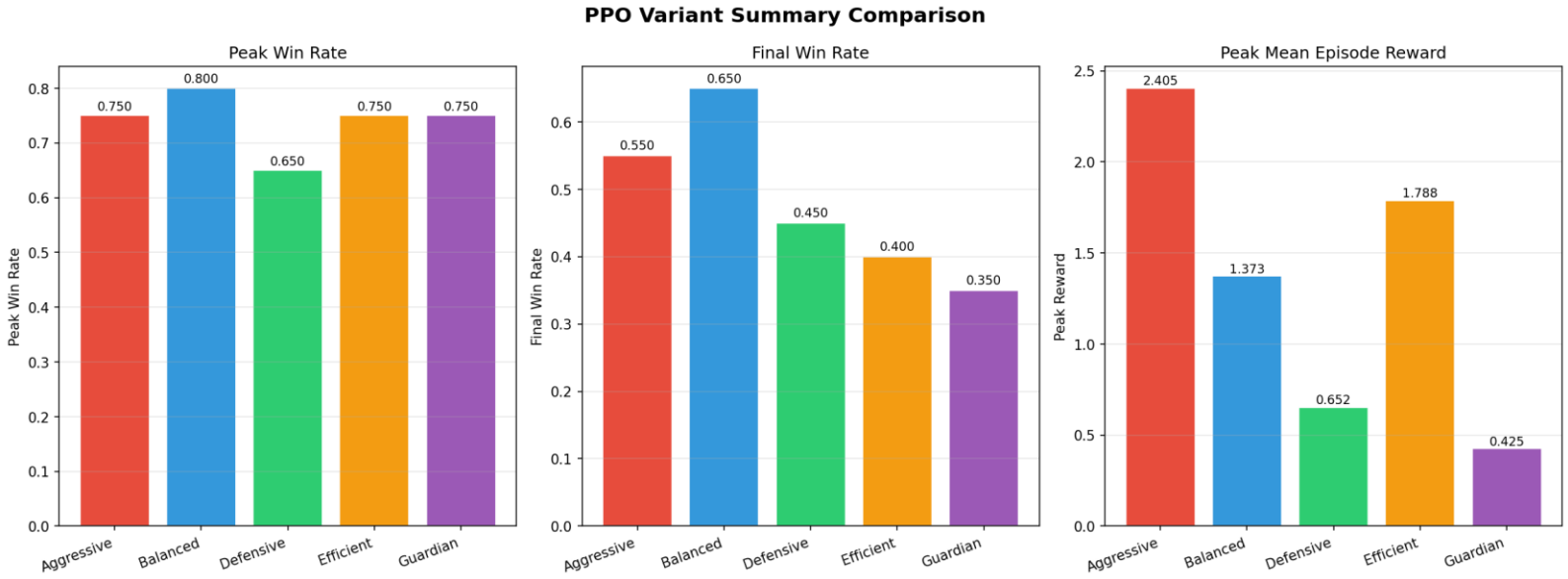
Model 1 – Training

Other key training settings

- Learning rate = 3e-4
- Gradient clipping = 0.5 (max_grad_norm)
- Patience = 50 updates without improvement (early stopping active)
- Min improvement = 0.01
- Dropout = none (PPO MLP has no dropout)
- GAE $\lambda = 0.95$
- γ (discount) = 0.99
- Clip $\varepsilon = 0.2$
- Value loss coefficient = 0.5
- Entropy coefficient = 0.05
- Hidden sizes = (512, 512)
- GRU hidden size = 128

Model 1 – Training

Version Trained for Model 2.2



Model 1 – Each Winning Game to .h5 file

- Win 10 games per 40 maps per 5 policies and store 52 action tokens per game
- Max 500 attempts per map to collect 10 winning games

Dataset	Shape	dtype	Description
token_ids	(52,)	int8	Token type per step (START, MOVE, ATTACK, etc.)
prev_action	(52, 11)	float32	Previous action encoding at each step
global_scalar	(52,)	float32	Grid power at each step
unit_table	(52, 9, 13)	float32	Per-unit features at each step
tile_sensor	(52, 17, 8, 8)	float32	Board state at each step
move_masks	(52, 9, 8, 8)	float32	Legal moves at each step
attack_masks	(52, 9, 8, 8)	float32	Legal attacks at each step
vek_intents	(52, 6, 8, 8)	float32	Vek intent masks at each step
pad_mask	(52,)	bool	True = padding (no real token)
push_seq	(52,)	int8	Number of Vek pushed at ATTACK tokens
bld_save_seq	(52,)	int8	Building-targeting Vek killed/pushed at ATTACK tokens

Attributes:

result - "WIN", "LOSE"
n_real_tokens - how many of the 52 slots contain real data
variant_label - 0-4 integer identifying the variant (added after saving)
map_id - e.g. "easy_map_003" (added after saving)

Model 1 – Datasets gathered from Model 1 vs Heuristic Adversary

Theory

5 policy variants × 40 maps × 10 wins = 2,000 total winning games

Reality

Not all maps were winnable by the policies

Some “winnable” maps could not be won by any policy even though won by human

Map	Missing Variant Policies (Models 2.1 and 2.3)	Missing Variant Policies (Model 2.2)
difficult_map_008	guardian, efficient	
difficult_map_015	balanced, aggressive, defensive, efficient	aggressive
difficult_map_020	aggressive, guardian, efficient	guardian
easy_map_012	balanced, defensive, guardian	balanced, defensive, guardian
easy_map_013	defensive	
easy_map_018	All (no policies could win)	All (no policies could win)

Model 1 – Datasets gathered from Model 1 vs Heuristic Adversary

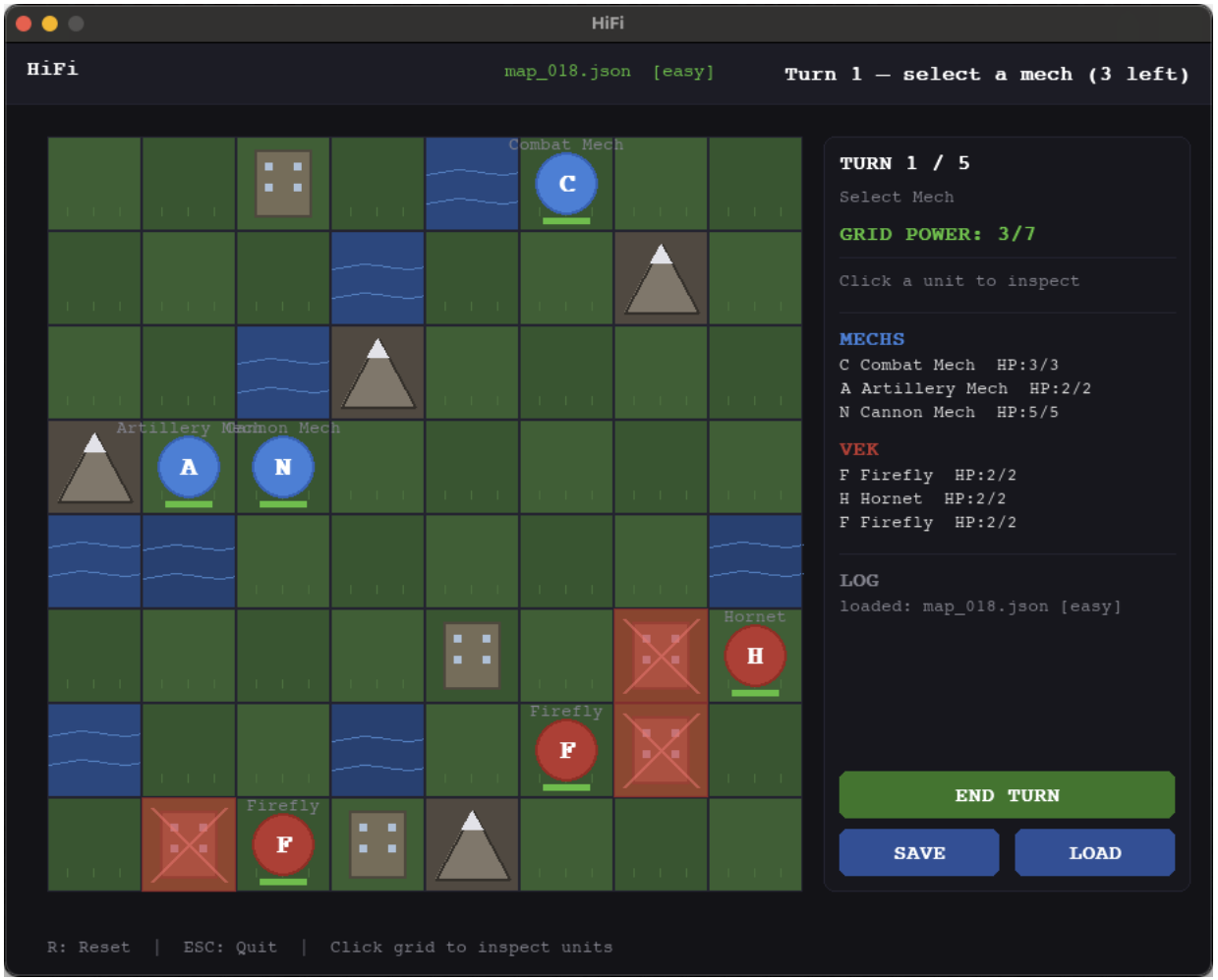
easy/map_018.json

One of 2 moves required for survival of turn 1

Obvious for experienced human player

Difficult for policies to learn

Never won by any policy



Model 2 – Datasets gathered from Model 1 vs Heuristic Adversary

Solution

- Split data collected as planned (80% / 10% / 10%) for training, validation, and test
- Models 2.1 and 2.3 – 1440 / 180 / 180 games
- Model 2.2 – 1500 / 190 / 190 games

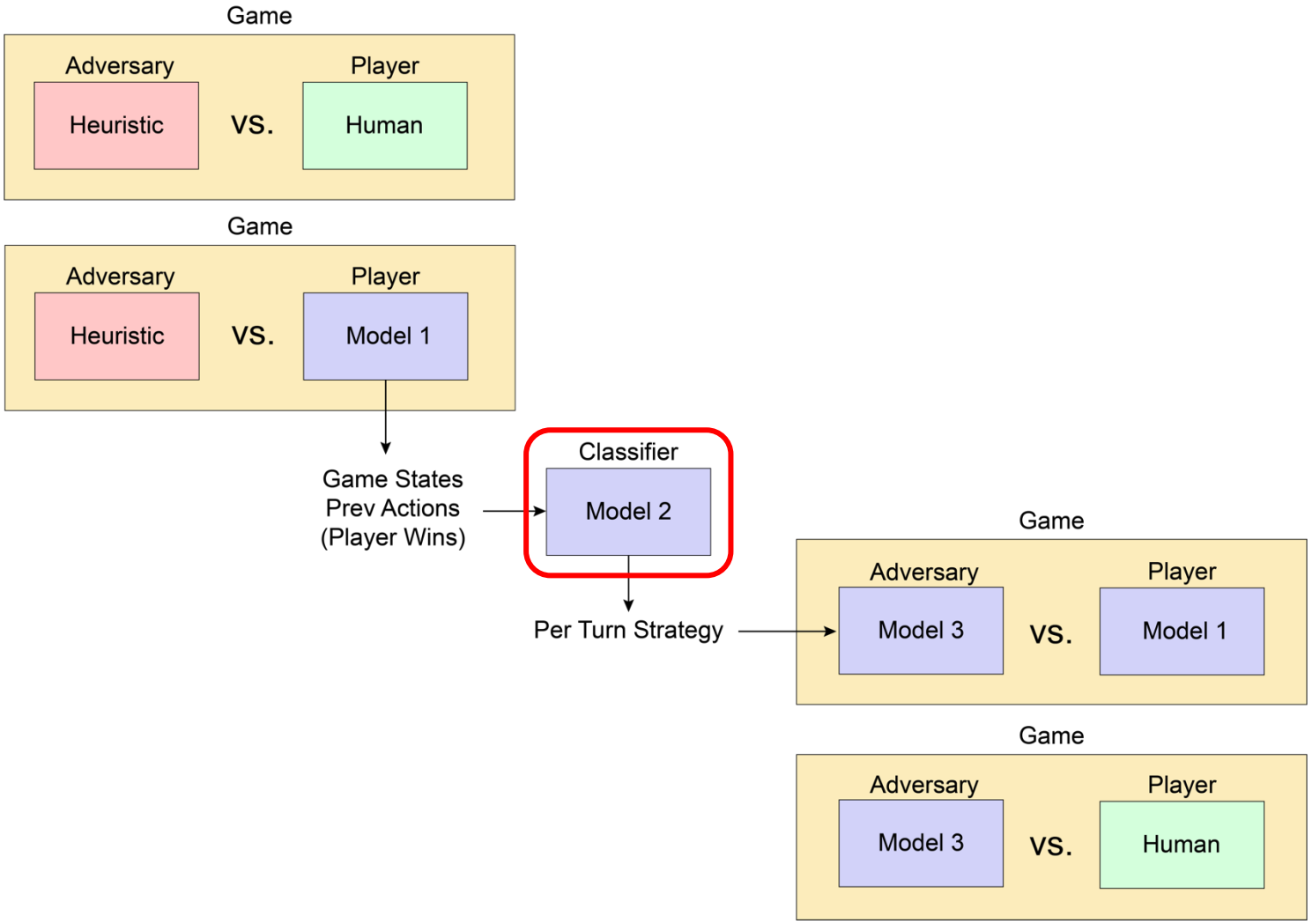
Model 1 – Merge Files into Set for Training, Validation and Test

- **Collect** - all .h5 files from each variant
- **Shuffle** - all files from all 5 variants are pooled into one list and randomly shuffled (seed=42)
- **Split** - sliced into three merged files:
  - dataset_train.h5 — 80% missions
  - dataset_validation.h5 — 10% missions
  - dataset_test.h5 — 10% missions
- **Write** - each split is written as a single large .h5 file where each dataset gains an extra leading dimension N (missions). So tile_sensor goes from (52, 17, 8, 8) per mission to (N, 52, 17, 8, 8) in the merged file.
- **Add Metadata** - 2 arrays are added: variant_label and map_ids.

Model 2

Strategy Classifier

Model 2 – Strategy Classifier Model for Training Expert Adversaries



Vector Quantized Variational Autoencoder (VQ-VAE)

Why is VQ-VAE a good fit?

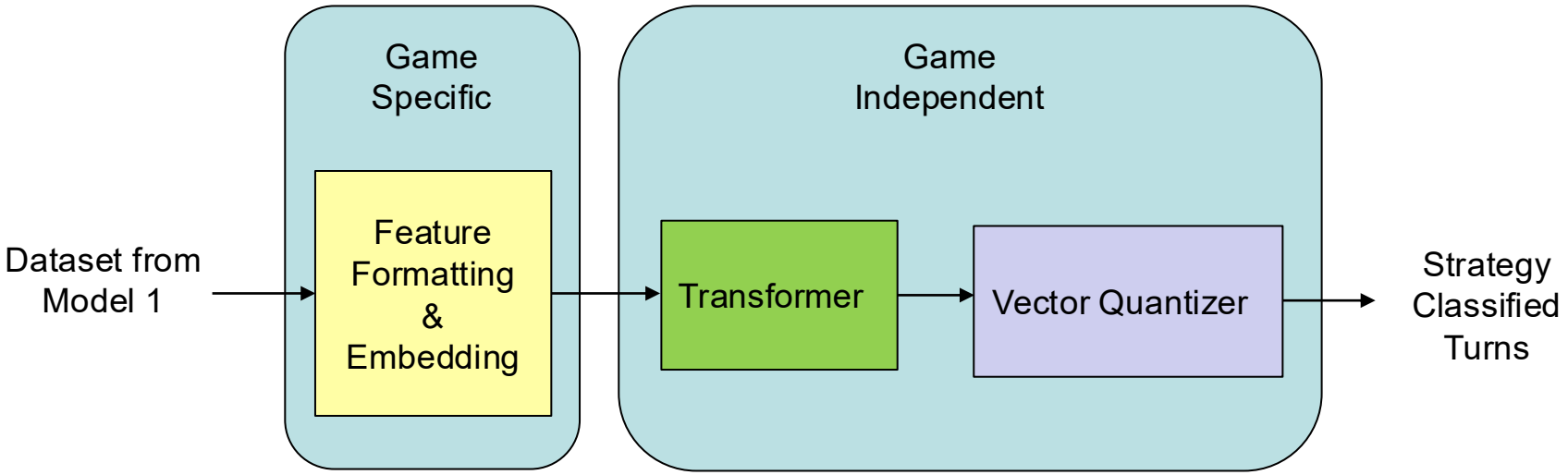
- Learns the latent representation of strategy and clusters it
- Clusters are task-aware
- Handles sequential structured (game actions as turns)
- Works like slow-moving K-means

Vector Quantized Variational Autoencoder (VQ-VAE)

Why not use K-means?

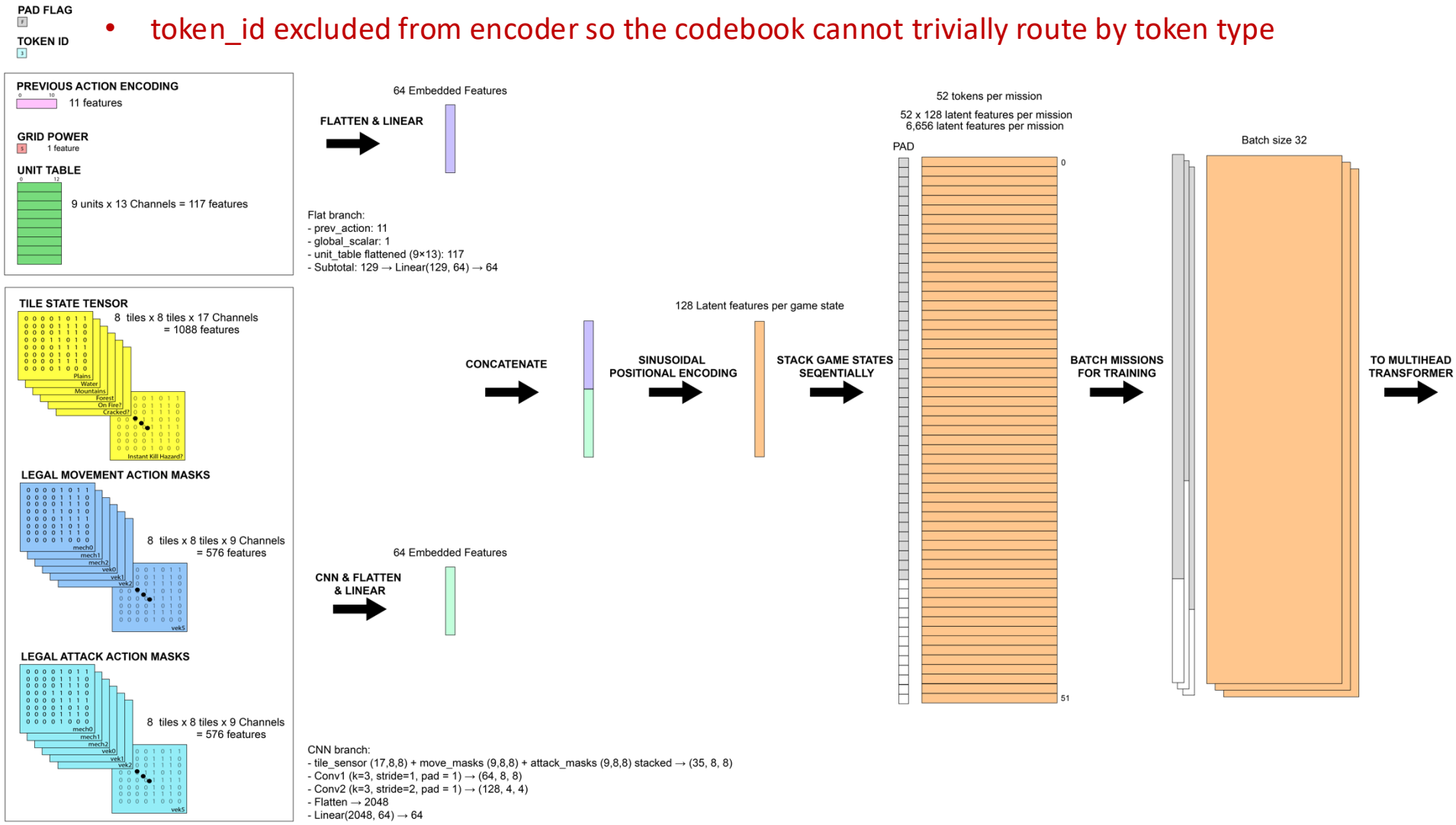
- K-means
  - Groups similar points
  - Expensive in high dimensions
  - Unstable with large datasets - final result depends heavily on initial centroids
  - clusters in whatever space you give - garbage in, garbage out
- VQ-VAE
  - Learns what matters, then groups based on that
  - Trained with Stochastic Gradient Descent (SGD) – batched data
  - The latent space representation is optimized for clustering

Model 2 – Vector Quantized Variational Autoencoder (VQ-VAE)



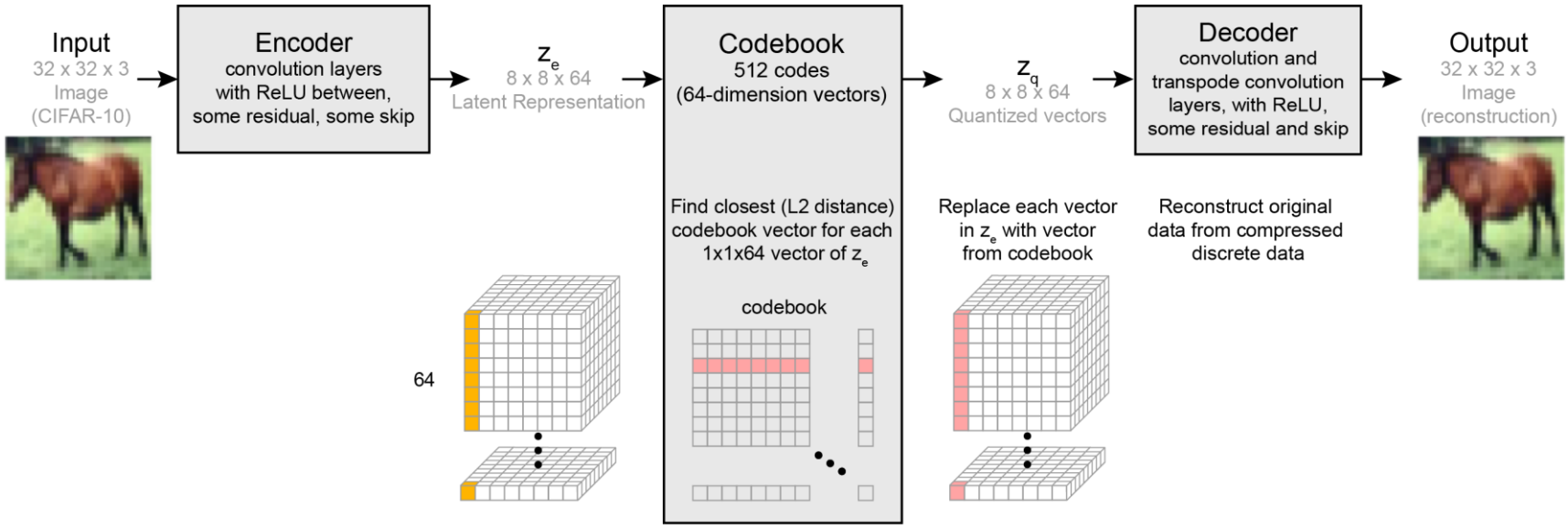
Model 2 – Token Data Embedding

- token_id excluded from encoder so the codebook cannot trivially route by token type



Vector Quantized Variational Autoencoder (VQ-VAE)

Seminal paper - *Neural Discrete Representation Learning* (van den Oord, et al., 2017)

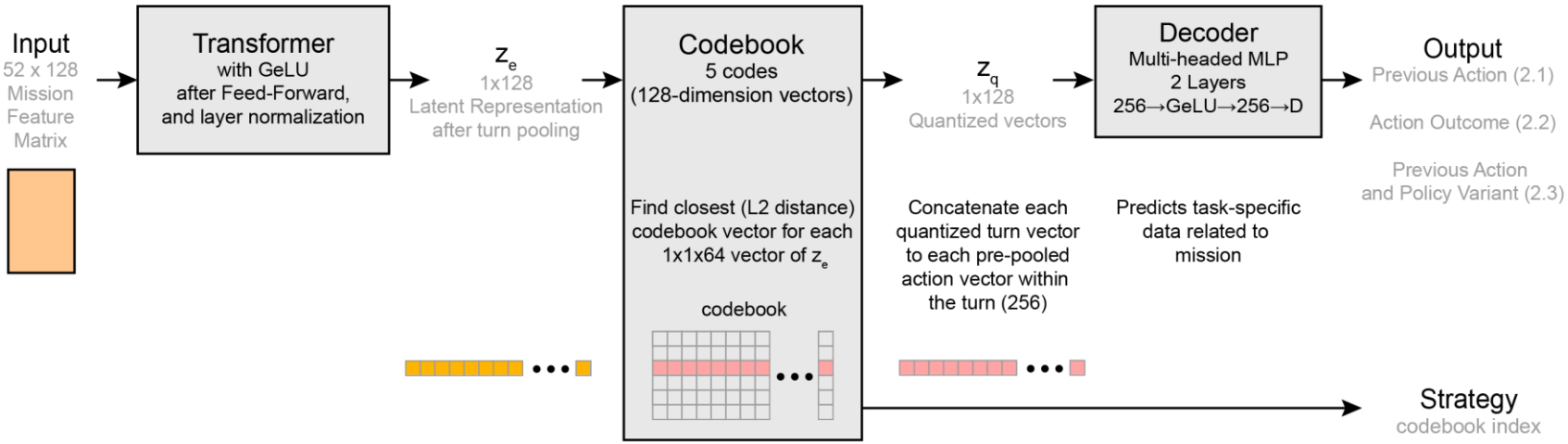


$$\mathcal{L} = \underbrace{\log p(x \mid z_q(x))}_{\text{reconstruction}} + \underbrace{\|\text{sg}[z_e(x)] - e_k\|^2}_{\text{codebook}} + \underbrace{\beta\|z_e(x) - \text{sg}[e_k]\|^2}_{\text{commitment}}$$

	reconstruction	codebook	commitment
Encoder	✓	✗	✓
Codebook	✗	✓	✗
Decoder	✓	✗	✗

Model 2 VQ-VAE

EMA in VQ-VAE - *Generating diverse high-fidelity images with VQ-VAE-2* (Razavi et al. (2019)



$$\mathcal{L} = \underbrace{\log p(x \mid z_q(x))}_{\text{reconstruction}} + \underbrace{\beta \| z_e(x) - \text{sg}[e_k] \|^2}_{\text{commitment}}$$

	reconstruction	commitment
Encoder	✓	✓
Decoder	✓	✗

Codebook Updated by
Exponential Moving Average (EMA)

*codebook vectors are initialized with uniform random values in the range [-1/K, +1/K] where K = n embeddings = 5, so [-0.2, 0.2]

Exponential Moving Average (EMA) for Codebook Updates

Generating Diverse High-Fidelity Images with VQ-VAE-2 (Razavi et al., 2019)

EMA turns codebook learning into stable, low-noise clustering instead of noisy gradient chasing

VQ-VAE

gradient descent with codebook loss

$$e_k \leftarrow e_k + \eta(z_e(x) - e_k)$$

VQ-VAE-2

exponential moving average

$$e_k \leftarrow \frac{\text{running sum of assigned } z_e}{\text{running count}}$$

$$N_k^{(t)} = \gamma N_k^{(t-1)} + (1-\gamma)\, n_k^{(t)}$$

$$m_k^{(t)} = \gamma m_k^{(t-1)} + (1-\gamma) \sum_{i=1}^{n_k^{(t)}} z_e^{(i)}$$

$$e_k^{(t)} = \frac{m_k^{(t)}}{N_k^{(t)}}$$

e_k is the k-th vector in the codebook (embedding table)

Exponential Moving Average (EMA) for Codebook Updates

Why EMA instead of codebook loss?

- Codebook loss creates a tug-of-war between the encoder and codebook
  - each is a gradient signal pulls against the other, making convergence sensitive to relative learning rates.

- EMA provides direct statistical tracking of encoder outputs, so the codebook adapts smoothly without exerting a counterforce on the encoder.

Why GeLU vs ReLU or ELU or Tanh?

- ReLU hard-kills negative inputs, causing dying neurons
- Tanh/ELU saturates at extremes - causes vanishing gradients in deep nets.
- GeLU weights inputs by probability of being positive - smooth, non-dying, and acts as implicit regularization.

GeLU provides stable gradients across many layers with widely varying input magnitudes

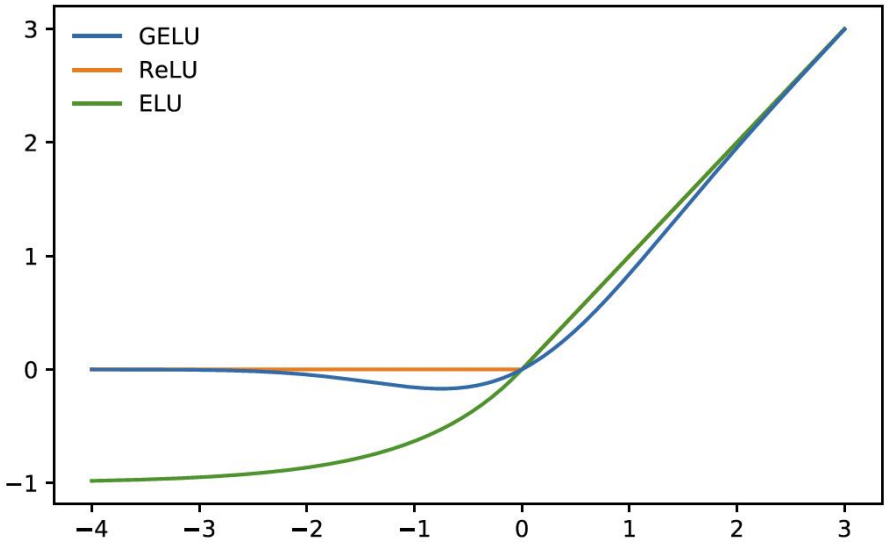
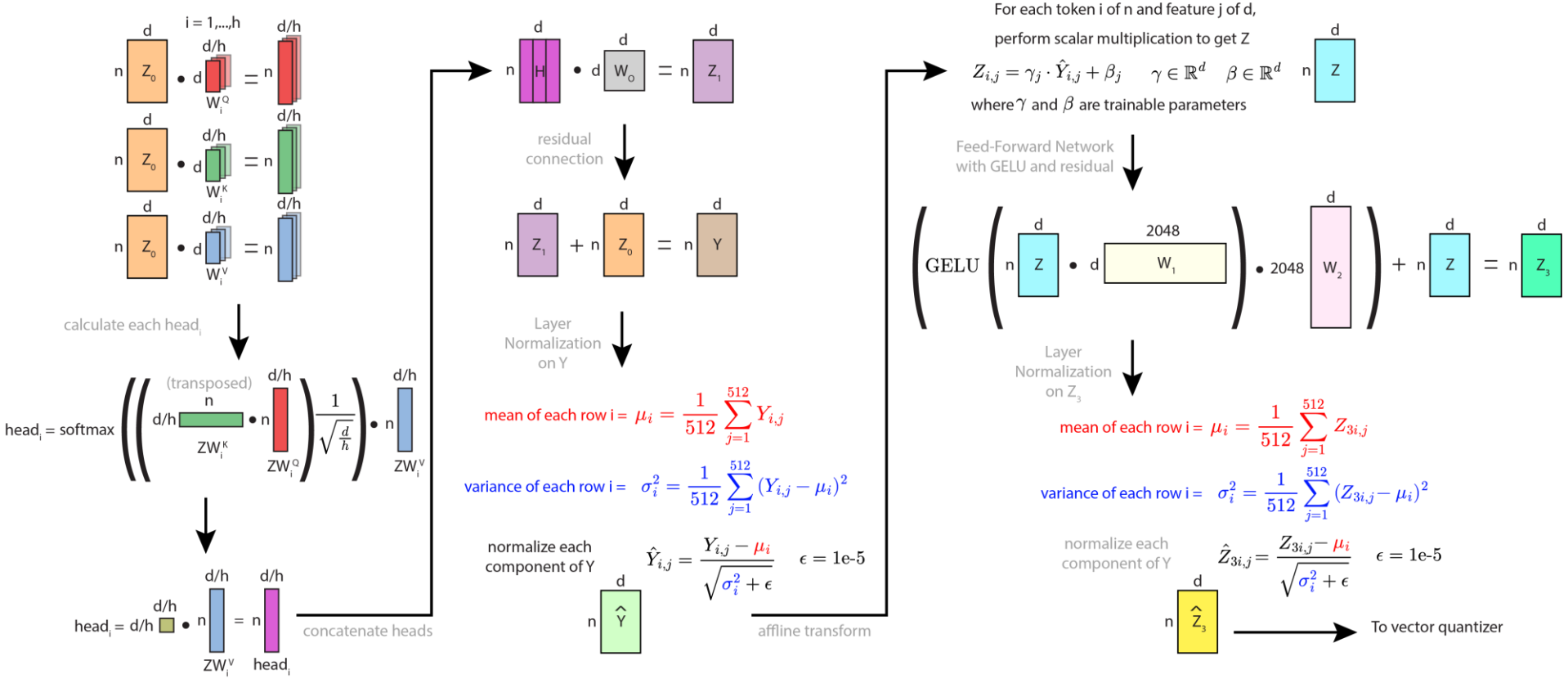


Image Source: *Gaussian Error Linear Units (GELUs)* (Hendrycks & Gimpel, 2016)

Model 2 – VQ-VAE Transformer

*Bias values are rolled into the matrices for diagram simplicity (e.g. $Z = [1\ Z]$, $W_1 = [b_1\ W_1]^T$)



$$\text{head}_i = \text{softmax}\left(\left(\left(Z_0 W_i^K\right)\left(Z_0 W_i^Q\right)\right)\frac{1}{\sqrt{\frac{d}{h}}}\right)\left(Z_0 W_i^V\right)$$

$$\text{head}_i = \text{d/h} \cdot \left(Z_0 W_i^V\right) = \text{head}_i$$

$$H \cdot W_O = Z_1$$

$$Z_1 + Z_0 = Y$$

mean of each row i = $\mu_i = \frac{1}{512}\sum_{j=1}^{512} Y_{i,j}$

variance of each row i = $\sigma_i^2 = \frac{1}{512}\sum_{j=1}^{512}(Y_{i,j} - \mu_i)^2$

normalize each component of Y $\hat{Y}_{i,j} = \frac{Y_{i,j} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \quad \epsilon = 1\text{e-}5$

$$Z_{i,j} = \gamma_j \cdot \hat{Y}_{i,j} + \beta_j \qquad \gamma \in \mathbb{R}^d \quad \beta \in \mathbb{R}^d$$

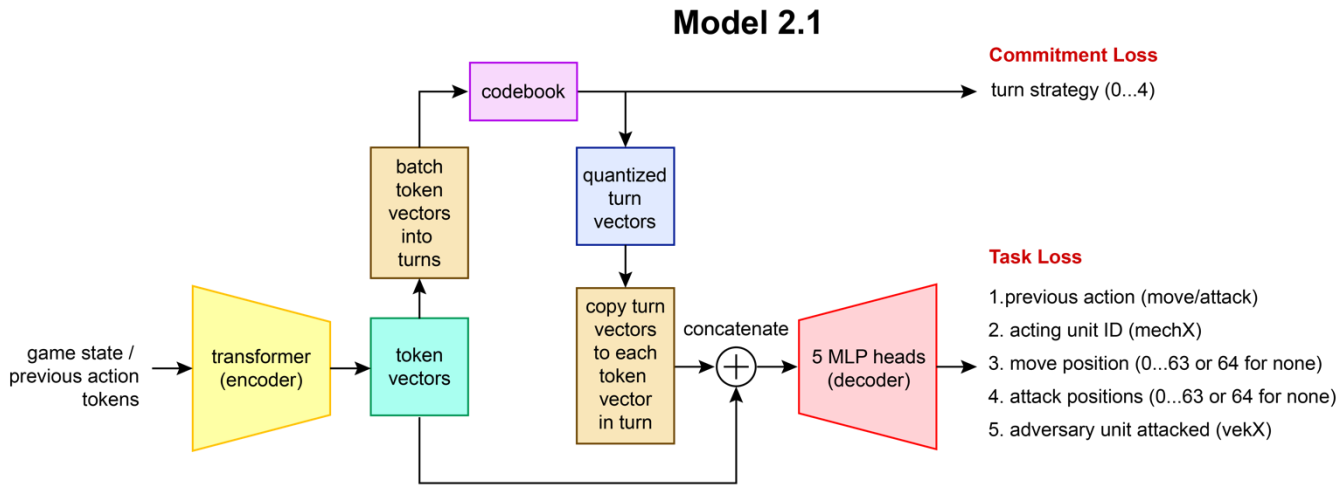
$$\text{GELU}\left(Z \cdot W_1\right) \cdot W_2 + Z = Z_3$$

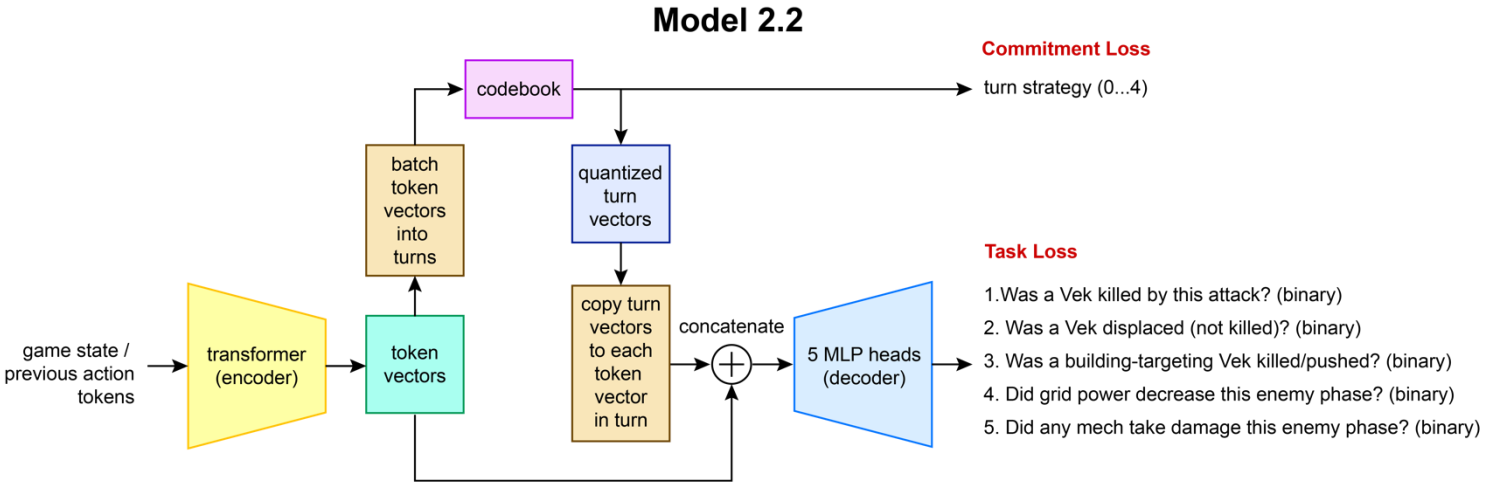
mean of each row i = $\mu_i = \frac{1}{512}\sum_{j=1}^{512} Z_{3i,j}$

variance of each row i = $\sigma_i^2 = \frac{1}{512}\sum_{j=1}^{512}(Z_{3i,j} - \mu_i)^2$

normalize each component of Y $\hat{Z}_{3i,j} = \frac{Z_{3i,j} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \quad \epsilon = 1\text{e-}5$

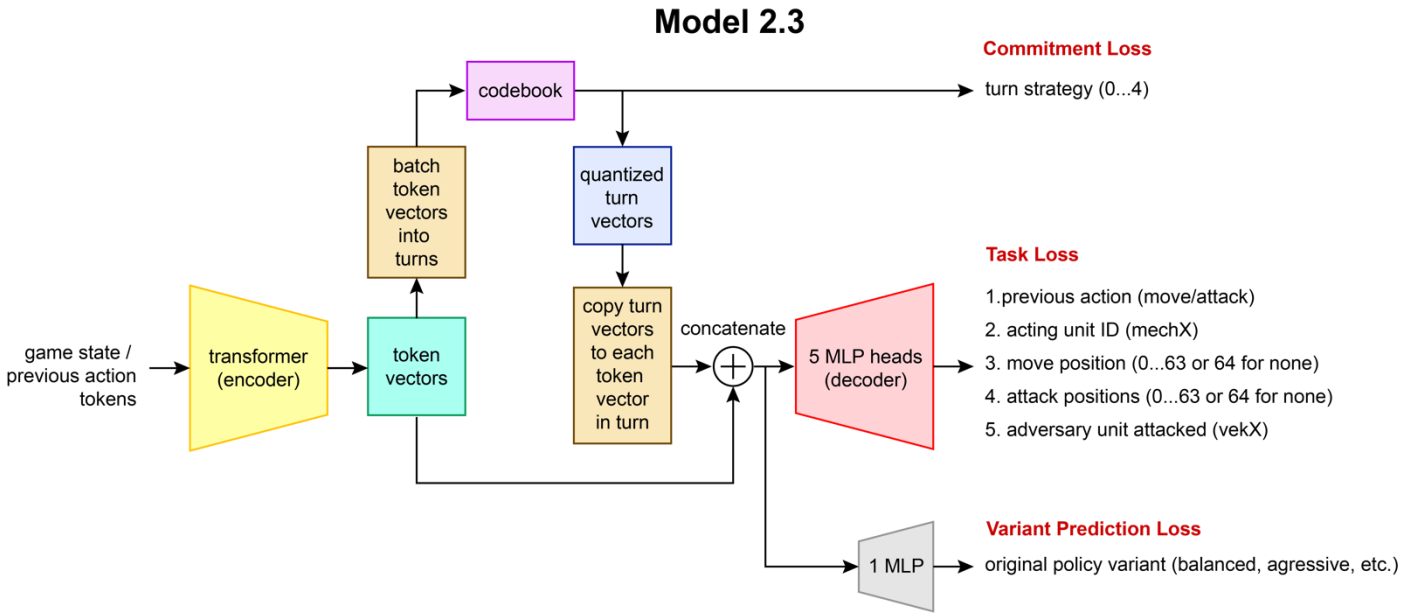
Model 2 – Three Variants





Model 2 – Three Variants

How does prediction task at decoder affect the resulting clusters? (see slide 58)



Model 2 – Variant Classification Loss

Cross-Entropy

$$\mathcal{L}\text{variant} = -\sum b \log p(\hat{y}_b = y_b)$$

$$\mathcal{L}\text{variant} = -\frac{1}{N}\sum_{b=1}^{N} \log \frac{\exp(l_{b,y_b})}{\sum_{k=1}^{K} \exp(l_{b,k})}$$

b batch index (0 to 31)

$\hat{y}_b$ predicted variant label (0 to 4)

y_b true variant label (0 to 4)

$l_{b,k}$ logit for class k from the variant MLP head

N number of valid non-padded missions in the batch (up to 52)

K number of codebook entries (5)

Model 2 – Training

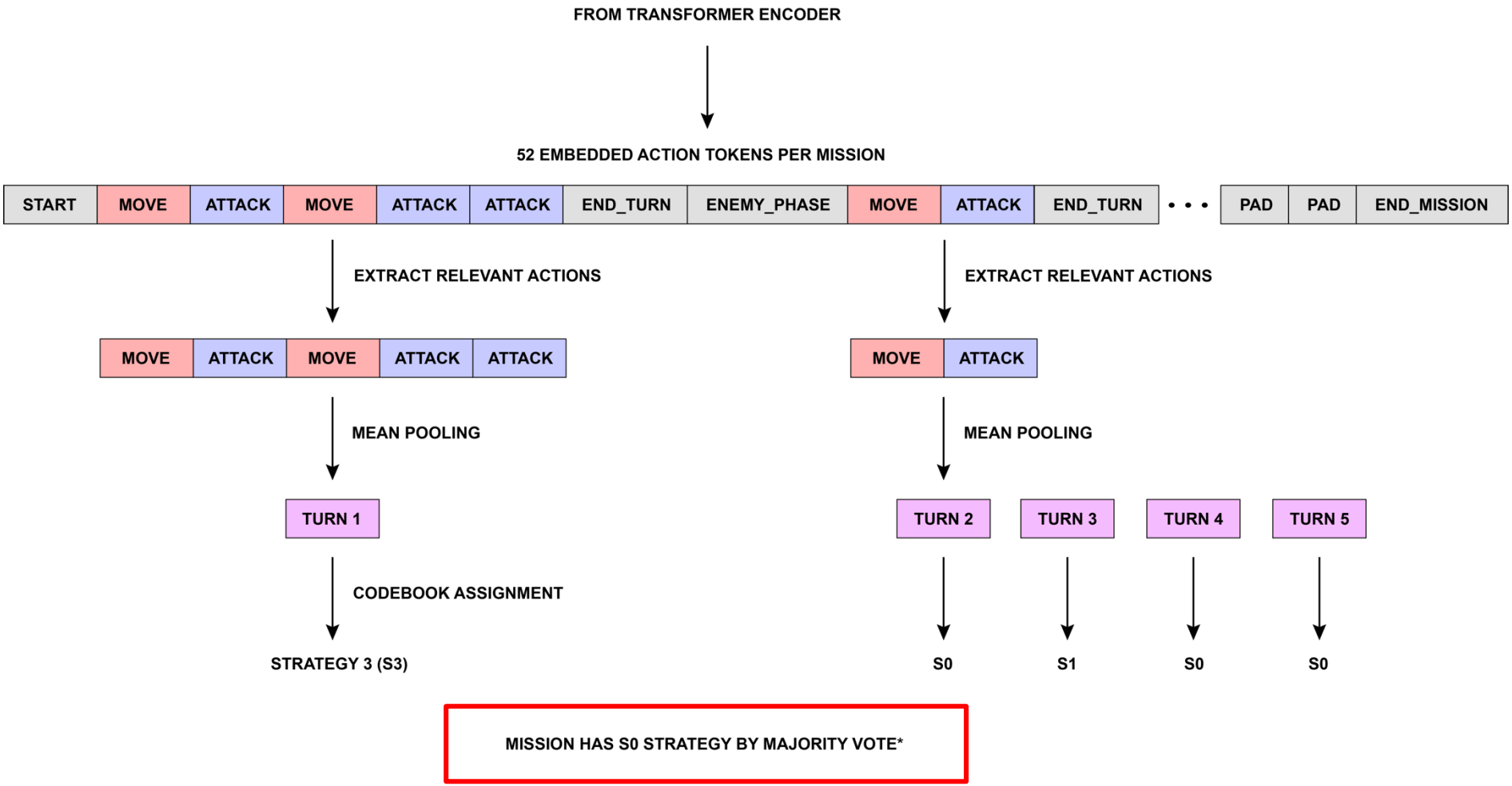
Training

- 150 epochs
- 45 steps/epoch
- 32 missions per batch (1 step)
- Total training steps = 150 × 45 = 6,750 steps

Other key training settings

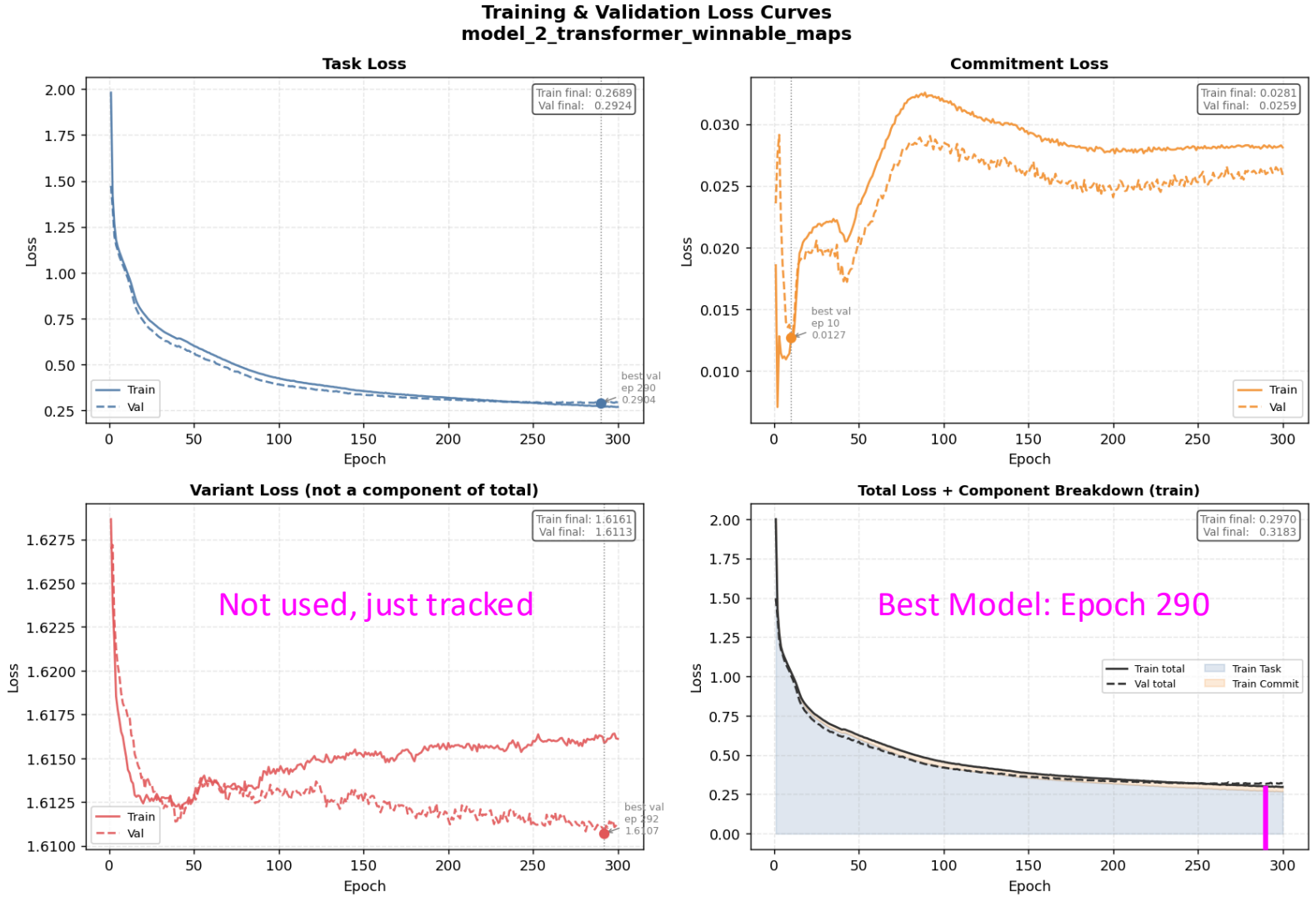
- Learning rate = 1e-4
- Weight decay = 1e-4
- Gradient clipping = 1.0
- Patience = 150 (it always trains the full 150 epochs, no early stopping)
- Dropout = 0.1
- EMA decay = 0.95 (fast codebook updates, tuned for the small dataset)

Model 2 – How turns and missions are classified by strategy



*Tie break goes to lowest indexed strategy

Model 2.1 – Convergence (Total Validation Loss)

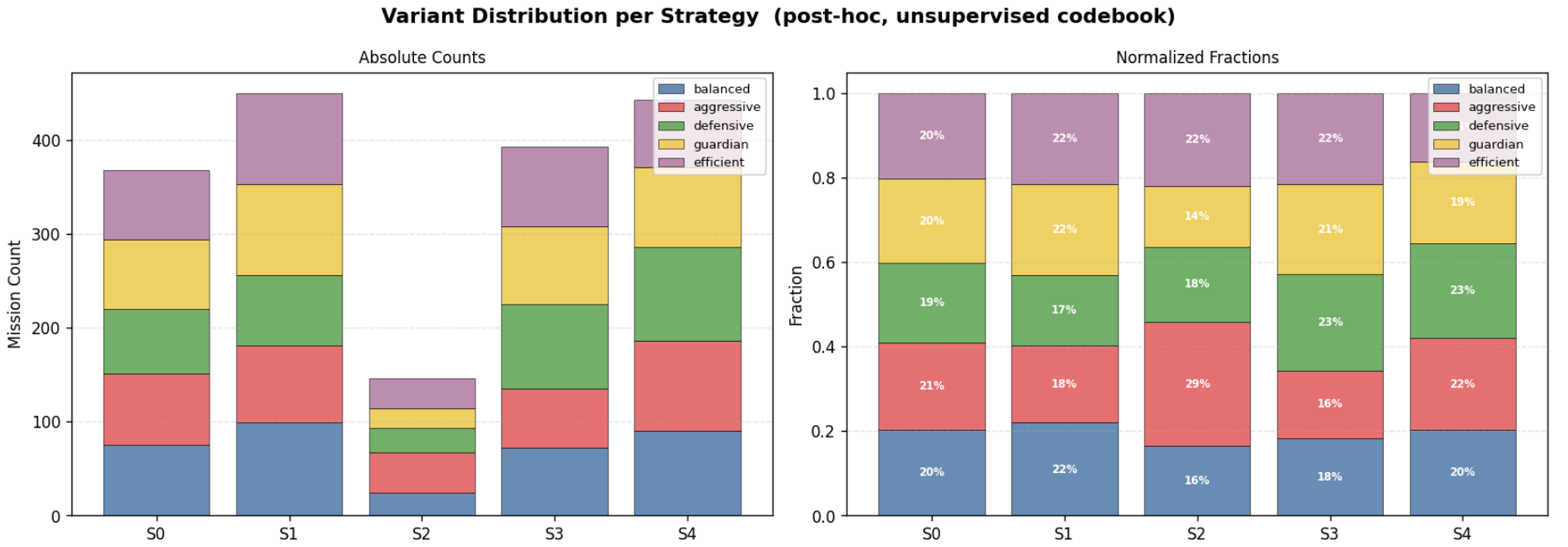


Model 2.1 Results – Strategy by Mission Variant Distribution

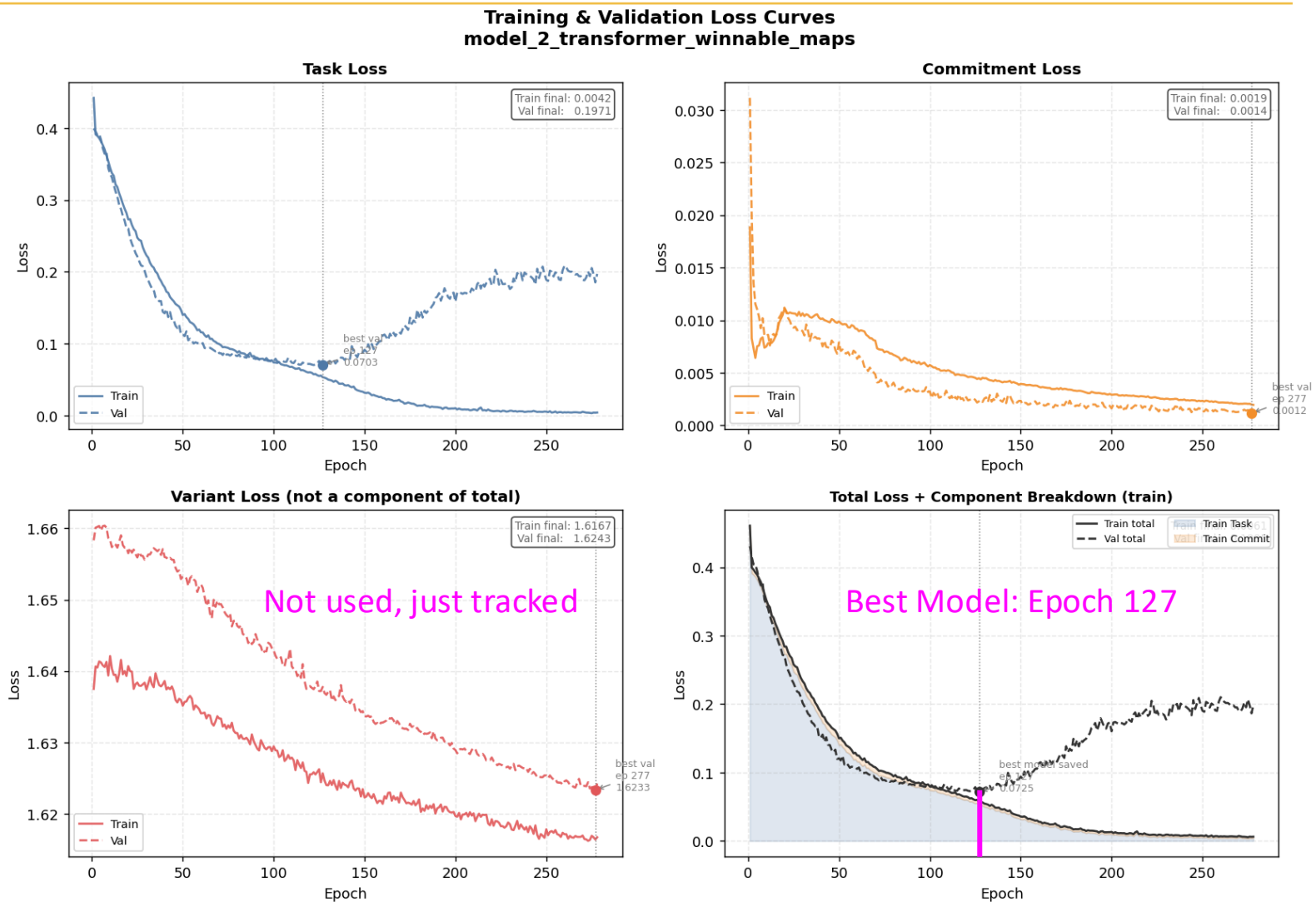
Codebook classifies turns - each mission gets one strategy label assigned by majority vote of turns

Near-uniform mixture of all five injected policy variants (14–29% each)

S2 contains significantly fewer missions than other strategies



Model 2.2 – Convergence (Total Validation Loss)



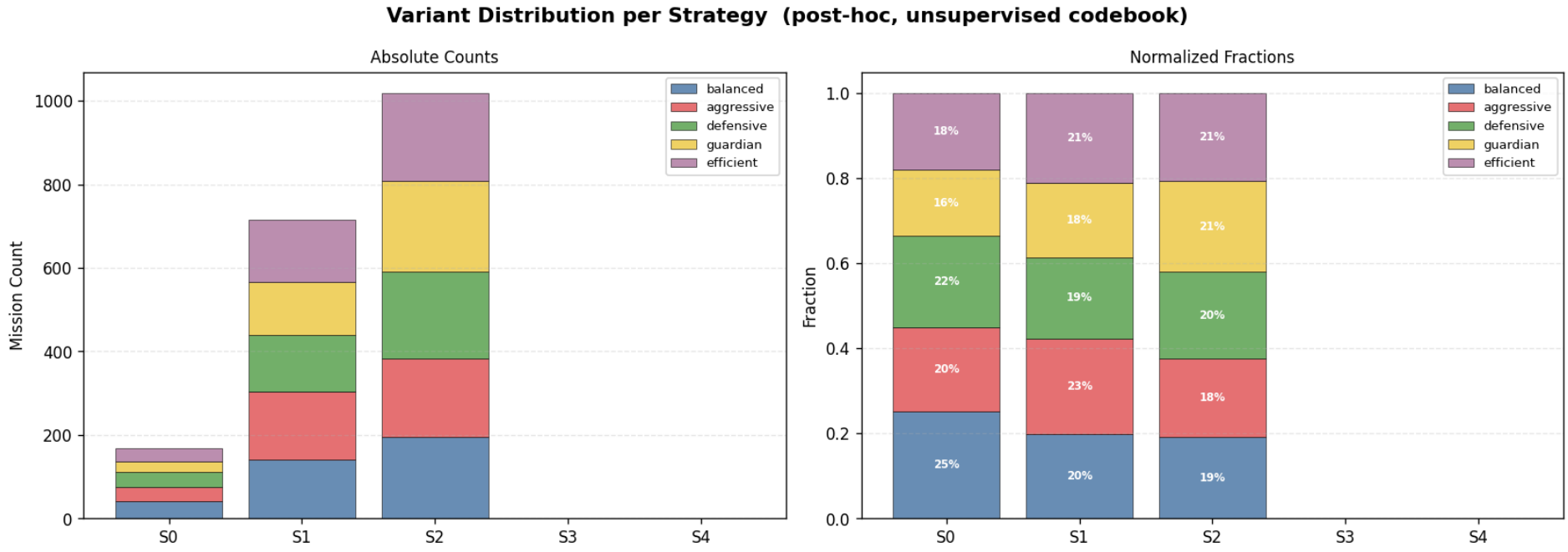
Model 2.2 Results – Strategy by Mission Variant Distribution

Codebook classifies turns - each mission gets one strategy label assigned by majority vote of turns

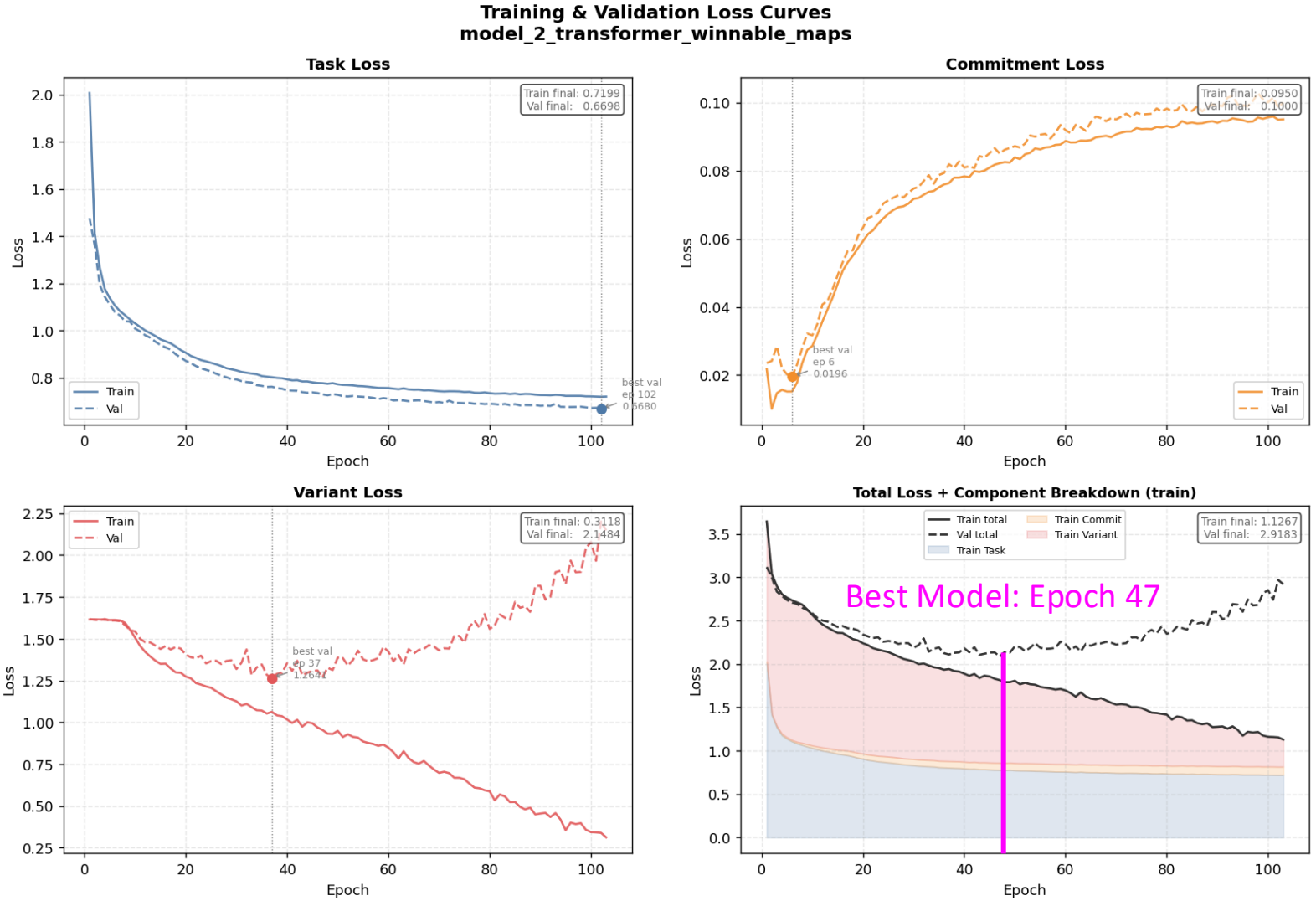
Near-uniform mixture of all five injected policy variants (16–25% each)

S0 contains significantly fewer missions than other strategies

S3 and S4 contain no missions



Model 2.3 – Convergence (Total Validation Loss)

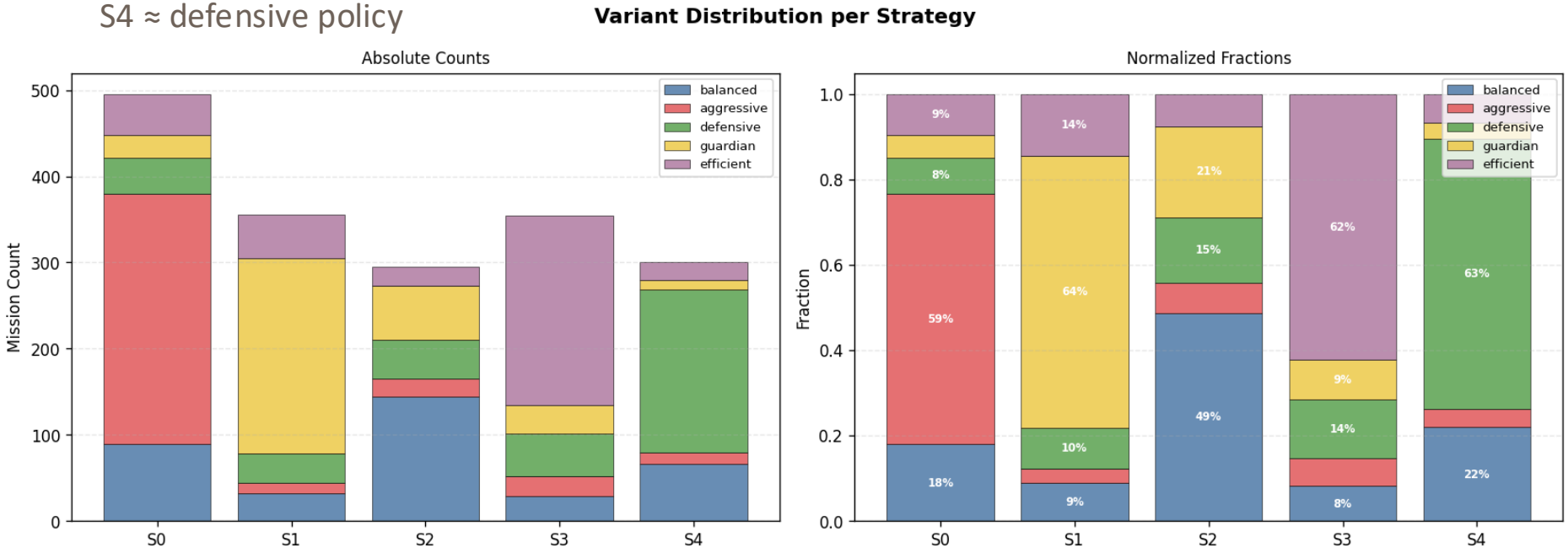


Model 2.3 Results – Strategy by Mission Variant Distribution

Codebook classifies turns - each mission gets one strategy label assigned by majority vote of turns

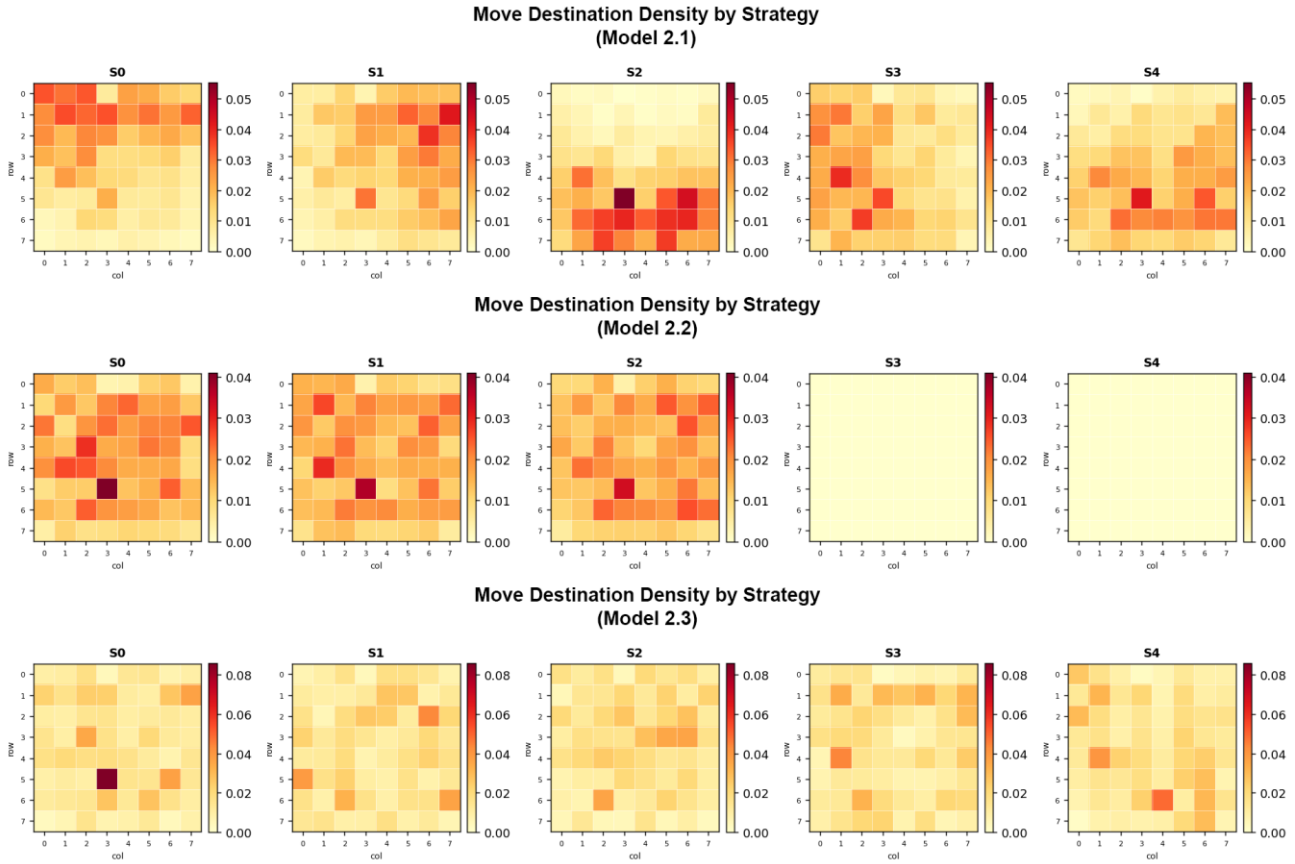
Each trained policy dominated a unique codebook strategy when decoder predicts Model 1 policy

S0 ≈ aggressive policy
S1 ≈ guardian policy
S2 ≈ balanced policy
S3 ≈ efficient policy
S4 ≈ defensive policy



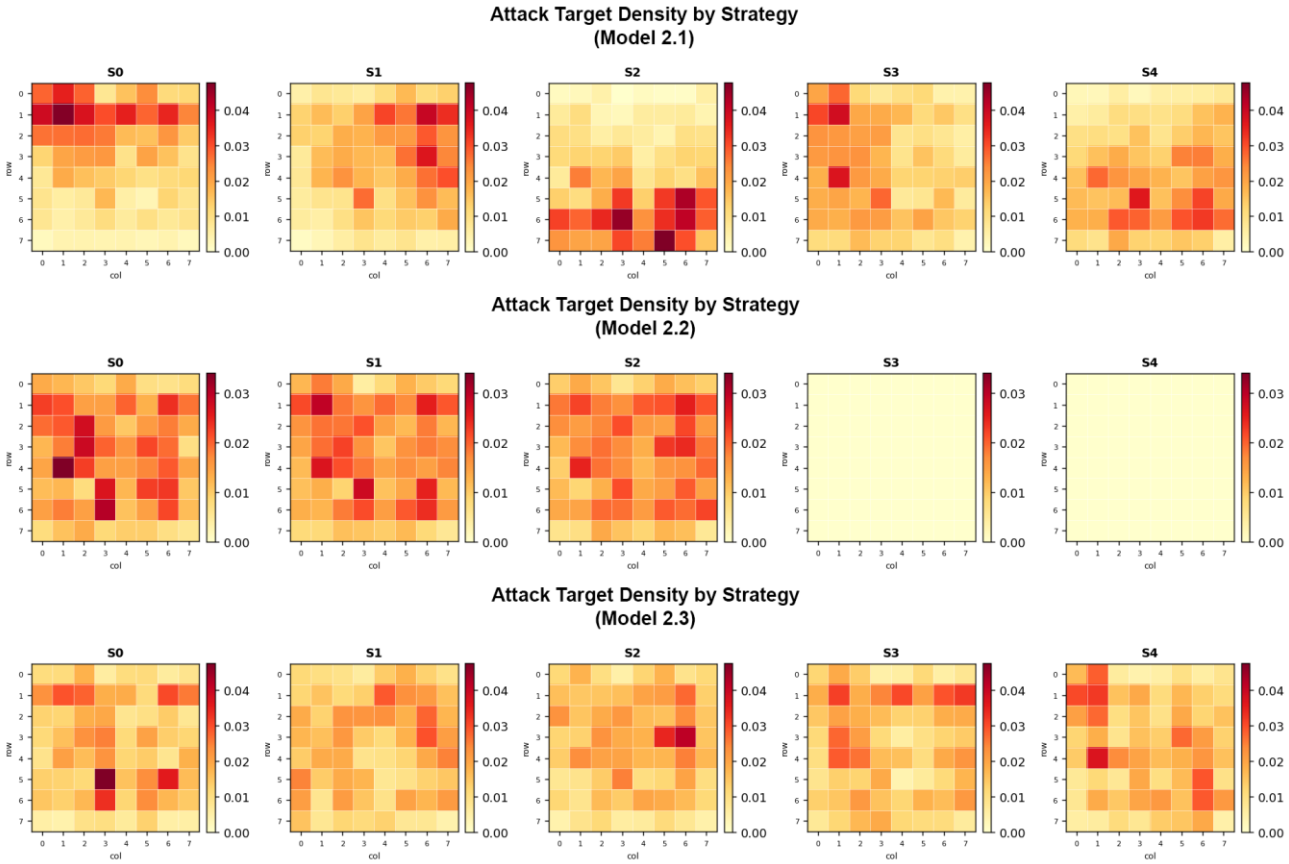
Model 2 Results – Move Destination Spatial Density

When task is dominated by move and attack spatial information, clustering follows



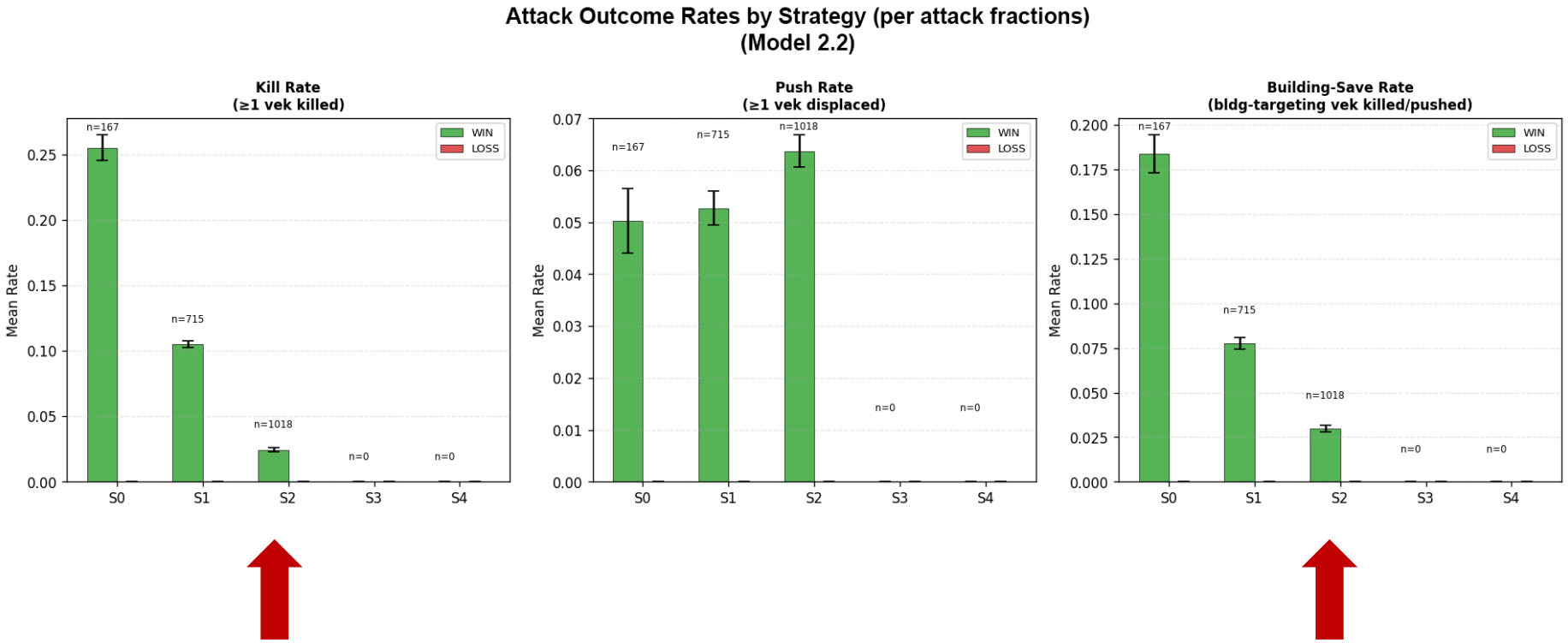
Model 2 Results – Attack Destination Spatial Density

When task is dominated by move and attack spatial information, clustering follows



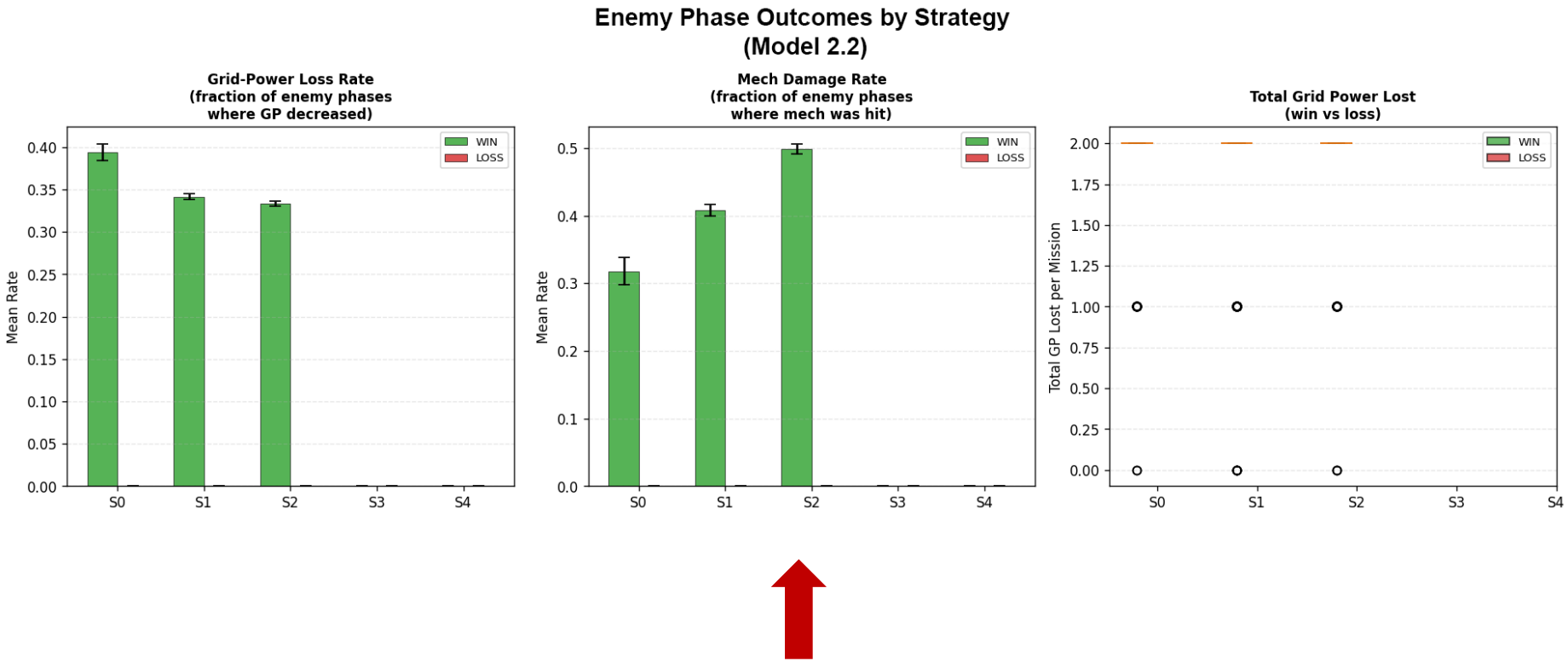
Model 2.3 Results – Strategy by Mission Variant Distribution

Kill Rate and Building Save Rate to strategy differentiation



Model 2.3 Results – Strategy by Mission Variant Distribution

Mech Damage contributes to strategy differentiation



Model 3

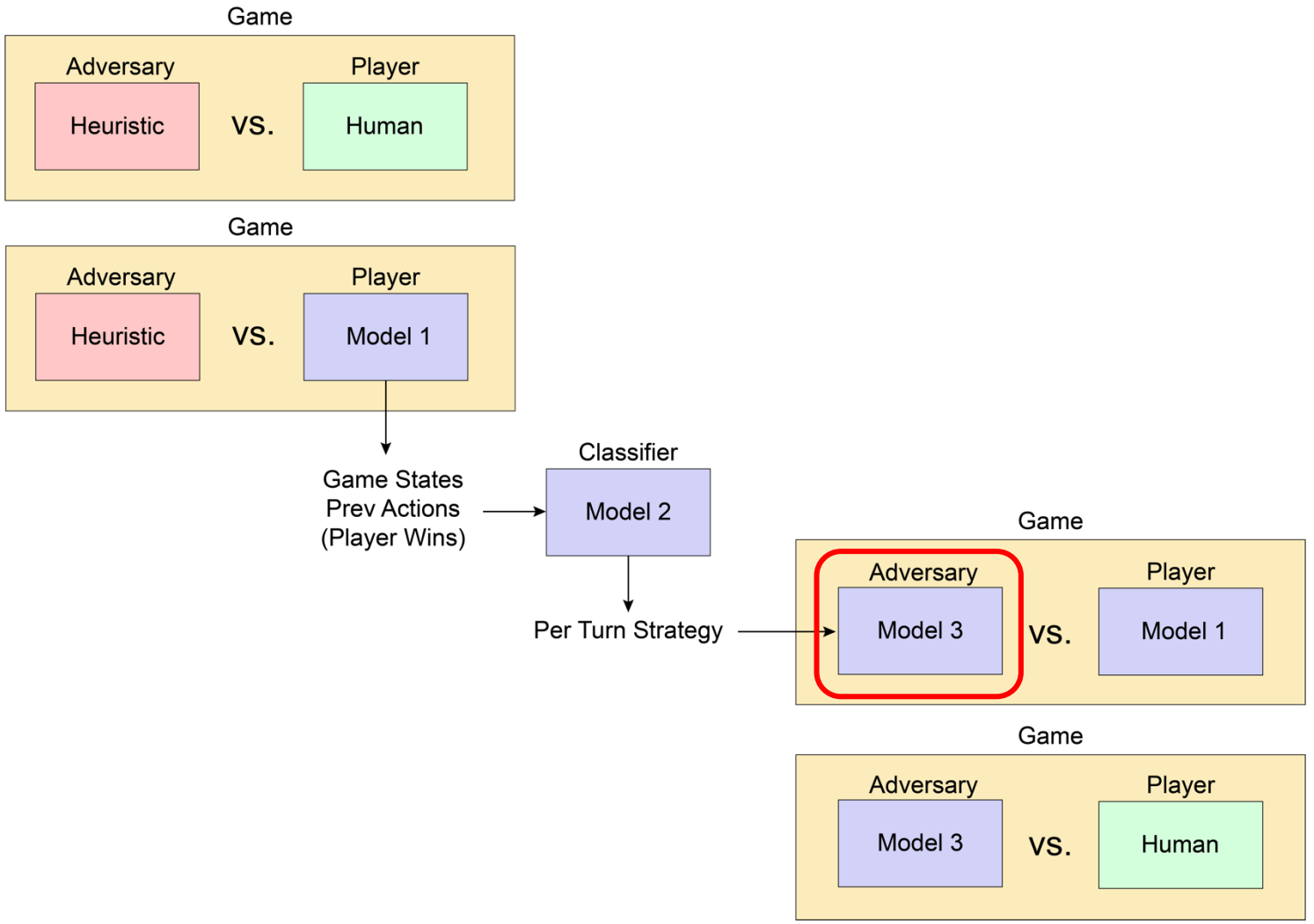
Adversary Agent

(Strategy Mitigator)

One multi-head PPO model by

Jessica Kinnevan

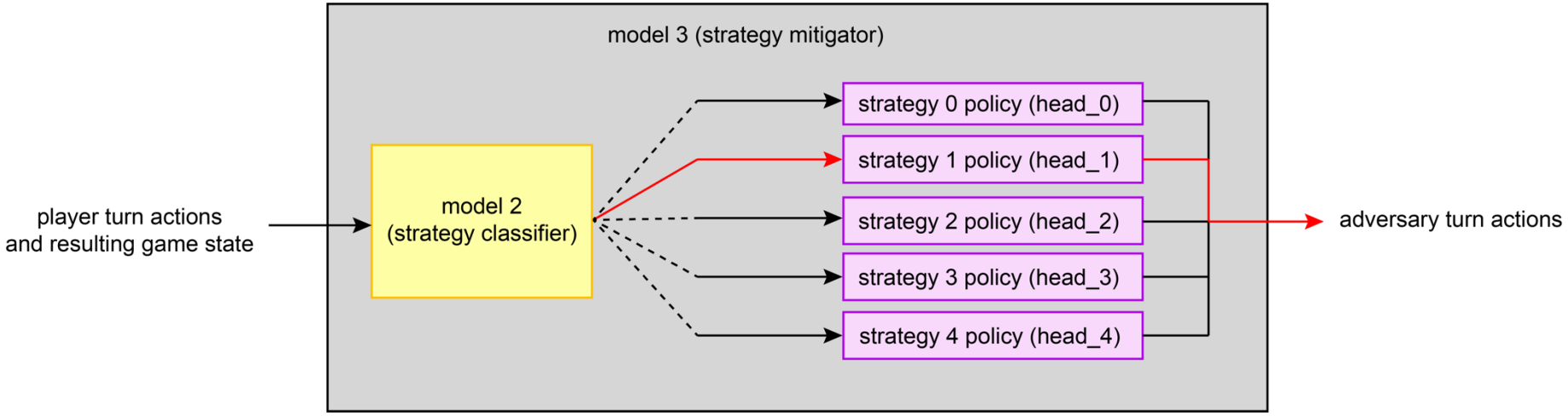
Model 3 – Adversary Model for Mitigating Strategies



Model 3 – Adversary Model PPO Structure

PPO Structure like Model 1 except:

- Doesn’t use a GRU layer
- Transformer already handles the "what strategy is the player using?" question at each turn
- Each head only needs to answer "given the current board state and that I'm countering strategy X, what's my best action right now?”
- Model 2 acts as router to route player turn by strategy to corresponding policy



Model 3 – Training

Effectively trained 3 of 5 available heads, due to only 3 strategy classes being predicted by Model 2

Transition split among active heads was roughly:

h0: ~27%

h1: ~43%

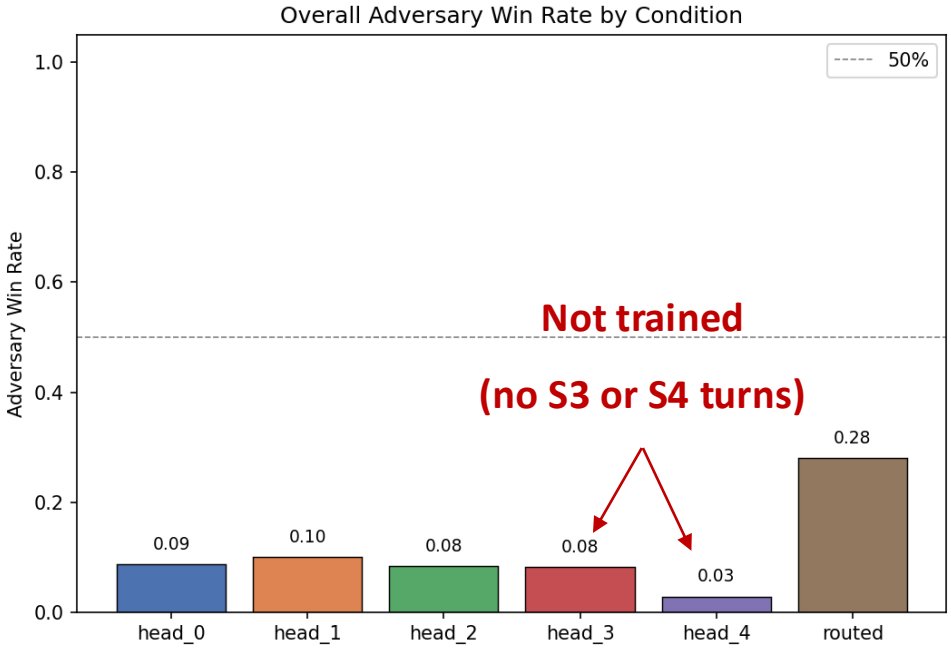
h2: ~30%

Setting	Model 1	Model 3
Total timesteps	3,000,000	4,000,000
Rollout steps	2,048	2,048
Minibatch size	256	256
Update epochs	4	4
Learning rate	$3x10^{-4}$ (linear decay)	$3x10^{-4}$ (linear decay)
Clip epsilon	0.2	0.2
Entropy coef	0.05	0.02
Patience	50	100
Hidden sizes	(512, 512) + GRU 128	(512, 512) + GRU 128

Model 3 – Results Against "Balanced" Policy

40 maps – 30 games per map – 1200 total observations

- At n=30, win rate estimate has a maximum standard error of ~0.091 (at p=0.5) steps/epoch
- Sufficient for regression/ANOVA across conditions
- ~5% difference in overall win rate between two conditions is statistically meaningful



head_0: expert on S0

head_1: expert on S1

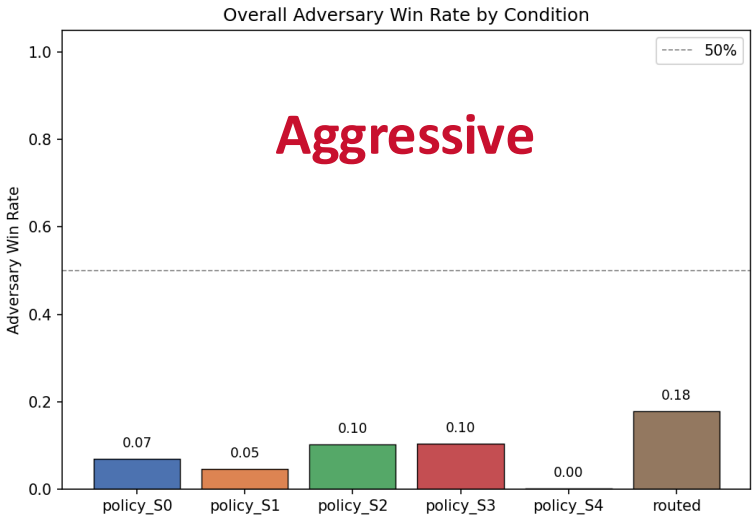
head_2: expert on S2

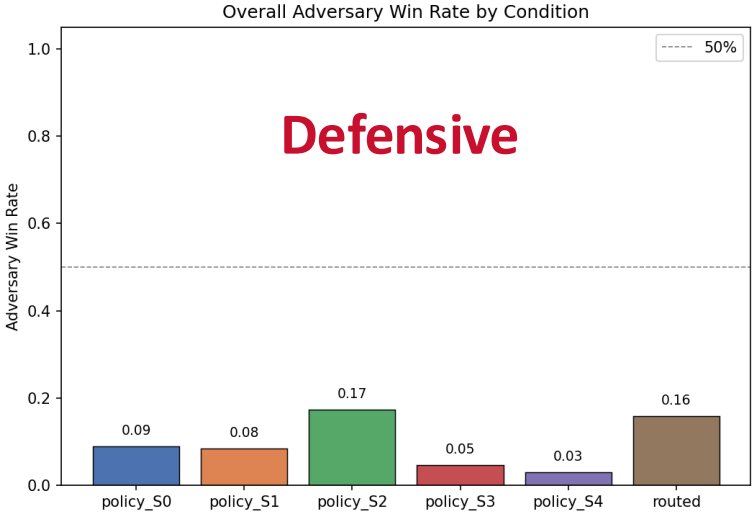
head_3: not trained (initialized)

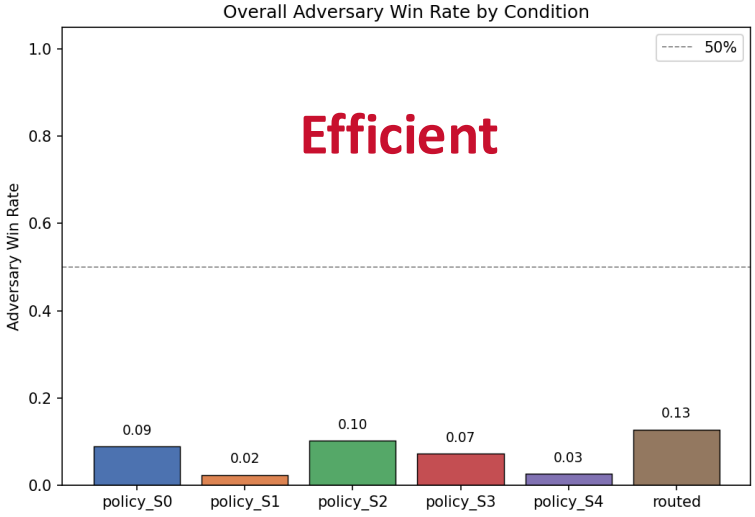
head_4: not trained (initialized)

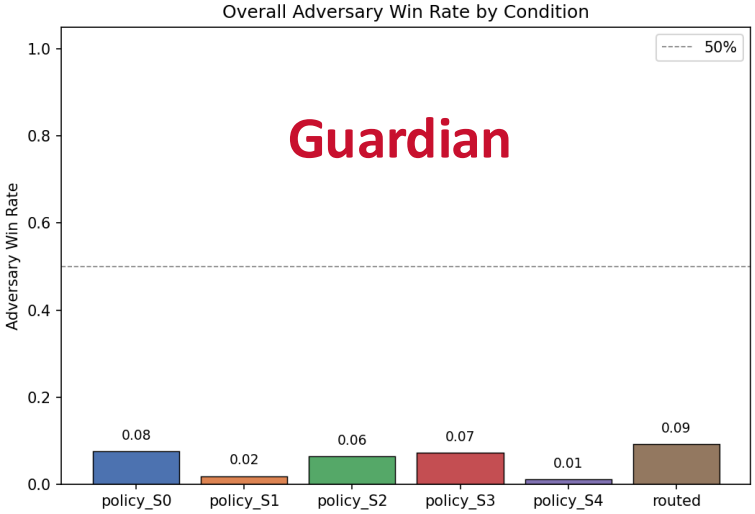
18% difference to next highest

Model 3 – Against Other Model 1 Polices (Model 3 trained on Balanced)



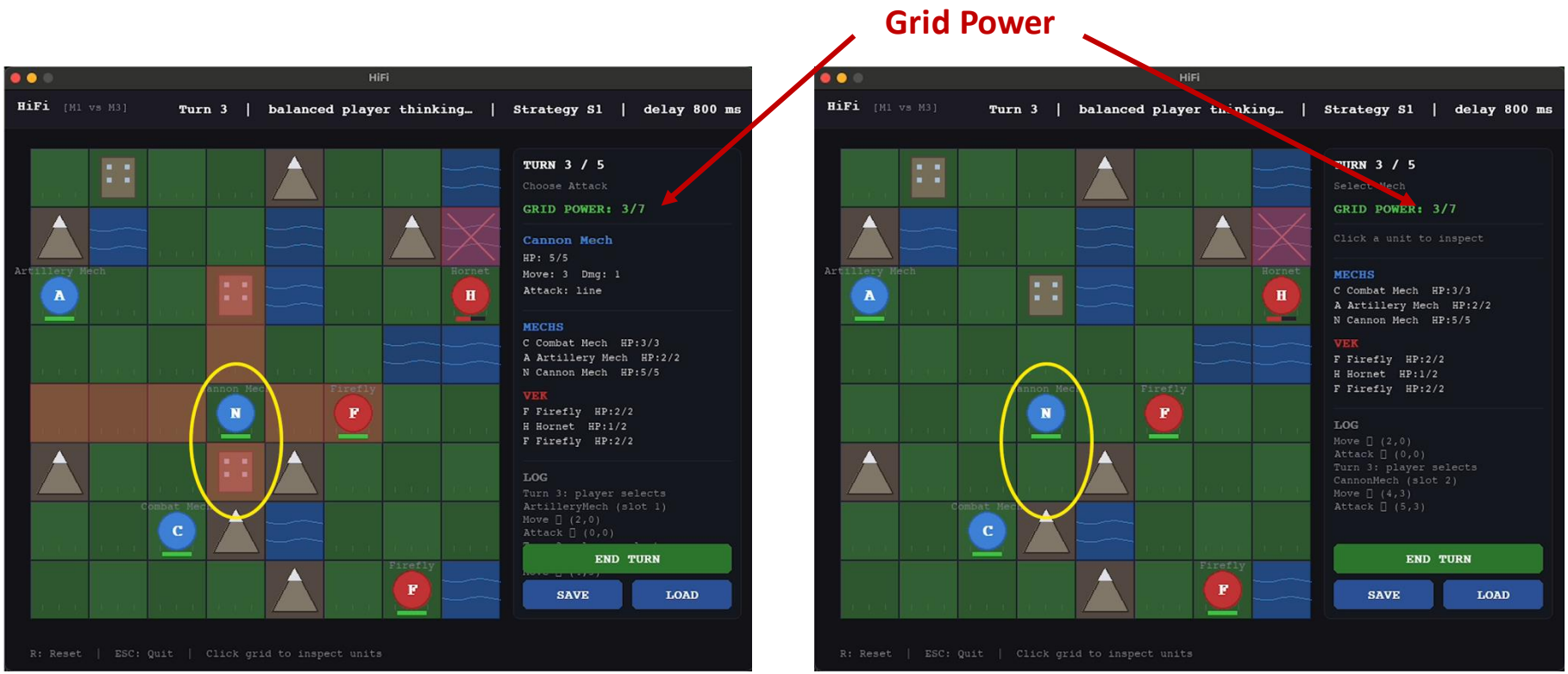






Future Work – Game Environment Changes

- Mech destroying building does not currently reduce grid power – fix this
- Prevent emergent strategy - mech destroying building to protect grid power



Future Work – Reward System Changes

- Add friendly fire penalty

Reward	When	Balanced	Aggressive	Defensive	Guardian	Efficient
Vek killed (player phase)	per kill	+0.3	+0.6	+0.15	+0.2	+0.3
Vek killed (enemy phase)	per kill	+0.3	+0.6	+0.15	+0.2	+0.3
Building destroyed (player phase)	per building	-0.2	-0.05	-0.15	-0.6	-0.2
Building destroyed (enemy phase)	per building	-0.2	-0.05	-0.15	-0.6	-0.2
Mech lost (friendly fire)	per mech	-0.3	-0.1	-0.8	-0.2	-0.2
Mech lost (enemy phase)	per mech	-0.3	-0.1	-0.8	-0.2	-0.2
Survived Enemy Phase	per turn	+0.1	+0.05	+0.2	+0.1	+0.02
All mechs acted	per turn	+0.15	+0.2	+0.1	+0.15	+0.3
Idle mech at end turn	per mech	-0.1	-0.15	-0.05	-0.1	-0.2
Win	terminal	+1.0	+1.0	+1.0	+1.0	+1.0
Early win bonus	per turn	+0.0	+0.0	+0.0	+0.0	+0.25
Lose	terminal	-1.0	-0.5	-1.5	-1.0	-1.0

Models 1, 2, and 3 - Key Findings (Jessica Kinnevan)

1. Model 2 - A Vector Quantized Variational Autoencoder (VQ-VAE) codebook clusters game data in a way that is indirectly influenced by the VQ-VAE decoder's predicted output.

2. Model 3 - A multi-head Proximal Policy Optimization (PPO) adversarial model generally performed better when training and inference was routed to heads using the VQ-VAE as the strategy classifier, selecting the head by strategy index (S0, S1, S2, S3, S4)

3. Game and Model 1 - changing the game environment has cascading effect Game environment must enable varieties of strategies, and Model 1 must exploit them for VQ-VAE (Model 2) to differentiate them.

Model 3

Adversary Agent

(Strategy Mitigator)

Comparison of 19 Model Variants

by Toral Chauhan

Contributions

Game

- Rule-faithful demo environment
- Dedicated testbed for adversarial models

Model 3

- Integrated 19 adversarial models into the game
- Evaluated and benchmarked performance in parallel

VBRS

- Introduced value-based reward shaping
- Improved learning by aligning rewards with strategic outcomes

Three-Model System Architecture

MODEL 1

Player Model

UNDERGRADUATE TEAM

- PPO + GAE
- 5 behavioral variants
- ~1,200 winning trajectories
- Fixed opponent for evaluation

MODEL 2

Strategy Classifier

GRADUATE TEAM

- VQ-VAE + Transformer encoder
- 5-code discrete codebook
- Real-time per-turn output
- Informs adversary strategy

MODEL 3

Adversary Framework

PRIMARY CONTRIBUTION — Toral Chauhan

- 19 adversarial agents
- Rule / Search / Learning
- 2,850 total episodes
- Novel VBRS adversary

The Core Problem

Limitations of Existing Agents

- **Fixed strategies** : rule-based agents cannot adapt mid-game.
- **Reward exploitation** : most agents win by finding loopholes, not genuine play.
- **No strategy awareness** : no real-time classification of opponent behavior exists.

Four Granular Problems

- What does "strategy" mean in a specific game context?
- What signals make one strategy distinguishable from another?
- What data is needed to train a strategy-aware adversarial model?
- How do we evaluate an agent that counters multiple strategies objectively?

Domains Impacted: Defense | Financial | Gaming | Education

The Core Problem and Approach

01 → **What is a strategy?**

5 PPO-trained behavioral variants operationalize "strategy" as a measurable policy.

02 → **How do we tell them apart?**

- 19 architectures spanning Rule-Based, Search & Learning families.
- Score variance separates exploitation from genuine adaptation.

03 → **What data do we train on?**

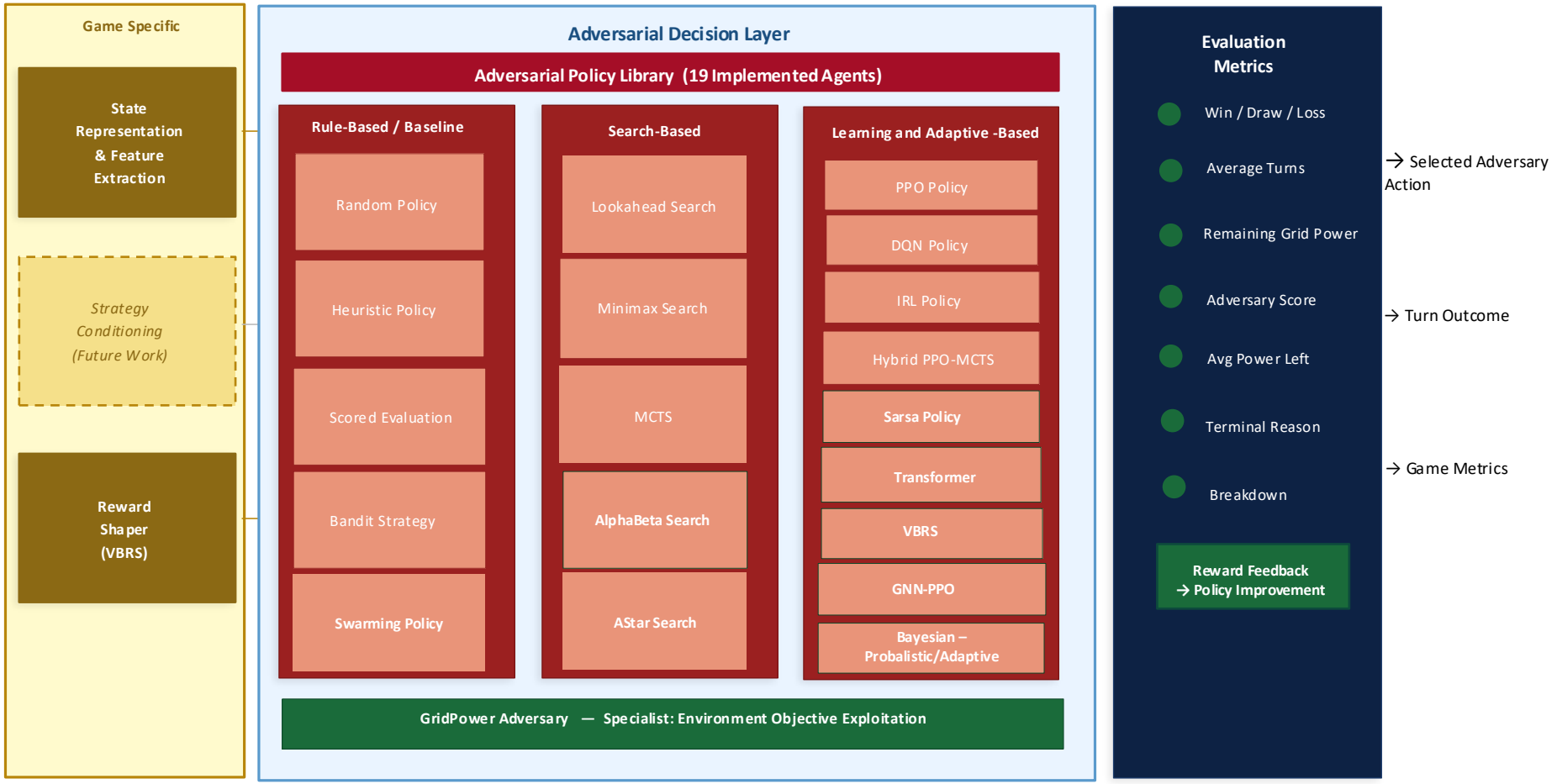
- 1,200 winning trajectories across 40 maps, split 80/10/10.
- Each architecture trained & evaluated on same fixed player.

04 → **How do we measure it?**

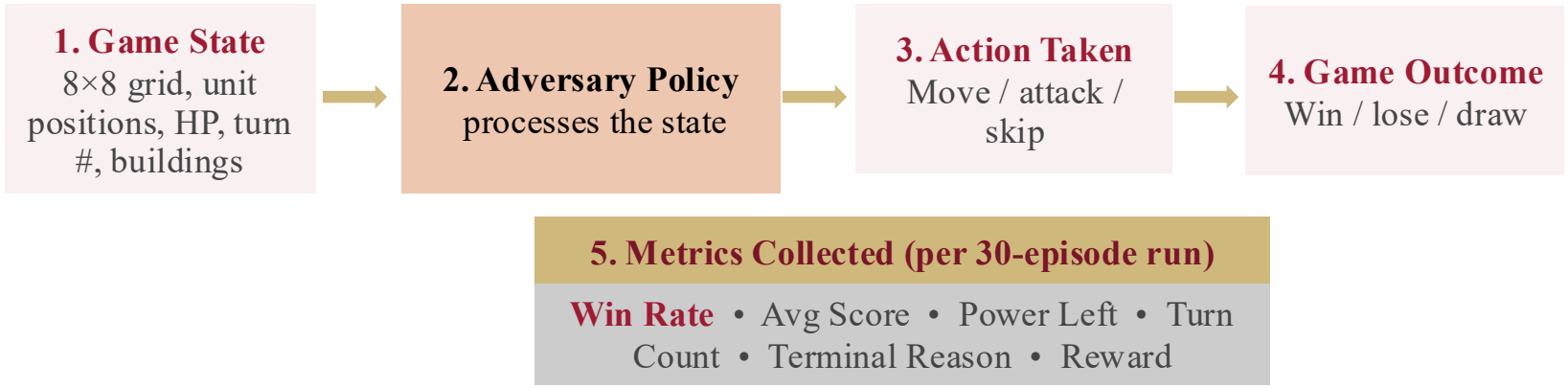
6-metric framework: Win Rate, Avg Score, Avg Power Left, Turn Count, Terminal Reason, Reward.

Model 3 — Multi-Adversary Benchmarking Framework for Tactical Game Environments

19 Implemented Adversaries | 5 Variants | 2,850 Episodes |



Model 3 — Execution Flow

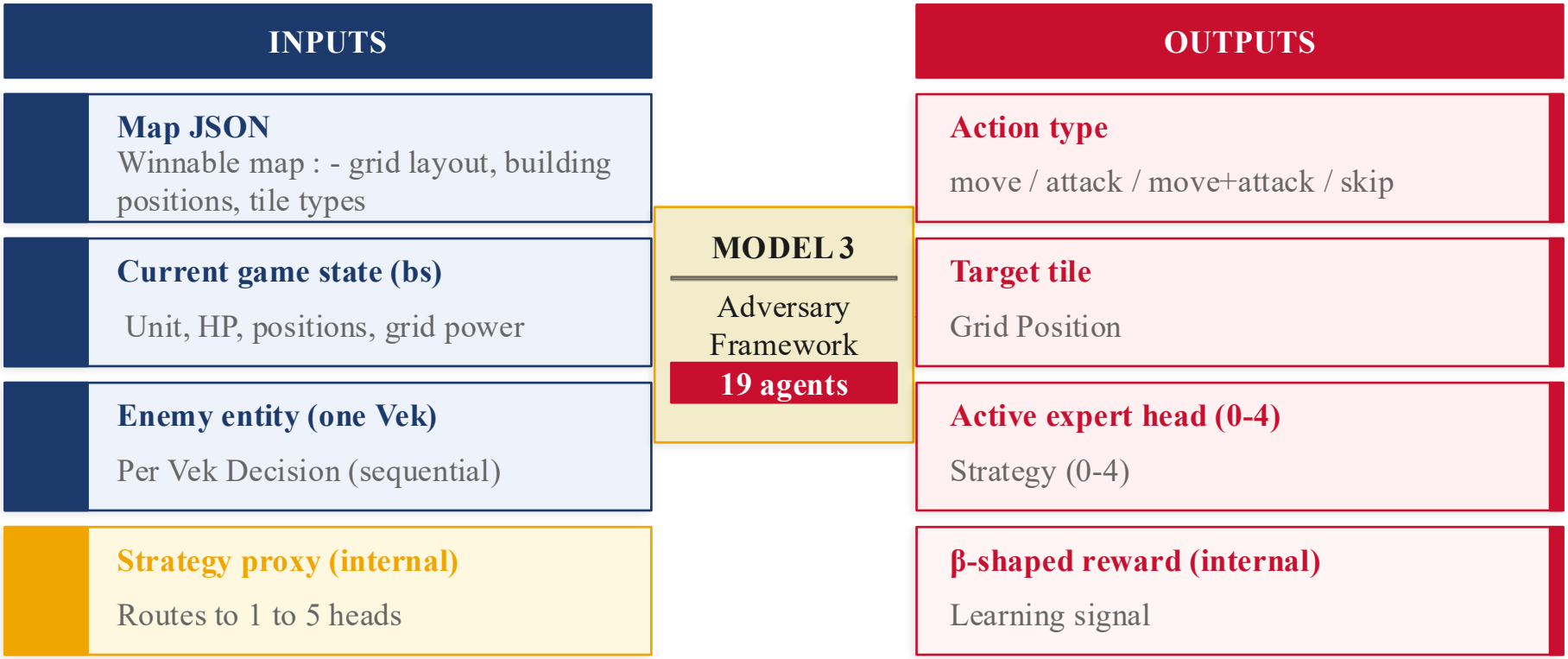


The ONLY variable changing across all 2,850 episodes is the adversary decision logic

Actually a controlled experimental system

Fixed player + fixed game + fixed maps = results are 100% attributable to adversary intelligence

Model 3 — Input and Output



Model 3 — Rule-Based & Baseline Agents (5)

Random Policy 11.3%

Random baseline. Sets the performance floor.

Heuristic Policy 12.7%

Fixed priority rules. Mimics Into the Breach enemy behavior.

GridPower Adversary ★ 85.3%

Targets buildings only. Exploits a training blind spot.

Swarming Policy 38%

Concentrate all Vek units on single highest-priority target simultaneously. Focuses all units on one target. Forces difficult player decisions.

Bandit Strategy 12.7%

UCB multi-armed bandit: $Q(a) + c\sqrt{(\ln N / n(a))}$. Incremental reward estimates without neural network. Lightweight but cannot reason about multi-step consequences.

Model 3 — Search-Based Agents (6)

Lookahead Search 8.0%

Simulates 1–2 steps ahead. Too shallow for complex decisions.

Minimax Search 7.3%

Maximizes adversary score, minimizes player score. Computationally expensive at scale.

MCTS 7.3%

Random rollouts produce noise in a 5-turn game. Cannot distinguish good early moves from bad.

AlphaBeta Search 13.3%

Pruned Minimax. Doubles effective search depth within the same compute budget.

A* Search 6.7%

Spatial heuristic navigates toward nearest building. Limited to spatial reasoning only.

Hybrid PPO-MCTS ★ 40%

PPO policy replaces random rollouts fixes MCTS noise problem. Highest-ranked search agent.

Model 3 — Learning-Based Agents (8) — Part 1

PPO Policy ~10%

Standard PPO adversarial baseline. Clipped surrogate with GAE advantage. Serves as the backbone for Hybrid PPO-MCTS and VBRS.

DQN Policy 39%

Deep Q-Network with experience replay. Off-policy learning is competitive in this environment.

IRL Policy 10.7%

Infers player reward, then trains PPO to counter it. Smart approach but limited by a small 1,800-mission dataset.

SARSA Policy 6.7%

Conservative on-policy learner. Lower than DQN confirms off-policy sample efficiency matters more here.

Model 3 — Learning-Based Agents (8) — Part 2

Transformer Reward Shaper 10.0%

Attends over multi-turn history for context-aware reward shaping. Efficient with resources but hasn't learned to target buildings yet.

GNN-PPO 0.7%

Graph Neural Network over the game grid. Lowest performer graph features don't capture short-horizon strategy.

Bayesian Probabilistic Agent ★ — 2nd Overall: 73% Win Rate (109/150)

Avg Win Rate:	73%	vs Aggressive:	53%
vs Balanced:	60%	vs Defensive:	83%
vs Efficient:	80%	vs Guardian:	90%

Updates beliefs via Bayes' rule: $P(\text{profile}|\text{action}) \propto P(\text{action}|\text{profile}) \times P(\text{profile})$. Switches targeting strategy based on dominant profile. 30-point variance across variants = genuine behavioral responsiveness.

VBRS Adversary

Value-Based Reward Shaping

VBRS — Motivation & Core Formula

Based on: Hou, Pan & Chen — "Enhancing Adversarial Policy Learning via Value-Based Reward Shaping" — Electronics 2026, 15, 463

The Problem:

Standard PPO in the 5-turn game receives reward ONLY at terminal states. Intermediate turns have zero immediate reward — the gradient signal is too sparse for the policy to learn which early actions set up a winning position.

$$r'(s,a) = (1 - \beta) \times r(s,a) + \beta \times V(s)$$

r'(s,a)

Shaped reward — what the actor trains on

r(s,a)

Raw environment reward (sparse, terminal only)

V(s)

Main Critic's long-term value estimate for state s

$\beta \in [0, 0.05]$

Adaptive mixing coefficient — grows as critic confidence increases

VBRS — Component Architecture

Main Critic $V(s;\omega)$

- Trained on raw environment rewards
- Updated via temporal difference + bootstrapped targets
- Provides V(s) shaping signal + drives β schedule
- High variance → low confidence → keeps β near 0

Auxiliary Critic $V'(s;\omega')$

- Trained on SHAPED reward r'(s,a)
- Provides advantage estimates for actor updates
- Decoupled from Main Critic prevents contamination
- Stabilises training as β shifts reward landscape

Actor $\pi(a|s)$

- Standard PPO policy over {attack, move, skip}
- Trained using Auxiliary Critic advantage
- PPO clipping maintains stability as β changes
- Same backbone as baseline PPO adversary

Adaptive β Controller

- $\beta_0 = 0$ → no shaping at initialization
- Grows toward $\beta_{max} = 0.05$ as critic variance drops
- Max change $\delta\beta = 0.002$ per update (prevents jumps)
- $\beta_{max} = 0.05$ from Hou et al. ablation study

VBRS

How it looks

Treated as: just one of 19 adversaries

- Compared directly to GridPower (85%) and Bayesian (73%) — unfair and misleading.
- Early-stage VBRS at 26% looks weak.

What it actually is

A continuously improving learning framework

- β starts at 0
- Grows as critic confidence rises.
- Its current 26% is a snapshot, not a ceiling.

The key distinction: 18 agents are STATIC. VBRS is DYNAMIC.

Sparse gradient problem solved

- Standard PPO gets reward only at terminal states.
- VBRS mixes in V(s) to create dense per-turn signal

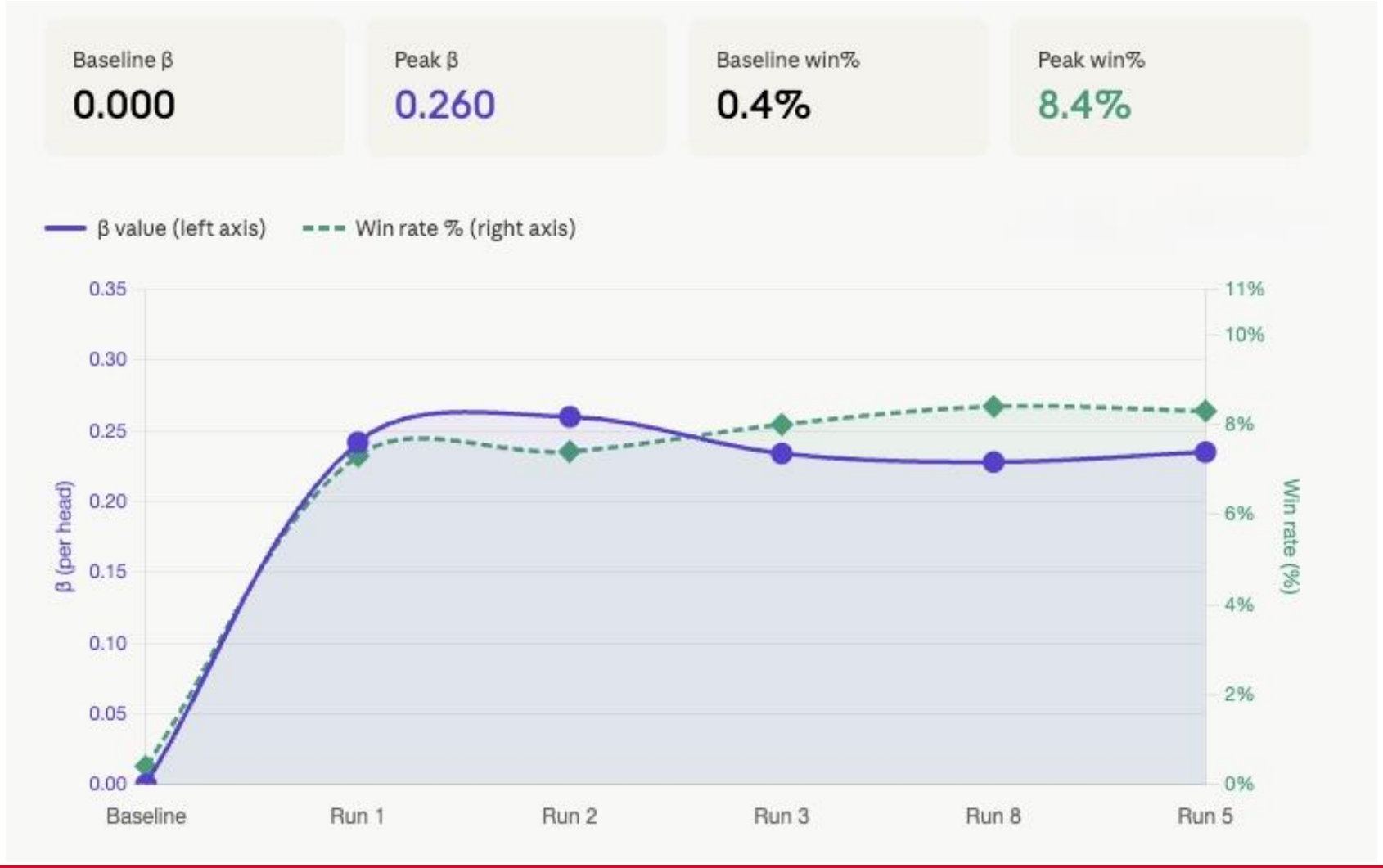
Dual-critic architecture

- Main critic trained on raw reward.
- Auxiliary critic trained on shaped reward.

VBRS — Hyperparameter Specifications

Parameter	Value	Rationale / Design Choice
β_0 (initial)	0	No shaping at init — critics must earn their influence
β_max	0.05	Upper bound from Hou et al. ablation (0.03 too weak, 0.10 over-biases)
$\delta\beta$_max	0.002	Max β change per update — prevents sudden instability
EMA rate (κ)	0.1	Controls smoothing of TD loss for confidence estimation
PPO clip (ε)	0.2	Standard clipping to limit policy update magnitude
Discount (γ)	0.96	Long-horizon discounting for 5-turn episodes
GAE λ	0.95	Bias-variance tradeoff for advantage estimation
Buffer size	2,048 steps	Episodes collected before each update pass
Minibatch	128	Samples per gradient update step
PPO epochs (K)	6	Passes over collected trajectory per update
LR (actor)	3e-4	Standard Adam learning rate
LR (main critic)	1e-3	Slightly higher — faster value function convergence
Network	5 × 128 MLP	ReLU activation; same for all components
Expert heads	5	One per player variant (future — pending Model 2 integration)

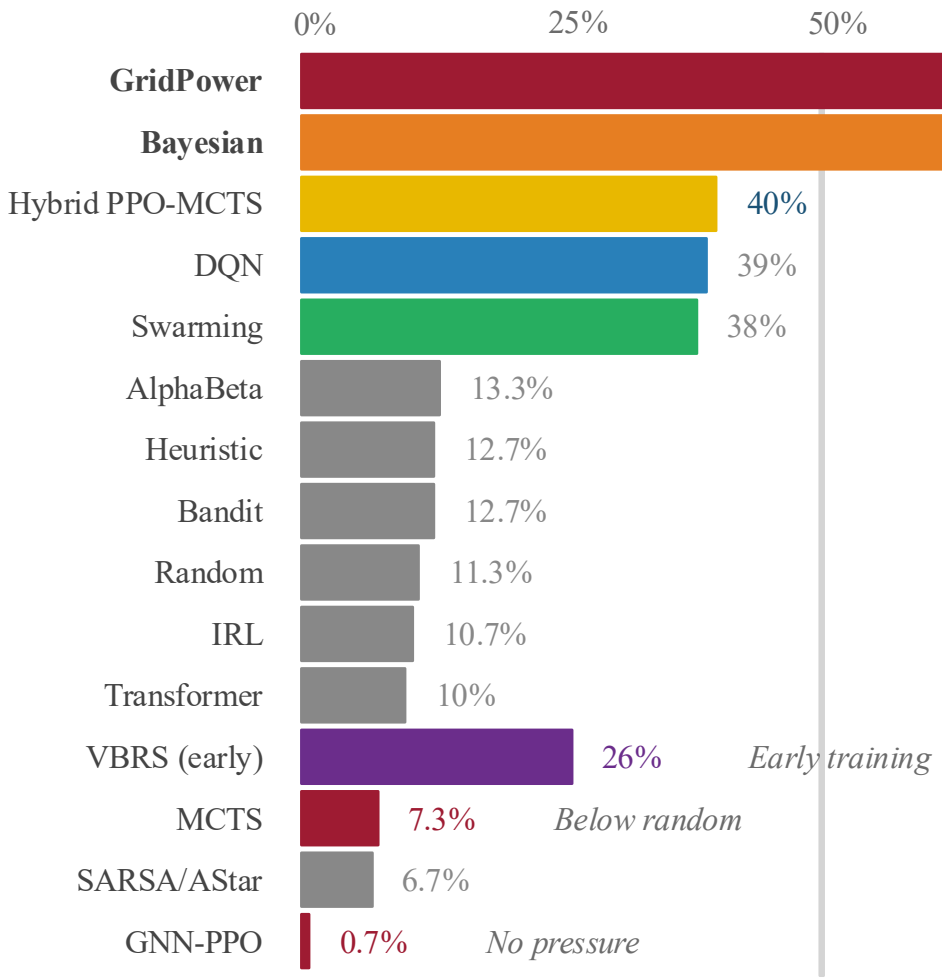
VBRS — β progression and win rate training runs



Experimental Results

2,850 Episodes · 19 Adversaries · 5 Variants · 30 Episodes · 6 Metrics

Overall Adversary Win Rate Rankings (150 Episodes Each)



Top Performers

1	**GridPower** 128/150 (85%)	
2	**Bayesian** 109/150 (73%)	
3	**HybridPPOMCTS** 60/150 (40%)	
4	**DQN** 58/150 (39%)	
5	**Swarming** 57/150 (38%)	

GridPower & Bayesian far exceed all others

Adversary Win Rate (%) — Adversary × Player Variant



Key Findings

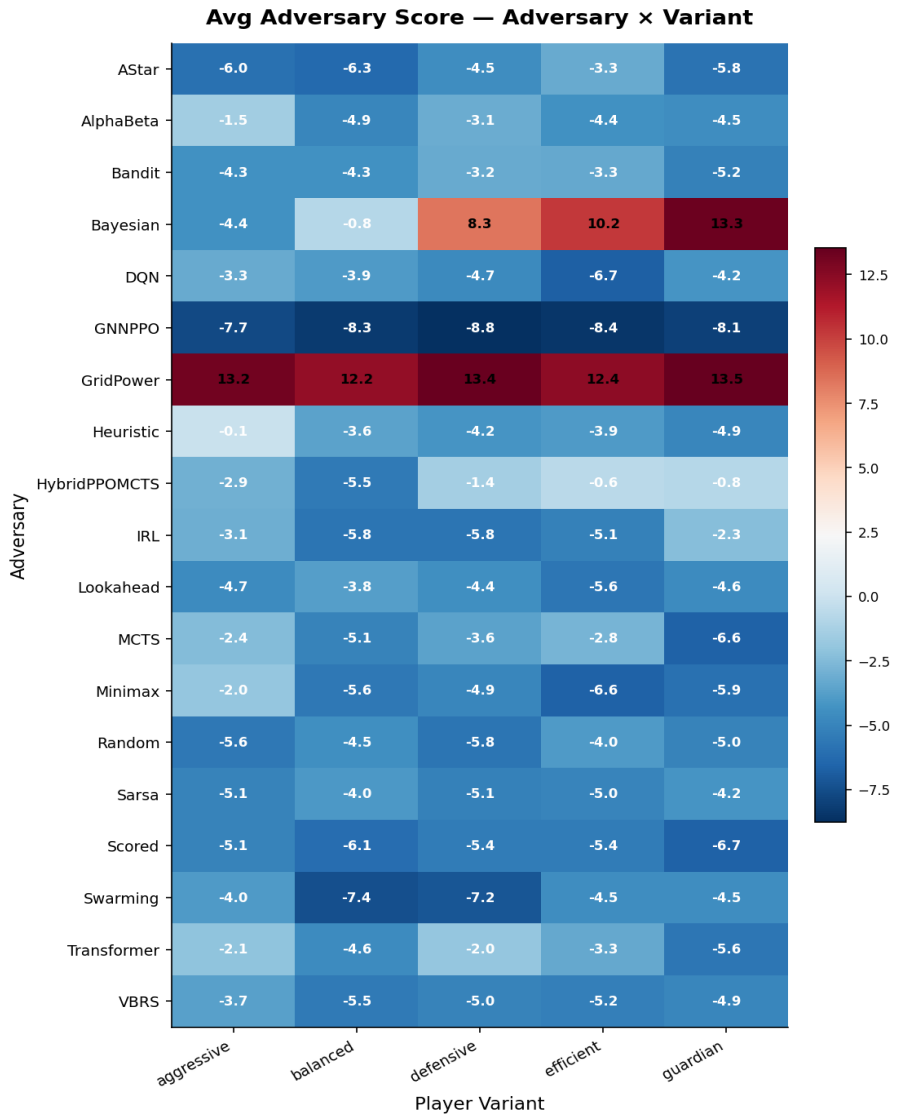
GridPower dominates
80–90% win rate across ALL variants — consistent environment exploitation

Bayesian adapts well
53–83% win rate, strongest in defensive (83%) and efficient (80%) variants

MCTS consistently weakest
Only 17–23% win rate — poor fit for this turn-based environment structure

VBRS early stage
20–40% win rate — needs more training episodes to stabilize performance

Avg Adversary Score — Strategic Effectiveness by Variant



Score Interpretation

Score > 10
Highly effective strategic adversary

Score 3–10
Moderate strategic effectiveness

Score 0–3
Low but positive contribution

Score < 0
Net negative — player outclasses adversary

Notable Scores

GridPower: 12.9–14.3 across all variants

Bayesian: 6.8–13.5 (defensive peak)

VBRS: -2.3 to +2.9 (still training)

Average Power Left — Building Damage Effectiveness

	Aggr	Balanced	Defensive	Efficient	Guardian
AStar	2.2	2.0	2.0	1.8	2.0
AlphaBeta	1.5	1.6	1.9	2.1	1.7
Bandit	2.0	1.7	1.8	1.6	2.0
Bayesian★	0.8	0.6	**0.2**	**0.4**	**0.2**
DQN	1.7	1.6	1.9	1.9	1.6
GNN-PPO	2.3	2.4	2.4	2.5	2.6
GridPower★	**0.2**	**0.3**	**0.3**	**0.3**	**0.1**
Heuristic	1.6	1.9	1.9	1.8	1.9
HybridMCTS	1.6	1.4	1.2	1.3	1.3
IRL	1.8	2.1	2.2	2.2	1.7
Lookahead	2.0	1.7	2.1	1.9	2.2
MCTS	1.8	2.2	2.3	2.3	2.7
Minimax	1.6	2.0	1.9	2.2	2.1
Random	1.6	1.6	1.9	1.6	1.6
SARSA	1.9	1.7	1.9	2.0	2.0
Scored	2.0	1.9	2.1	1.9	2.1
Swarming	1.8	2.0	2.1	1.7	1.8
Transformer	2.8	2.7	2.9	2.9	2.8
VBRS	1.7	1.9	1.8	1.7	1.9

Avg Power Left (max 7)

- < 0.5 (near total)
- 0.5–1.0
- 1.0–1.5
- 1.5–2.0
- 2.0+ (low damage)

GridPower: 0.1–0.3 (destroys nearly all buildings every episode) | Transformer: 2.7–2.9 (almost no building damage despite 10% win rate)

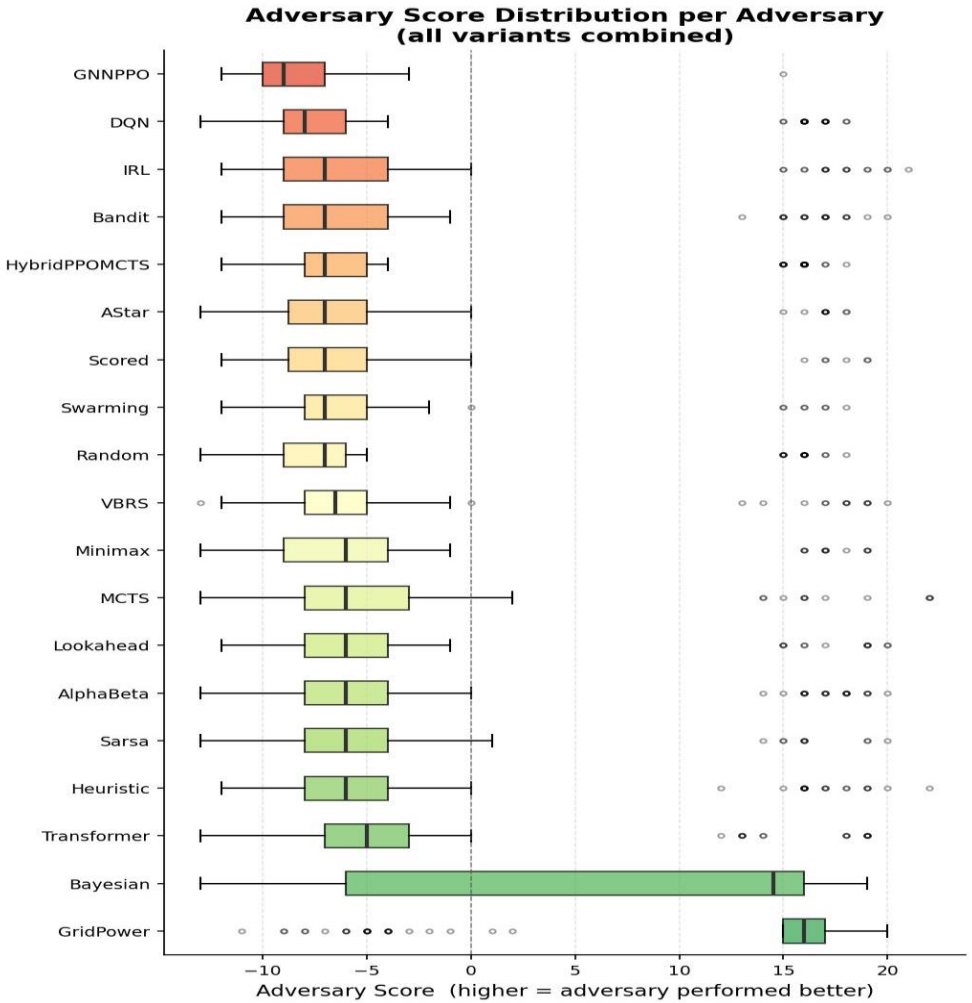
Distribution Analysis — Adversary Score

GridPower & Bayesian hardest adversaries

VBRS wins majority — most scores negative

GNNPPO, DQN, IRL toughest RL opponents

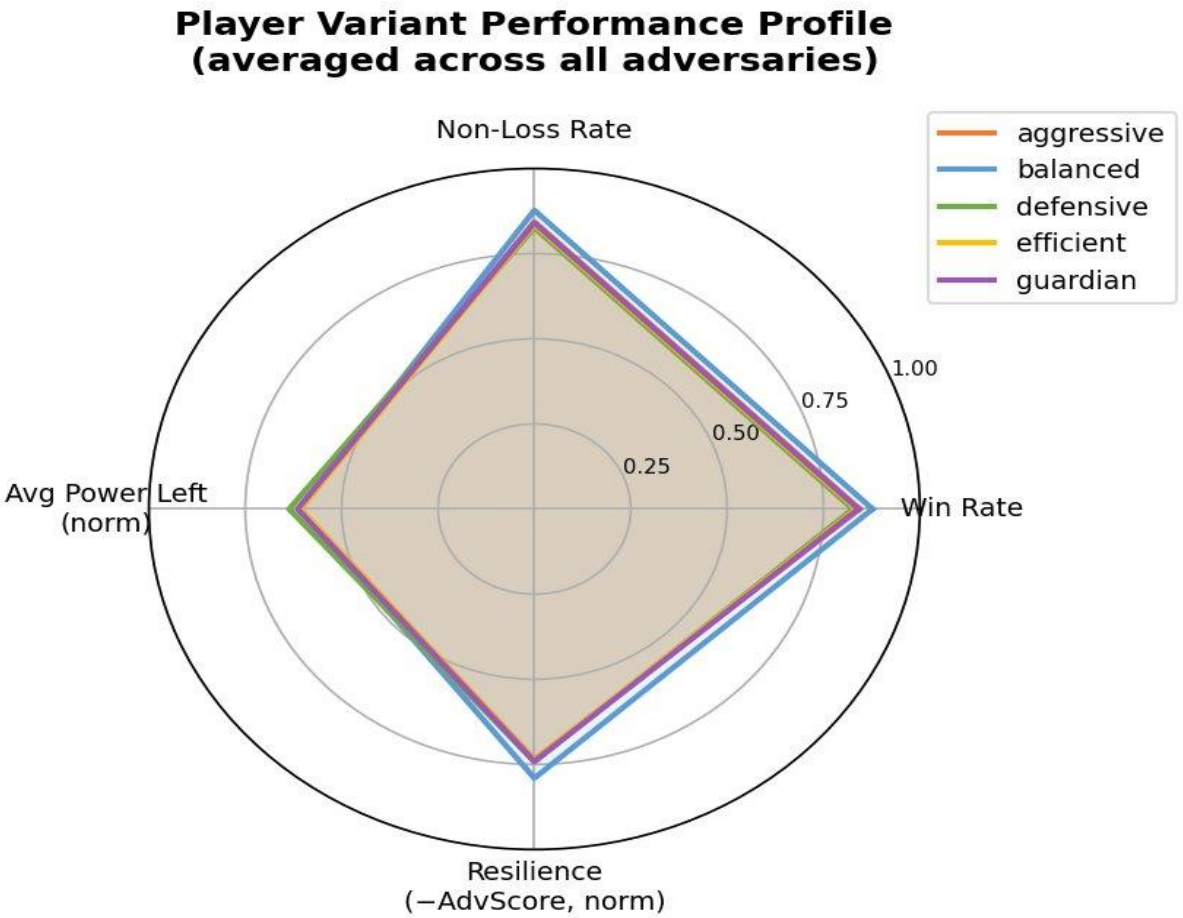
Bayesian: widest IQR = high variance



4

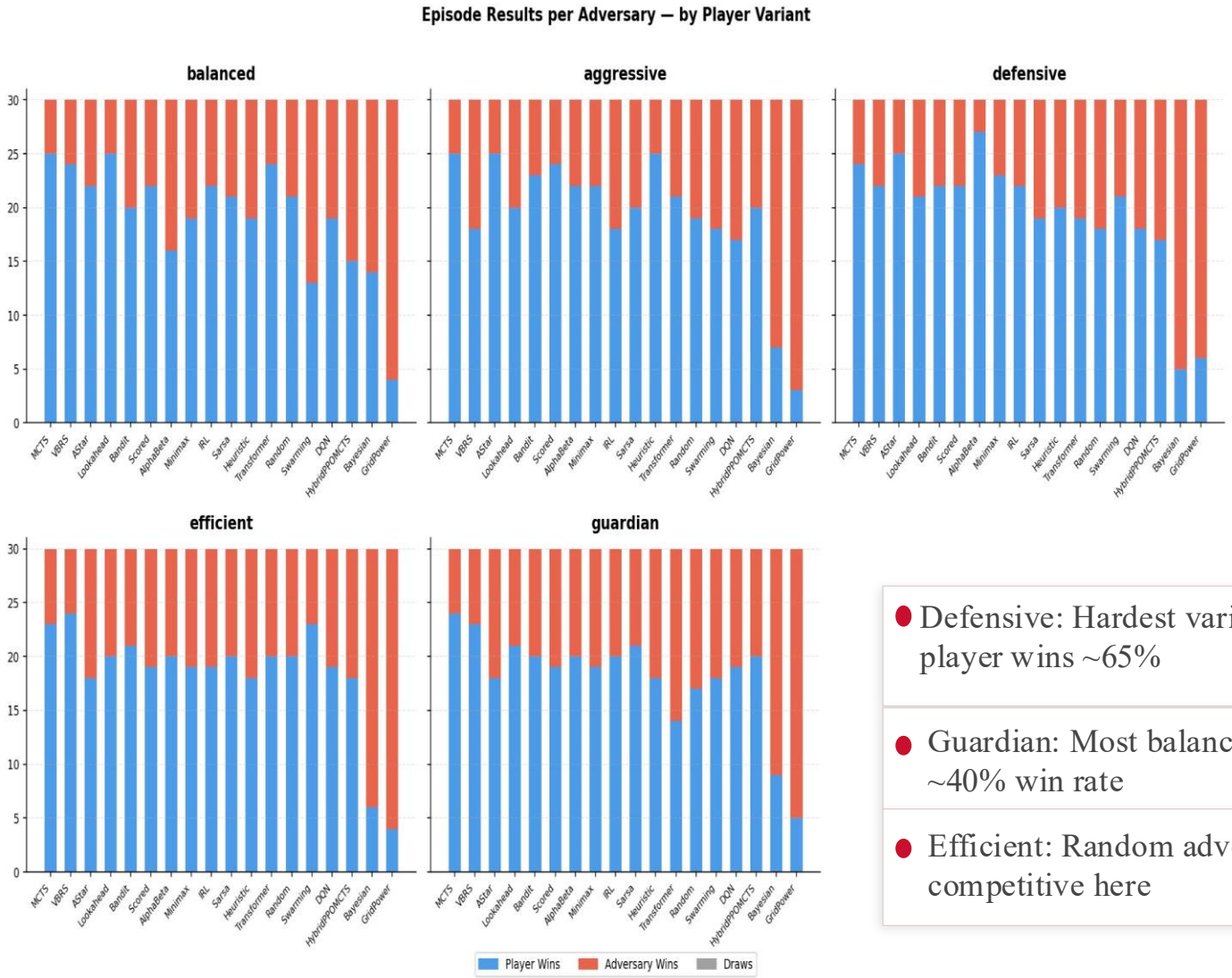
Player Variant Performance Profile

- aggressive
- balanced
- defensive
- guardian
- efficient



Averaged across all adversaries · Normalized metrics

Win Breakdown by Player Variant



- Defensive: Hardest variant for adversaries — player wins ~65%
- Guardian: Most balanced — adversaries reach ~40% win rate
- Efficient: Random adversary surprisingly competitive here

Deep Dive — How GridPower & Bayesian Win (Mechanisms Only)

GridPower Adversary • 85.3% Win Rate	
Primary Target	Building tiles (grid power) — not mechs
Attack Logic	Targets mechs adjacent to buildings for splash damage
Movement	Always moves toward nearest surviving building
Why It Wins	Exploits reward misalignment PPO trained to fight, not defend buildings
Implication	Fix: retrain Model 1 with elevated building_loss_penalty

Bayesian Adversary • 73% Win Rate
Tracks 3 player profiles: aggressive, defensive, passive
Bayes' rule on mech centroid movement each turn
Switches attack strategy based on dominant inferred profile
PPO has consistent policy; Bayesian quickly identifies & exploits it
Strength: expands to 5-profile likelihood model with Model 2 integration

Research Note: GridPower dominance signals environment imbalance, not pure strategic superiority. Bayesian is the only adversary that truly adapts — not rule-based exploitation.

VBRS Adversary (Early-Stage Results & Trajectory)

Metric	Balanced	Aggressive	Defensive	Guardian	Efficient
Adversary Wins	6	12	8	7	6
Win Rate	20%	**40%**	27%	23%	20%
Avg Score	-2.3	+2.9	+0.6	-0.9	-2.1
Avg Turns	4.70	4.55	4.63	4.75	4.80
Avg Power Left	1.95	1.72	1.88	1.90	1.85

β is still low. VBRS improves with experience, unlike rule-based adversaries. Value-shaping signal beginning to engage.

Best: Aggressive variant (40%) — Positive avg score (+2.9) confirms learning signal is working against action-oriented player.

Projected: 300–500 more episodes — β schedule engages → building targeting emerges → 5 strategy-conditioned expert heads → full Model 2 integration.

Discussion & Key Findings

What the results tell us about adversarial agents

Win Rate Alone Is Not Enough — The Central Argument

GridPower wins 85% — Bayesian wins 73%. Is GridPower a better adversary?

GridPower says: YES

- 85% > 73% — numeric advantage
- Consistent 80–87% across ALL 5 variants
- Score +12.2 to +13.5 (highly effective)
- Rapid building destruction in 1–3 turns

Evaluation framework says: NO

- 7-point variance = structural exploit only
- Fails immediately if player retrained
- Does not adapt to player behavior at all
- 'Skill' not 'Intelligence' (Chollet 2019)

Answer: NO. GridPower has task-specific skill. Bayesian has genuine intelligence. Bayesian's advantage persists after retraining; GridPower's does not.

What is Strategy Shift Rate (SSR)

SSR = Turns where Strategy Label Changed ÷ Total Adversary Actions

Measures Disruption

SSR captures how often the adversary forces the player to switch strategy — going beyond win/loss as a metric

Built on Model 2

Each incoming turn is classified by the VQ-VAE (Model 2) into a strategy label. A label change = a shift

Novel Metric

No prior adversary evaluation framework measures strategy disruption per turn. SSR = 0 for all non-Model 2 adversaries

Actionable Signal

High SSR adversaries can be used to reward-shape Model 3: reward the adversary for causing strategy shifts

How SSR is calculated in Code

1. **Classify Every Turn**

   Model 2 (VQ-VAE) processes each game turn and assigns a strategy label (0–4). This runs at inference time during evaluation.

2. **Detect Label Changes**

   After each episode, compare consecutive strategy labels. If label[i] ≠ label[i-1], a strategy shift has occurred at that turn.

3. **Count & Divide**

   SSR = total shift count ÷ total turns in the episode. This gives a normalized 0–1 score per episode, then averaged across 30 episodes.

```
# Strategy Shift Rate Implementation

strategy_history = []
shift_count = 0

for turn in episode:
  label = model2.classify(turn)
  strategy_history.append(label)

  if len(strategy_history) > 1:
    prev = strategy_history[-2]
    curr = strategy_history[-1]
    if curr != prev:
      shift_count += 1

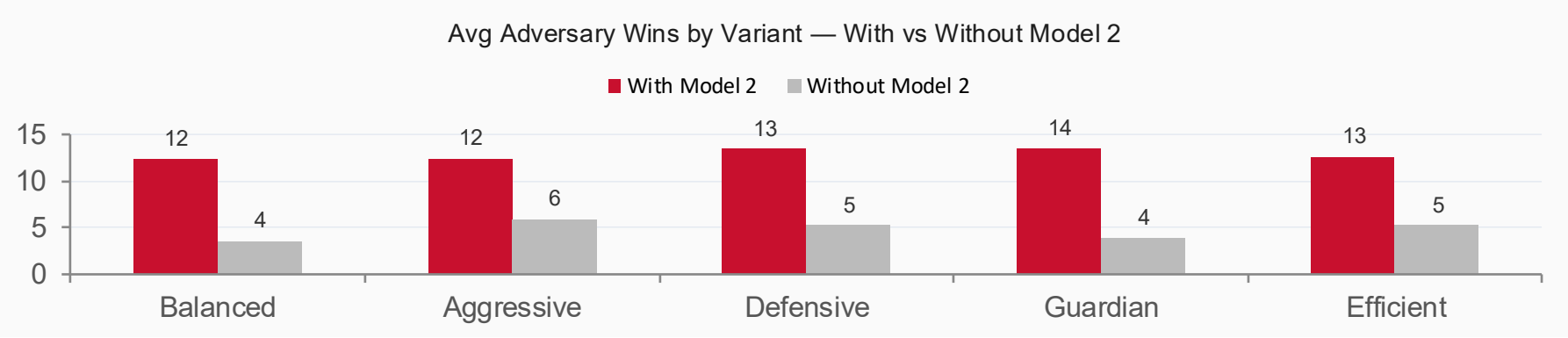
total_turns = len(episode)
SSR = shift_count / total_turns
```

Model 2 Integration : The Core Result

15.0
Avg Adversary Wins
with Model 2

4.7
Avg Adversary Wins
without Model 2

3.2×
Win Rate Improvement
Model 2 vs baseline



Implementing SSR in Model 3 Adversary training

Step 1: Classify Turn (Model 2)

Model 2 runs at each turn during adversary training. Strategy label is produced for every player action.

Step 2: Detect Strategy Shift

Compare current turn label with previous. If different → shift detected. Increment shift counter.

Step 3: Compute Shift Reward

SSR bonus added to adversary reward:
reward += shift_weight × (1 if shift else 0)
shift_weight tuned at 0.1–0.5

Step 4: Route Counter-Policy

Model 3 selects the counter-policy matching the current detected strategy. SSR reward reinforces successful disruption.

SSR as a reward signal transforms Model 3 from a win-maximizer into a strategy-disruption agent — a novel adversary training objective.

Key Findings

F1 **Bimodal Performance Gap**
GridPower (79%) and Bayesian (57%) sit 33+ pts above all others.

F2 **Exploitation ≠ Adaptation**
GridPower wins via a game flaw (targets buildings). High wins, zero SSR.

F3 **Model 2 Boosts Every Adversary 3.2×**
Avg wins jump from 4.7 → 15.0 when Model 2 is integrated. Strategy-awareness is the single biggest performance lever.

F4 **SSR = 0 Without Model 2**
All baseline adversaries (PPO, DQN, MCTS, Minimax, A*) show SSR = 0.0. Strategy-blind agents never force the player to adapt.

F5 **Planning Fails in 5-Turn Environments**
MCTS, Minimax, and A* perform at or below Random (≤2.8 avg wins). Short horizon eliminates look-ahead advantage entirely.

F6 **Building Damage = Key Differentiator**
Only GridPower and Bayesian reduce avg power left below 1.0. All others leave the player with comfortable grid margins.

F7 **VBRS Architecture Ceiling Not Reached**
VBRS results reflect early low-β stage. Designed to improve continuously current results are a floor, not a ceiling.

References

Chollet, F. (2019). On the measure of intelligence. arXiv:1911.01547.

Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (GELUs). arXiv:1606.08415. https://arxiv.org/abs/1606.08415

Hou, B.; Pan, G.; Chen, Y. Enhancing Adversarial Policy Learning via Value-Based Reward Shaping. Electronics 2026, 15, 463. https://doi.org/10.3390/electronics15020463

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS), 6379–6390. Link

Puigdomènech Badia, A., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., & Blundell, C. (2020). Agent57: Outperforming the Atari human benchmark. arXiv:2003.13350

Razavi, A., van den Oord, A., & Vinyals, O. (2019). Generating diverse high-fidelity images with VQ-VAE-2. Advances in Neural Information Processing Systems, 32. https://arxiv.org/abs/1906.00446

References

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., & Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. International Conference on Learning Representations. https://arxiv.org/abs/1506.02438

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv:1707.06347.

van den Oord, A., Vinyals, O., & Kavukcuoglu, K. (2017). Neural discrete representation learning. Advances in Neural Information Processing Systems, 30. https://arxiv.org/abs/1711.00937

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30. https://arxiv.org/abs/1706.03762

Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; and Wu, Y. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS). Link

Thank You