Strategy Classification in Adversarial Game Modeling

by

Jessica Kinnevan

A creative component submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Artificial Intelligence

Academic Plan Committee:
Simanta Mitra, Major Professor

The student author, whose presentation of the scholarship herein was approved by the academic plan committee, is solely responsible for the content of this creative component. The Graduate College will ensure this creative component is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University
Ames, Iowa
2026

Copyright © Jessica Kinnevan, 2026. All rights reserved.

DEDICATION

To my husband, for his encouragement and support, and to my kids, for their patience.

TABLE OF CONTENTS

Page

LIST OF FIGURES v

LIST OF TABLES vi

NOMENCLATURE vii

ACKNOWLEDGMENTS viii

ABSTRACT ix

CHAPTER 1. INTRODUCTION 1

CHAPTER 2. RELATED WORK 6
- Policy Gradient Methods in Deep Reinforcement Learning 6
- Multi-Agent Reinforcement Learning 7
- Discrete Latent Representation Learning 8
- Transformer Architectures for Sequential Encoding 9
- Adversarial Policy Learning and Strategic Intelligence in Games 10

CHAPTER 3. METHODOLOGY 11
- Game Environment 12
  - Objective 12
  - Game Board Generation 13
  - Building Placement 13
  - Unit Spawning 14
  - Unit Abilities 14
  - Modifications to the Game Environment 15
- Gameboard Map Curation Procedure 18
- Model 1 - Player Agent 19
  - Behavioral Variants 19
- Data Collection Procedure 21
- Model 2 - VQ-VAE Strategy Classifier 23
  - Model Architecture 23
  - Transformer Encoder 24
  - Turn-Level Vector Quantization 24
  - Decoder Skip Connection 25
  - Decoder Heads 26
  - Loss Calculation 26
  - Models 2.1 and 2.3 - Action Reconstruction Task Loss 28
  - Model 2.2 - Outcome Prediction Task Loss 28
  - Model 2.3 - Additional Variant Prediction Loss 29
  - Input Features 30
- Model 3 – Multi-Head PPO Adversary Agent 32

Overview 32
Episode Structure and Action Decomposition 33
Strategy Classification and Experience Routing 33
Network Architecture 34
Observation Space (2,372 features) 34
Reward Function 35
Training Configuration 35

CHAPTER 4. RESULTS 37
Results by Model 2 Variant 37
Model 2.1 – Spatial Strategy Discovery via Action Reconstruction 37
Model 2.2 – Outcome-Defined Strategy Discovery via Tactical Prediction 39
Model 2.3 – Supervised Strategy Alignment via Variant Prediction Loss 42
Model 3 – Player Strategy Mitigation with Policy Routing 43
Win Rate 44
Secondary Metrics 47
Routing Distribution 52
Generalizing Over Other Policies 52

CHAPTER 5. DISCUSSION 55
Model 2 as a Strategy-Specific Mitigation Engine for Model 3 55
Feature Prediction Fidelity and Codebook Adherence 56
Secondary Metric Interpretation and Head Utilization 57

CHAPTER 6. FUTURE WORK 59

REFERENCES 60

APPENDIX A. 61

LIST OF FIGURES

Page

Figure 3-1. The role of Model 1, Model 2, and Model 3 in the approach for strategy classification and mitigation. 12

Figure 3-2. Gameboard map_001 of the easy set of 20 maps. Mech are in blue, Vek are in red. 17

Figure 3-3. The interface used for manual selection of "winnable" gameboard maps. 18

Figure 3-4. The map_018 that was found to be unwinnable by all policies trained by Model 1.. 23

Figure 3-5. Turn-level embedding of tokens for quantization and mission-level (game-level) strategy assignment by majority vote of turn strategy. 25

Figure 3-6. The three variants of the VQ-VAE (Model 2) decoder configurations. 27

Figure 3-7. Model 3 with policy routing for each turn using Model 2. 36

Figure 4-1. The policy variant distribution per strategy across Models 2.1, 2.2, and 2.3..... 38

Figure 4-2. Gameboard attack target density by strategy across Models 2.1, 2.2, and 2.3..... 39

Figure 4-3. Gameboard move destination density by strategy across Models 2.1, 2.2, and 2.3... 40

Figure 4-4. Attack outcomes by strategy per attack fractions for Model 2.2..... 41

Figure 4-5. Enemy phase outcomes by strategy for Model 2.2 41

Figure 4-6. Overall adversary win rate by Model 3 policy head. 45

Figure 4-7. Win rate per map and condition by Model 3 policy head. 46

Figure 4-8. Grid Power at episode (game) end by Model 3 policy head. 47

Figure 4-9. Mean grid power over turns by Model 3 policy head. 48

Figure 4-10. Mean buildings destroyed per turn by the adversary by Model 3 policy head. 49

Figure 4-11. Total Mech HP lost by player per episode (game) by Model 3 policy head. 49

Figure 4-12. Mean number of Vek killed per turn by the player by Model 3 policy head. 50

Figure 4-13. Model 2.2 routing distribution in fraction of turns per head per map. 51

Figure 4-14. Overall adversary (Model 3) win rate by Model 3 policy head when playing against other player (Model 1) policies. 53

LIST OF TABLES

Page

Table 3-1: Unit movement profiles 15

Table 3-2: Unit attack profiles 15

Table 3-3. Unit damage and hit point profiles 15

Table 3-4. Reward components for each player agent policy (induced strategy). 20

Table 3-5. Observation vector used in Model 1. 21

Table 3-6. Gameboard maps that could not be won by policy for Model 2.1 and 2.3 dataset. 22

Table 3-7. Gameboard maps that could not be won by policy for Model 2.2 dataset. 22

Table 3-8. Summary input features detailed in Table 3-5 through Table 3-14. 30

Table 3-9. Token ID feature – shape (52,) 30

Table 3-10. Previous action features – shape (52,11) 30

Table 3-11. Grid Power feature – shape (52,) 31

Table 3-12. Unit table features – shape (52, 9, 13) 31

Table 3-13. Tile sensor features – shape (52, 17, 8, 8) 32

Table 3-14. Masks and intents features 32

Table 3-15. Model 3 reward components. 35

Table A-6-1. Model 3 adversary evaluation results by condition and map. 61

NOMENCLATURE

Term	Definition of the term
PPO	Proximal Policy Optimization
VQ-VAE	Vector Quantized Variational Autoencoder
SEM	Standard Error of the Mean

ACKNOWLEDGMENTS

I would like to thank my graduate major professor, Simanta Mitra, my teaching assistant and mentor, Haniyeh Fekrmandi, my project partner, Toral Chauhan, and Corbin Graham of Lockheed Martin Advanced Technology Labs for their guidance and support throughout the course of this research. In addition, I would also like to thank my friends, colleagues, the department faculty and staff for making my time at Iowa State University a wonderful and growing experience.

ABSTRACT

This work presents an unsupervised strategy classification pipeline for adaptive adversarial game AI, addressing the challenge of inferring player strategies in real time without labeled behavioral data. Existing approaches optimize for winning, treating difficulty as a scalar quantity that fails to capture the richness of human strategic diversity. To bridge this gap, we adapt the Vector Quantized Variational Autoencoder (VQ-VAE) with a transformer encoder and task-specific decoders to map sequences of game states and actions to discrete latent codes, enabling label-free strategy discovery from raw gameplay. The system is trained and evaluated on a custom turn-based tactical puzzle game played on an 8×8 grid, using a corpus generated by five behaviorally distinct policy variants: balanced, aggressive, defensive, guardian, and efficient.

We compare three decoder configurations: a reconstructive decoder that recovers unit actions and positions, an outcome-predictive decoder that forecasts tactical consequences such as enemy neutralization and building protection, and a label-supervised multi-task decoder that augments reconstruction with an auxiliary policy-variant classification loss. Both unsupervised models converge on strategy clusters whose policy-variant distributions are near-uniform, indicating that the learned representations reflect emergent behavioral structure rather than the designer's labeling scheme. The reconstructive model discovers five spatially differentiated clusters corresponding to distinct board-region activity patterns, while the outcome-predictive model produces three clusters organized around combat outcome profiles and temporal engagement rhythm. In contrast, the multi-task model with policy-variant classification recovers the imposed policy taxonomy, establishing the decoder objective as the primary driver of

codebook organization. These findings validate unsupervised representation learning as a principled foundation for strategy-aware adversarial agent training.

CHAPTER 1. INTRODUCTION

Effective artificial intelligence opponents in games require more than raw competence. They must adapt to the specific behaviors of individual players. Existing approaches to game AI have largely optimized for task-specific skill and winning instead of the kind of flexible, generalizable intelligence needed to adapt to individual player behavior (Chollet, 2019). These approaches treat difficulty as a scalar quantity, failing to capture the richness of human strategic diversity. A player who excels at aggressive, forward-pressing tactics but struggles with defensive positioning needs a fundamentally different challenge than one who favors cautious, resource-efficient strategies. Building adversarial agents capable of identifying and exploiting individual strategic tendencies requires a principled approach to real-time player modeling.

The central bottleneck in adaptive adversarial game AI is strategy classification: inferring the approach a player is currently employing from game state observations alone, without access to ground-truth behavioral labels. In offline settings, labeled datasets may be constructed and used to train supervised classifiers, but in deployed interactive systems this is impractical. Labels require either manual annotation, which is costly and highly subjective, or access to the true generating policy, which is unavailable during live play. What is needed is an unsupervised method capable of discovering meaningful strategy representations directly from raw gameplay data, one that operates in real time and generalizes across diverse player behaviors.

Prior work in game AI has achieved remarkable results through reinforcement learning, with approaches such as Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Multi-Agent Actor-Critic (Lowe et al., 2017) demonstrating strong adversarial and cooperative performance across a range of environments. The effectiveness of PPO in cooperative multi-agent settings has been further documented by Yu et al. (2022), while Agent57 (Puigdomènech

Badia et al., 2020) has demonstrated superhuman performance across the full Atari benchmark suite. More recent work on adversarial policy learning, such as value-based reward shaping (Hou et al., 2026), has extended these capabilities toward more strategically responsive opponents. However, these methods do not explicitly model the opponent's strategy in an unsupervised, label-free way. They optimize for winning, not for understanding how the opponent plays, and therefore cannot serve as a direct foundation for strategy-aware adaptive behavior.

This work addresses that gap through a representation learning approach to unsupervised strategy discovery. We present a Vector Quantized Variational Autoencoder (VQ-VAE) (van den Oord et al., 2017; Razavi et al., 2019) that encodes sequences of game states and actions into discrete latent codes, then assigns game-level strategy labels through majority vote over a learned codebook. The encoder is implemented as a transformer (Vaswani et al., 2017) with GELU activations (Hendrycks & Gimpel, 2016), enabling it to capture the temporal and spatial structure of gameplay across turns. The system is trained and evaluated on a custom turn-based tactical puzzle game inspired by Into the Breach, played on an 8×8 grid over five turns, with a training corpus of winning games generated by five behaviorally distinct policy variants: balanced, aggressive, defensive, guardian, and efficient. The model is without access to these policy labels during the training of two versions of the VQ-VAE model, while a third model did have access to labels for comparative purposes.

Post-training analysis of the first label-free model whose decoder was trained to reconstruct the action leading to the current game state, specifically what action was performed by what player unit and where, reveals that the model's codebook converges on five spatially differentiated strategy clusters that do not reproduce the injected policy-variant labels. Instead, the discovered strategies reflect genuine spatial-behavioral structure in the data: an upper-board

high-activity style, a right-edge flanking style, a compressed lower-center cluster distinguished by reduced mobility and mech spread, a wide-spreading left-flank style, and a center-anchored generalist style. Each discovered cluster contains a near-uniform mixture of all five injected policy variants (14–29% each), demonstrating that the VQ-VAE recovers structure that cuts across policy identity. This property is precisely what is needed for a strategy classifier intended to generalize across diverse real-world players instead of simply reproducing training conditions.

A second variant of the model was trained with a decoder redesigned to predict tactical outcomes instead of reconstruct low-level action and position data. Instead of learning to recover what each unit did and where it moved, this decoder was trained to answer whether each turn produced consequential game-state changes such as whether enemies were neutralized, buildings were protected, grid power was preserved, and mech damage was sustained. The shift in decoder objective produced a noticeably sparser codebook. Instead of populating all five available strategy slots, the model converged on three high-utilization clusters, each defined by a distinctive profile of combat outcomes and temporal rhythm. One cluster is characterized by high enemy elimination and early mission resolution, another reflects a sustained, moderate-attrition approach spanning the full engagement window, and the third achieves mission success through low direct kills, suggesting a positioning and displacement-dominant style that avoids lethal confrontation. The distribution of injected policy variants across these outcome-defined clusters remains near-uniform, once again demonstrating that the model's learned representations do not recover the hand-specified training labels. Together, the two model variants suggest that meaningful strategy structure in multi-agent adversarial play can be captured from either behavioral traces or their downstream consequences, and that in both cases, the emergent organization reflects genuine tactical diversity instead of artifacts of training label design.

A third model configuration extended the first model by introducing an additional loss component alongside the existing action and position reconstruction objective. The decoder was simultaneously trained to predict the identity of the injected policy variant responsible for each behavior sequence. The reconstruction objective remained intact, grounding the latent representation in the same behavioral content as the first model, however, the auxiliary classification loss introduced a supervisory signal that rewarded the encoder for producing codebook embeddings from which policy identity could be inferred. The effect was a decisive shift in how the codebook organized the data. While the first model, trained on reconstruction alone, produced clusters with near-uniform policy variant distributions, indicating that its representations were organized around features unrelated to the injected labels, the multi-task model produced clusters each dominated by a single variant, effectively recovering the imposed policy taxonomy instead of discovering emergent structure. This demonstrates that even a supplementary loss term is sufficient to redirect the codebook's organizational logic, and that the features necessary to discriminate between labeled policy styles are meaningfully different from those that arise when the model is free to organize behavior by other criteria.

The primary contributions of this work are threefold. First, we present a label-free, unsupervised strategy classification pipeline for turn-based tactical games using a VQ-VAE with transformer encoder, and demonstrate that models trained on reconstruction and outcome-prediction objectives converge on strategy clusters that do not correspond to the policies used to generate training data, indicating that the representations reflect emergent behavioral structure instead of the designer's labeling scheme. Second, we provide a systematic comparison of three decoder configurations: reconstructive, outcome-predictive, and label-supervised, that together reveal the decoder objective as the primary driver of codebook organization. The first two

models discover policy-agnostic structure, while the introduction of a supplementary variant-prediction loss in the third model is sufficient to redirect the codebook toward recovering the imposed policy taxonomy. Third, we validate this comparative framework as a practical guide for strategy-aware adversarial agent training, offering evidence that unsupervised representation learning can surface tactically meaningful structure that labeled policy design does not anticipate.

The remainder of this report is organized as follows. Chapter 2 reviews related work in reinforcement learning, adversarial game AI, and representation learning. Chapter 3 describes the game environment, dataset generation procedure, and model architecture. Chapter 4 presents experimental results and post-training analysis across all three model variants, examining how decoder objective shapes the structure of learned strategy representations. Chapter 5 discusses the implications of these findings and limitations of the current approach. Chapter 6 recommends directions for future work.

CHAPTER 2. RELATED WORK

This chapter reviews the bodies of literature that inform the design and evaluation of the adversarial gameplay modeling system presented in this work. The relevant areas span deep reinforcement learning with policy gradient methods, multi-agent reinforcement learning, discrete latent representation learning, transformer architectures for sequential encoding, and prior work on adversarial policy learning and the assessment of strategic intelligence in games. Each area contributes a foundational component to the system described in the following chapters.

Policy Gradient Methods in Deep Reinforcement Learning

The agents in this work are trained using Proximal Policy Optimization (PPO), a widely adopted policy gradient algorithm introduced by Schulman et al. (2017). PPO addresses a fundamental instability in earlier policy gradient methods. Since gradient updates that are too large can push the policy into regions of parameter space from which recovery is difficult, naive approaches tend to suffer from high variance and sample inefficiency. PPO resolves this by introducing a clipped surrogate objective that constrains the magnitude of policy updates at each training step, preventing the policy from changing too drastically between iterations. The resulting algorithm is computationally simple, requiring no second-order optimization or trust region projections, and empirically robust across a wide range of continuous and discrete control tasks, making it a practical default for deep reinforcement learning applications.

A complementary contribution from Schulman et al. (2016) introduced Generalized Advantage Estimation (GAE), a technique for computing low-variance advantage estimates in policy gradient methods. Advantage estimation measures how much better a particular action is than the expected behavior of the current policy and is central to the stability of policy gradient

training. GAE parameterizes the trade-off between bias and variance in advantage estimates using an exponential weighting scheme, allowing practitioners to tune toward lower variance without discarding temporal credit assignment information. The GAE formulation is directly compatible with PPO and is employed in the training procedure described in Chapter 3.

The broader challenge of achieving competent game-playing behavior through deep reinforcement learning is illustrated by Agent57, introduced by Puigdomènech Badia et al. (2020). Agent57 demonstrated that a single agent architecture, equipped with a population of exploration policies and a meta-controller for selecting between them, could surpass human-level performance across all 57 games in the Atari benchmark suite. The work is notable for its framing of the challenge: broadly competent game play requires systematic exploration, memory, and the ability to balance short-horizon reward exploitation against long-horizon return maximization. These considerations motivate the present work's interest in characterizing emergent strategic variety instead of optimizing a single high-performing policy.

Multi-Agent Reinforcement Learning

Many sequential decision problems involve multiple interacting agents whose reward signals are interdependent, requiring frameworks that account for the non-stationarity introduced by co-evolving policies. Lowe et al. (2017) introduced Multi-Agent Deep Deterministic Policy Gradient (MADDPG), a framework for cooperative and competitive multi-agent settings that employs centralized training with decentralized execution. During training, each agent's critic has access to the observations and actions of all agents, enabling stable gradient estimates even as other agents' policies change. During execution, each agent acts using only its own observations. MADDPG was demonstrated to outperform independent learning baselines on a set

of mixed cooperative-competitive environments, and its centralized training paradigm has become a widely adopted design pattern in multi-agent reinforcement learning.

The applicability of PPO to cooperative multi-agent settings was systematically examined by Yu et al. (2022), who showed that a straightforward multi-agent extension of PPO, referred to as MAPPO, matched or outperformed more specialized algorithms across several established benchmarks. This finding is relevant to the present work because it suggests that the simplicity and stability of PPO extends to multi-agent configurations without requiring specialized algorithms. The authors also examined the sensitivity of MAPPO to design choices including value normalization and the structure of the critic, providing practical guidance that informs the training setup described in Chapter 3.

Discrete Latent Representation Learning

The representation learning component of this work is built on the Vector Quantized Variational Autoencoder (VQ-VAE), introduced by van den Oord, Vinyals, and Kavukcuoglu (2017). The VQ-VAE addresses a well-known limitation of continuous variational autoencoders, specifically, the tendency of continuous latent spaces to produce diffuse, difficult-to-interpret representations, by replacing the continuous latent variable with a discrete codebook of learned embeddings. During the forward pass, each encoder output is mapped to its nearest codebook entry via an argmin lookup. The decoder then reconstructs the input from this discretized representation. Since the argmin operation is not differentiable, gradients are passed from decoder to encoder using a straight-through estimator, while codebook vectors are updated via exponential moving average. The result is a representation learning framework that imposes discrete structure on continuous observations without requiring predefined categories.

Razavi et al. (2019) extended the VQ-VAE to a hierarchical architecture called VQ-VAE-2 that could generate high-fidelity images by learning discrete representations at multiple scales. The top-level codebook captures global structure while lower-level codes encode local detail, enabling coherent generation across spatial scales. Although the present work operates in a lower-dimensional behavioral domain instead of image generation, the VQ-VAE-2 results confirm that codebook embeddings can organize semantically meaningful structure at multiple levels of abstraction. The behavior-level codebook used in this work functions analogously to the global top-level code, capturing turn-level strategic patterns instead of the fine-grained details of individual actions. The choice of VQ-VAE as the core representational framework is motivated by two properties that are particularly valuable for strategy discovery: its ability to induce a finite set of categories from continuous behavioral trajectories without requiring the researcher to specify those categories in advance, and its tendency to reflect genuine statistical regularities in the data instead of imposed labels.

Transformer Architectures for Sequential Encoding

The encoder component of the VQ-VAE in this work is implemented as a transformer, following the architecture introduced by Vaswani et al. (2017) in the context of natural language processing. The transformer replaces recurrent architectures with self-attention mechanisms that compute pairwise relationships between all positions in an input sequence simultaneously, enabling parallelizable training and capturing long-range dependencies without the vanishing gradient problems associated with recurrent networks. Each self-attention layer produces context-aware representations by attending to all other positions with learned attention weights, and the outputs are refined through position-wise feedforward networks with residual connections and layer normalization. The transformer architecture has since become the

dominant backbone for sequence modeling across a wide range of modalities, including code, protein sequences, audio, and game state representations.

The feedforward sublayers in the transformer encoder used in this work employ Gaussian Error Linear Units (GELUs), introduced by Hendrycks and Gimpel (2016) as a smooth alternative to the ReLU activation. GELUs weight each input by the probability that it would be drawn from a standard normal distribution, producing a gating effect in which smaller-magnitude activations are attenuated more strongly. Empirically, GELUs have been shown to outperform ReLUs and ELUs across a variety of deep learning settings, and their use has become standard practice in transformer implementations.

Adversarial Policy Learning and Strategic Intelligence in Games

The challenge of building game agents that demonstrate genuine strategic flexibility, instead of narrow expertise optimized for fixed task configurations, is addressed philosophically and formally by Chollet (2019), who argues that measuring artificial intelligence solely through skill at predefined tasks conflates task-specific optimization with general intelligence. Chollet proposes that meaningful evaluation requires assessing generalization to novel configurations, with performance measured relative to the prior knowledge available to the system. In the context of game AI, this argument implies that an agent achieving high win rates under a fixed opponent policy may not have learned strategically meaningful representations, it may simply have overfit to the behavioral regularities of that particular opponent. This perspective motivates the unsupervised strategy discovery approach of the present work, in which the goal is not to produce an agent optimized for a single fixed setting, but to learn representations that reveal the underlying strategic structure of gameplay independent of which specific policies generated the training data.

CHAPTER 3. METHODOLOGY

This section describes how the game environment was structured and modified, how a PPO player agent model (Model 1) was used to generate datasets to train the VQ-VAE strategy classifier model variants (Models 2.1, 2.2, and 2.3), how the VQ-VAE is structured, how the PPO adversary model (Model 3) is structured, and how the VQ-VAE strategy classifier (Model 2.2) was used as a policy router in the adversary model (Model 3).

The game environment was designed to be played by a human player against a heuristic adversary. A three-model approach was used to classify and mitigate strategies (Figure 3-1) in the following steps:

1) Replace the human player with a PPO player agent model (Model 1).
2) Collect data from games won by Model 1 against the heuristic-based adversary.
3) Train the VQ-VAE strategy classifier (Model 2) on the data collected by Model 1.
4) Use Model 2 to route classified player turns to a multi-policy adversary model (Model 3) during training.
5) Play Model 3 against Model 1, using Model 2 to select which policy Model 3 uses to counter each turn from Model 1. Compare this routed approach to each individual adversary policy.
6) As a follow-up, have a human play against Model 3 and evaluate subjectively how Model 3 plays.

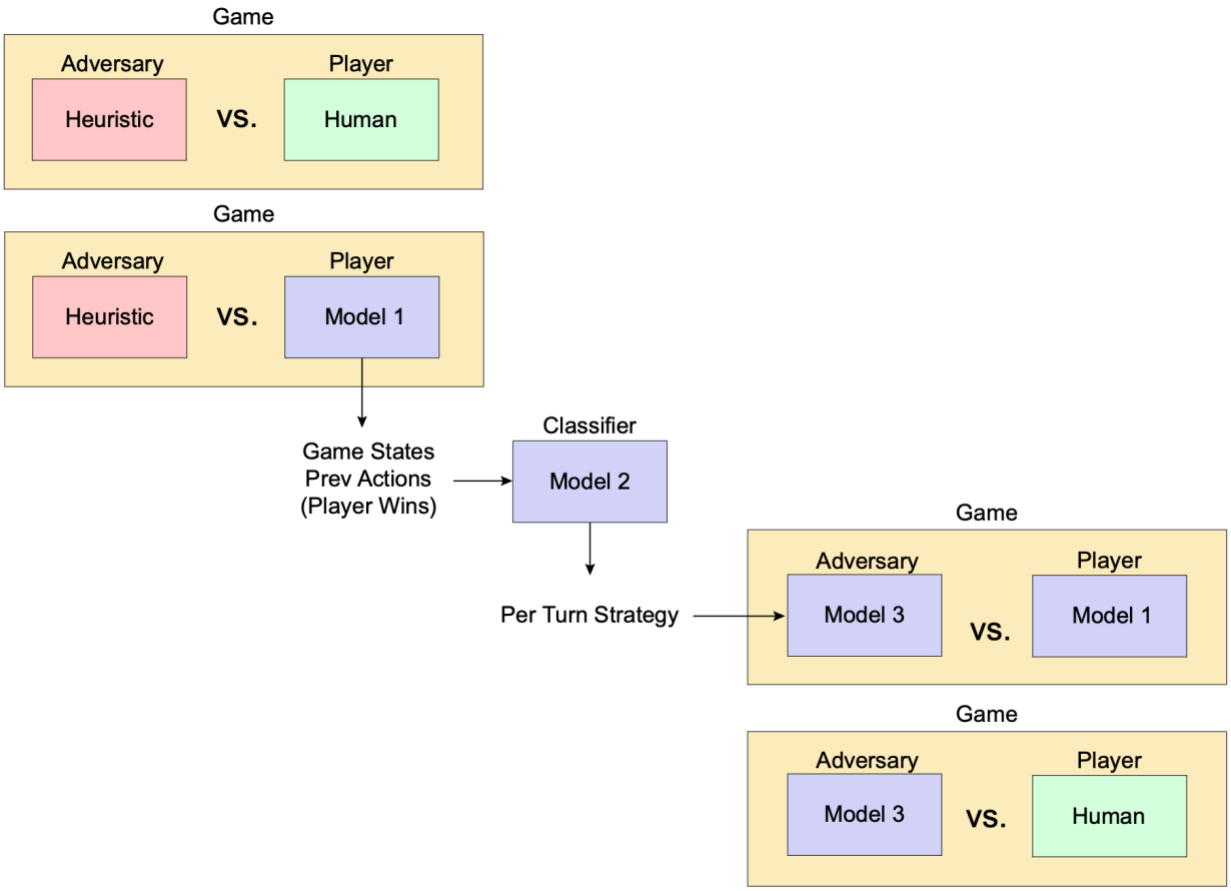


Figure 3-1. The role of Model 1, Model 2, and Model 3 in the approach for strategy classification and mitigation.

Game Environment

The game environment used in this work was inspired by Subset Games' Into the Breach, and was developed by an undergraduate team of computer science majors at ISU as part of their senior project. Due to limitations of parallel timelines, the environment was provided in a functional, but minimal state so that graduate teams could begin work.

Objective

The objective of the game is for the player to survive attacks by the adversary over 5 turns through movement and/or counter attacks.

Game Board Generation

The game environment procedurally generates maps represented by an 8x8 grid of green grass, blue water, and gray mountain tiles. The grid is initialized with all 64 tiles set to grass. A terrain budget is then drawn uniformly at random from the integer range [70, 90]. This budget represents the maximum allowable sum of tile movement costs across the entire map, where grass costs 1, water costs 2, and mountain costs 3. Since the initial all-grass configuration has a total cost of 64, the budget permits between 6 and 26 additional cost units to be distributed across the map as harder terrain. To place this harder terrain, all 64 tile positions are randomly shuffled. The algorithm iterates through them in this shuffled order. At each position, it considers whether any valid upgrade exists with an upgrade being a tile type more costly than the current one that would not cause the running total to exceed the budget. If valid upgrades exist, one is chosen uniformly at random and applied. The process stops as soon as the total cost reaches the budget. The result is a map whose terrain density varies with the budget draw. Low budgets produce mostly flat maps with sparse obstacles, while high budgets produce maps with many water and mountain tiles.

Building Placement

After terrain is generated, between 3 and 6 buildings are placed on the map. The exact number is drawn uniformly at random. Buildings are restricted to grass tiles only, so the algorithm first collects all remaining grass positions, shuffles them, and replaces the first N with buildings, where N is the drawn count. Buildings serve as objectives: the player loses grid power when a Vek unit destroys one. Figure 3-2 shows a fully initialized gameboard.

Unit Spawning

Once the terrain and buildings are in place, units are spawned by the EntityManager. Three player mechs are placed at fixed starting positions. For each mech, the game collects all unoccupied grass tiles and picks one uniformly at random. Each mech is placed before the next one's valid positions are computed, so no two mechs can share a tile. A rule was later added that all three classes of mech are spawned in the following order: CombatMech, ArtilleryMech, and CannonMech as this is consistent with Into the Breach and prevents more than one CombatMech, a more powerful unit, from being spawned more than once per game. Three Vek adversary units are then spawned. For each, an enemy type is chosen uniformly at random from Hornet or Firefly. The valid spawn positions depend on the type: Fireflies may only spawn on grass, while Hornets may spawn on grass or water.

Unit Abilities

All three Mech can either 1) move then attack, 2) move only, or 3) attack only within a single player turn. The Vek pre-turn at the start of the game involves a move only with an attack intent for each Vek indicated with a red highlighted tile. All subsequent Vek turns begin with each Vek attacking their intent tile, followed by a move with a new intent indicated. Table 3-1 delineates the specific Manhattan movement ranges for each class of Mech and Vek. Table 3-2 lists the attack profiles for each unit type. Table 3-3 lists damage inflicted and initial hit points by unit. Knockback occurs when a Vek is pushed 1 square away in the direction of the attack due to a direct hit from Mech (CombatMech, CannonMech), or an adjacent tile attack (ArtilleryMech). If the Vek collides with a building, mountain, or edge of the gameboard, it receives 1 additional damage to its HP. In the case of a building collision, the building is destroyed, and the grid power reduced by 1. In the case of a mountain collision, the mountain tile becomes a grass tile.

Table 3-1: Unit movement profiles

Unit	Side	Move Range	Valid Terrain
CombatMech	Player	3 tiles	Grass, Water
ArtilleryMech	Player	3 tiles	Grass, Water
CannonMech	Player	3 tiles	Grass, Water
Firefly	Adversary	2 tiles	Grass only for movement, can be knocked into Water
Hornet	Adversary	5 tiles	Grass, Water

Table 3-2: Unit attack profiles

Unit	Attack Range	Pattern	Invoked Knockback
CombatMech	1 tile	4 adjacent cardinal tiles only	1 tile straight
ArtilleryMech	2-3 tiles	Cardinal directions, skips adjacent tile	1 tile radial (adjacent)
CannonMech	1-3 tiles	Cardinal directions	1 tile straight
Firefly	Up to 7 tiles	Cardinal directions to board edge	None
Hornet	1 tile	4 adjacent cardinal tiles only	None

Table 3-3. Unit damage and hit point profiles

Unit	HP damage per attack	Hit Points (HP)
CombatMech	2	3
ArtilleryMech	1	2
CannonMech	1	5
Firefly	1	2
Hornet	1	2

Modifications to the Game Environment

Several modifications were made to the game environment provided by the undergraduate students.

Adversary Heuristic

The original game had a placeholder heuristic for determining how the Vek moved and attacked that randomly moved Vek to valid locations and chose “intent” tiles randomly from the valid set of those within range. The heuristic was modified such that each Vek unit independently rolls a random preference at the start of its decision: 75% chance it prefers targeting a building,

25% chance it prefers targeting a mech. Each Vek then simulates every possible move position available to it, and for each one checks what attacks would be reachable from there. It builds two candidate lists, one for moves that would let it attack a building, one for moves that would let it attack a Mech, with each entry storing the Manhattan distance from the Vek's current position to the attack target. Both lists get sorted by that distance, and the Vek picks the closest target from its preferred list. If the preferred list is empty (e.g., no buildings are reachable this turn), it falls back to the other list. If neither list has anything, meaning no attack is possible from any move position, the Vek instead just moves toward the nearest target of its preferred type, using Manhattan distance to pick both the target and the best move tile. It sets no attack intent in this case. The distance used for sorting is to the attack target, not to the move position. The Vek prioritizes the target it can threaten that is already closest to where it currently stands..

Grid Power

The game was initially too easy for the player to win because they only needed to outrun the Vek for 5 turns. A grid power element was added that acted as a universal health meter, such that in addition to killing all the Mechs, the Vek could also win by reducing the grid power to zero by destroying buildings on the map. This was a key element of Into the Breach that requires the player to balance the strategic tradeoff between protecting units and protecting buildings, sometimes at the cost of units. Grid Power was adjusted to always be 3 as any higher starting value made winning too easy for the player in early tests.

Unit Spawning

Originally, a varied combination of player unit classes could spawn on the map. In some cases, multiple CombatMech would be spawned, and since each CombatMech can inflict 2 damage per turn and all Vek have 2 hit points, Vek could easily be killed in one or two turns. A

rule was therefore put in place to select one of each class of Mech for the three spawned. The Vek were left to spawn in random combinations of three.

Mountain Conversion

Mountain tiles are not valid movement positions for any unit type, so they act as obstacles to movement and attack. In some procedurally generated maps, Mech and Vek were permanently fenced into zones of the map, and in some cases could not move at all. Mountains were therefore given 1 hit point (HP) so they could be converted to grass by Mech or Vek with one attack action.

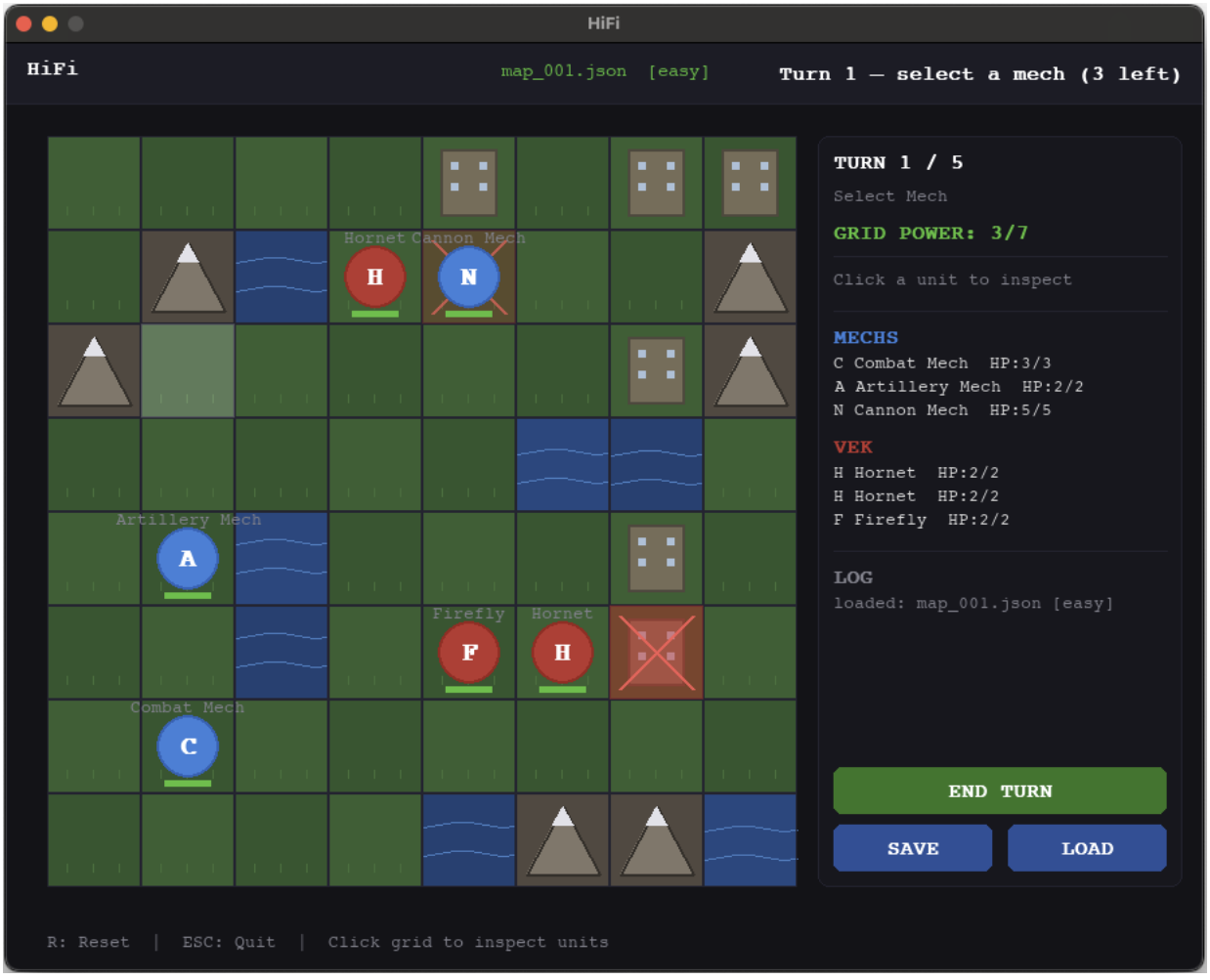


Figure 3-2. Gameboard map_001 of the easy set of 20 maps. Mech are in blue, Vek are in red.

Gameboard Map Curation Procedure

The VQ-VAE strategy classifier model variants require a dataset consisting of games won by the player against the adversary. Since there were no pre-existing datasets from human players of the game, a PPO player agent model was developed (Model 1) that trained to win against the heuristic-driven adversary. Originally, procedurally generated gameboard maps were used to train Model 1, but it was discovered that some maps were still "unwinnable" by the player, even with the added ability to convert mountain tiles to grass. To solve this problem, 20 "easy" and 20 "difficult" gameboard arrangements were manually selected via human play, their difficulty subjectively judged. Figure 3-3 shows the customized user interface for saving "winnable" gameboard maps. The "easy" and "difficult" maps were eventually combined into a single 40 maps set because Model 1 was not able to win against certain maps in both categories.

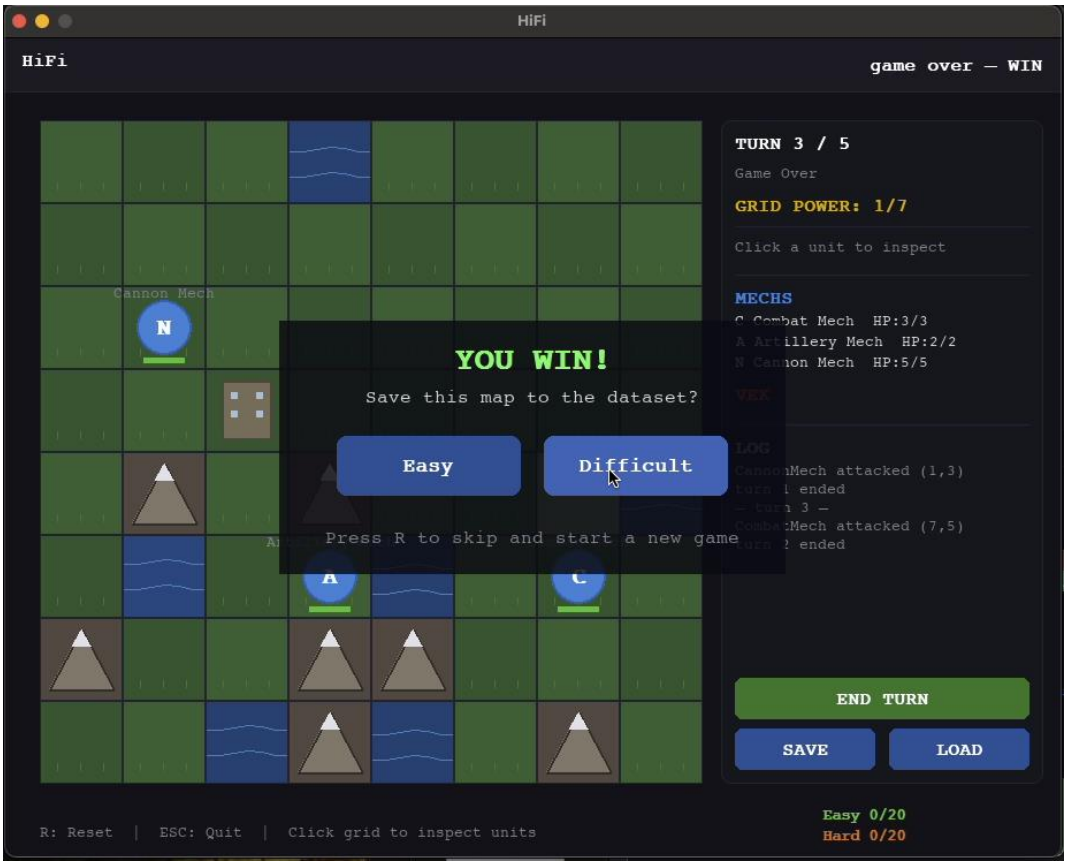


Figure 3-3. The interface used for manual selection of "winnable" gameboard maps.

Model 1 - Player Agent

Five behavioral variants of the player agent were trained on the full pool of 40 fixed winnable maps using reward shaping to induce qualitatively distinct playing styles. All variants share the same network architecture and PPO training infrastructure, differing only in the weights assigned to reward signal components (Table 3-4). Each variant uses a shared-trunk actor-critic network consisting of a two-layer MLP encoder (hidden sizes 512, 512, Tanh activations) followed by a single-layer GRU (hidden size 128) that carries state across the sub-steps within a turn. The actor and critic heads are linear projections from the GRU output. All layers are orthogonally initialized. Illegal actions are masked by setting their logits to -10^9 prior to sampling, ensuring the agent never selects a structurally invalid action.

Behavioral Variants

All five variants are trained on the complete 40-map pool with intentional overfitting, so no train/validation split is applied. This design choice reflects the goal of producing policies that reliably and distinctively solve these specific maps, instead of generalizing to unseen configurations. At each episode reset, a map is sampled uniformly at random from the pool and its exact initial state is restored from the saved JSON, giving the agent deterministic and reproducible starting conditions. Each variant is trained for 5,000,000 environment sub-steps using PPO with a rollout buffer of 2,048 steps, 4 update epochs per rollout, minibatch size 256, learning rate 3×10^{-4} with linear decay, clipping parameter $\varepsilon = 0.2$, generalized advantage estimation (GAE) ($\gamma = 0.99$, $\lambda = 0.95$), and online observation normalization via a running mean and standard deviation.

The five variants are differentiated through reward shaping (Table 3-4). The balanced variant uses equal weighting across kills (+0.3), mech loss (−0.3), and building loss (−0.2), with

a win bonus of +1.0 and loss penalty of −1.0. The aggressive variant doubles the kill reward (+0.6) while substantially reducing penalties for losing mechs (−0.1) and buildings (−0.05), encouraging high-tempo offensive play. The defensive variant inverts this priority, imposing a heavy mech loss penalty (−0.8) and an elevated survive-turn bonus (+0.2), shaping the agent to prioritize unit preservation above kills. The guardian variant shifts emphasis to building protection, assigning a heavy building loss penalty (−0.6) while moderating kill and mech survival rewards, producing an agent primarily concerned with maintaining grid power. Finally, the efficient variant adds an early-win bonus that scales with turns saved ($0.25 \times \max(0, 5 - \text{turn})$) and a strong all-mechs-acted bonus (+0.3), incentivizing the agent to act decisively with every unit and close out games as quickly as possible. These five trained policies serve as frozen player agents in the subsequent adversary training stage.

Table 3-4. Reward components for each player agent policy (induced strategy).

Reward	**balanced**	**aggressive**	**defensive**	**guardian**	**efficient**
kill_reward	0.3	0.6	0.15	0.2	0.3
mech_loss_penalty	-0.3	-0.1	-0.8	-0.2	-0.2
building_loss_penalty	-0.2	-0.05	-0.15	-0.6	-0.2
survive_turn_bonus	0.1	0.05	0.2	0.1	0.02
all_mechs_bonus	0.15	0.2	0.1	0.15	0.3
idle_mech_penalty	-0.1	-0.15	-0.05	-0.1	-0.2
win_bonus	1.0	1.0	1.0	1.0	1.0
lose_penalty	-1.0	-0.5	-1.5	-1.0	-1.0
early_win_bonus	0.0	0.0	0.0	0.0	0.25

The observation vector components provided by the game environment to Model 1 are shown in (Table 3-5). These components were chosen to represent as much of the game state and previous action as possible.

Table 3-5. Observation vector used in Model 1.

Component	Size	Description
token_id	1	Current token type (see Table 3-9)
prev_action	11	Previous action encoding (see Table 3-10)
global_scalar	1	Grid Power (See Table 3-11)
unit_table	117	9 units × 13 channels (see Table 3-12)
tile_sensor	1088	17 channels × 8×8 board (see Table 3-13)
move_masks	576	9 units × 8×8 - legal moves per unit (see Table 3-14)
attack_masks	576	9 units × 8×8 - legal attacks per unit (see Table 3-14)
vek_intents	384	6 Vek slots × 8×8 - planned attack tile per Vek (see Table 3-14)
phase	1	Current UI phase (0=mech_select, 1=mech_move, 2=mech_attack)
selected_slot	1	Which mech slot is currently selected (0, 1, or 2)
Total	2756	total observations

Data Collection Procedure

Once Model 1 was trained, game-play data was collected from each of the five trained variant policies by running them against the game environment's built-in heuristic Vek adversary across all 40 fixed winnable maps. This was done using a dedicated capture script that systematically iterates over every combination of variant and map, playing each map exactly once per variant. Action selection is deterministic, the policy takes the argmax of the masked action distribution at every sub-step, ensuring that each variant produces a single reproducible trajectory per map. The GRU hidden state in Model 1 is carried across sub-steps within a turn but reset to zero at the start of each new map, consistent with the conditions used during training. A hard cap of 300 agent sub-steps per map prevented infinite loops on maps the agent fails to resolve, which are recorded as timeouts and excluded from the saved output.

Across all five variants and 40 maps, this process yields up to 2000 winning game sequences (5 variants × 40 maps × 10 winning games), providing a structured dataset of how each reward-shaped behavioral strategy approaches and solves each specific map configuration. These games were randomly shuffled into 80/10/10 splits for training, validation, and test. In

practice, some maps could not be won within the 500-attempt limit by the player agent when using certain policies such that for Model 2 variants 2.1 and 2.3, the actual split resulted in 1,440/180/180 winning games. For Model 2.2, a 1520/190/190 split was achieved. Decoder changes to Model 2.2 required additional metrics (Vek pushed, buildings saved) stored and therefore required dataset retraining. The observation vector was not changed. Datasets were stored in HDF5 file format. Table 3-6 and Table 3-7 list which maps were unwinnable by certain policies. One map, map_018, was unwinnable by all policies for both datasets. A look at the initial game state of map_018 as shown in Figure 3-4 reveals that surviving the first turn requires either a CannonMech attack that invokes a knockback into the gameboard edge to kill the Vek in row 8, column 3, or an ArtilleryMech attack, followed by a CannonMech attack. Any other combination of actions in the first player turn results in a game over for the player agent. This map was subjectively labeled "easy" during manual play because a winning tactic was quickly devised and deployed, without noticing the limited number of ways the game could be won.

Table 3-6. Gameboard maps that could not be won by policy for Model 2.1 and 2.3 dataset.

Map	**Missing Variant Policies**
difficult_map_008	guardian, efficient
difficult_map_015	aggressive
difficult_map_020	guardian
easy_map_012	balanced, defensive, guardian
easy_map_013	defensive
easy_map_018	All (no policies could win)

Table 3-7. Gameboard maps that could not be won by policy for Model 2.2 dataset.

Map	**Missing Variant Policies**
difficult_map_015	aggressive
difficult_map_020	guardian
easy_map_012	balanced, defensive, guardian
easy_map_018	All (no policies could win)

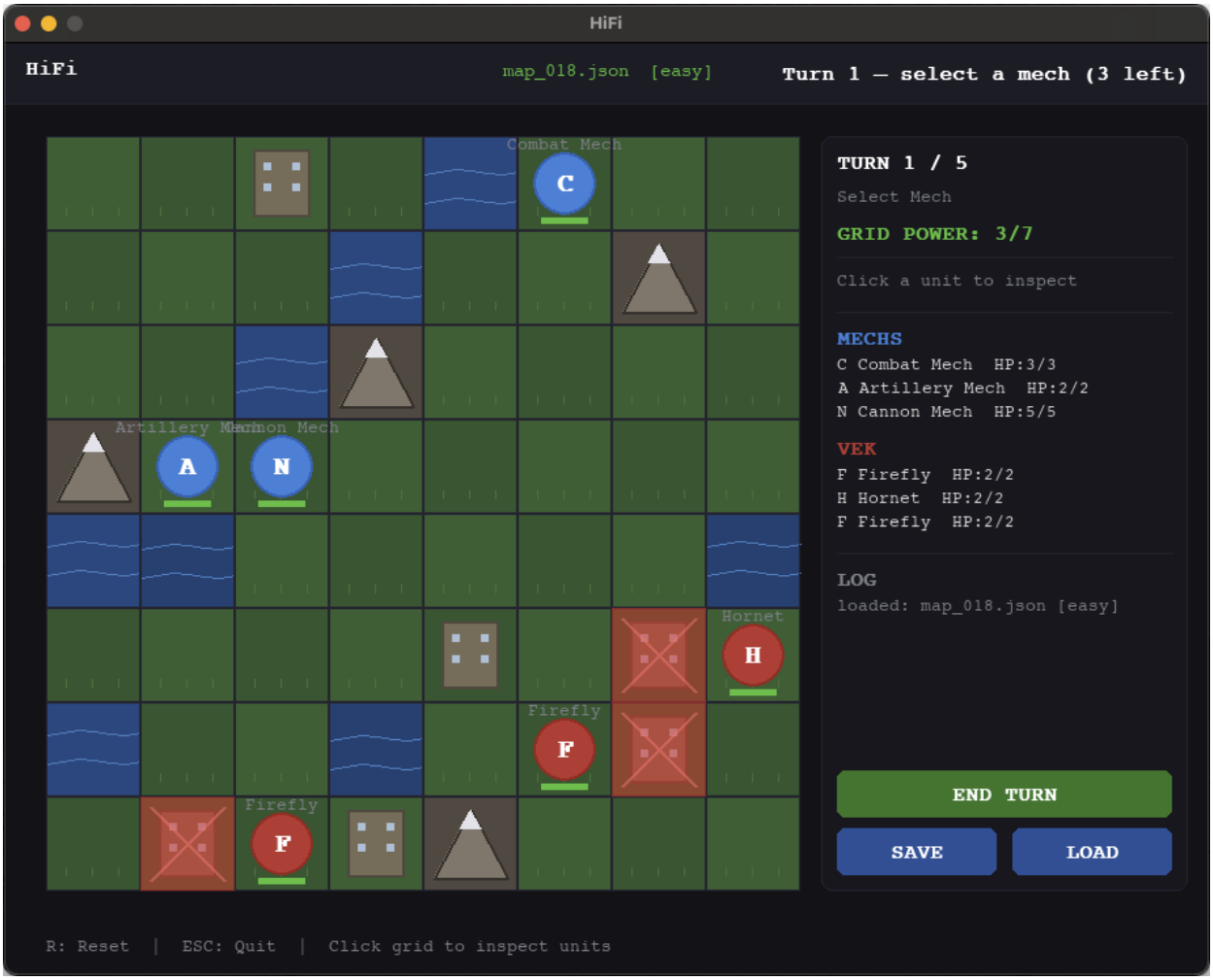


Figure 3-4. The map_018 that was found to be unwinnable by all policies trained by Model 1.

Model 2 - VQ-VAE Strategy Classifier

Model Architecture

The strategy classification model is a Vector Quantized Variational Autoencoder (VQ-VAE) built around a Transformer encoder, designed to compress full game trajectories into discrete strategy codes that correspond to the five behavioral variants of the player agent. Three variants for the model (2.1, 2.2, and 2.3) were trained and tested. They all shared a common transformer and codebook structure, varying only in the decoder and composite loss (Figure 3-6).

Transformer Encoder

The encoder consists of 2 Transformer blocks, each implementing post-layer-normalization multi-head self-attention (8 heads) with a feed-forward network of inner dimension 1,024 using GELU activations, residual connections, and dropout ($p = 0.1$). Padded token positions are masked from attention. The encoder produces a per-token contextualized embedding $z_e \in \mathbb{R}^{B \times T \times 128}$.

Turn-Level Vector Quantization

Instead of quantizing at the individual token level, which in earlier tests only produced token level strategy patterns, the model quantizes at the player turn level. Tokens are first assigned to player turns: a new turn begins at each START (token ID 1) or ENEMY_PHASE (token ID 6) token. Within each turn, only MOVE (2) and ATTACK (3) tokens contribute (Figure 3-5). These are mean-pooled across their encoder outputs to produce a single turn-level embedding $\overline{z}_e \in \mathbb{R}^{B \times 5 \times 128}$ (up to 5 turns per game). This design forces the codebook to capture variation in how the player acted within a turn, rather than the structural pattern of which token types appear.

The turn embeddings are quantized using a Vector Quantizer with Exponential Moving Average (EMA) codebook updates. The codebook contains $K = 5$ entries (matching the number of behavioral variants), each of dimension 128. During training, each turn embedding is assigned to its nearest codebook vector by L2 distance; codebook entries are updated via EMA with decay $\gamma = 0.95$ and Laplace smoothing to prevent cluster collapse.

The commitment loss $\mathscr{L}_{\text{commit}} = \beta \cdot |\overline{z}_e - \text{sg}[z_q]|^2$ (with $\beta = 0.25$) trains the encoder to commit to codebook entries ($\text{sg}[\cdot]$ denotes the stop-gradient operator), while gradients are passed

through the quantization step via the straight-through estimator. The resulting discrete codebook index per turn serves as the strategy label for that turn.

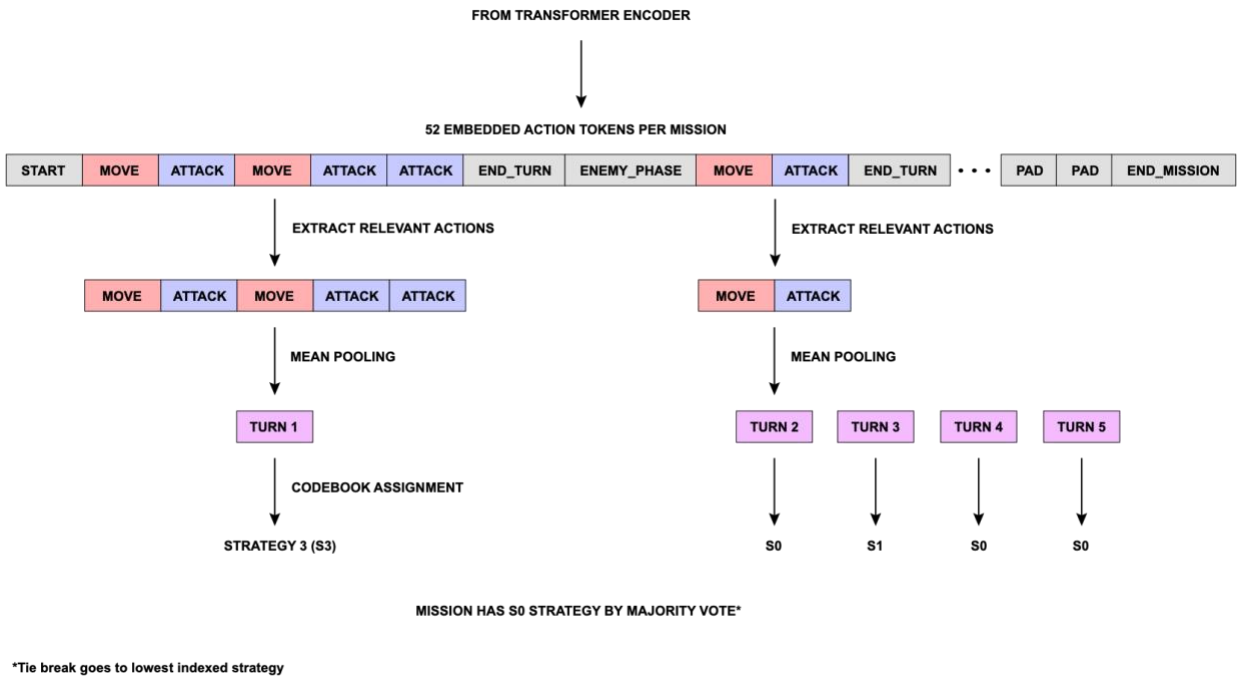


Figure 3-5. Turn-level embedding of tokens for quantization and mission-level (game-level) strategy assignment by majority vote of turn strategy.

The quantized turn embedding is then broadcast back from the turn level to every token in that turn, producing $z_{qtok} \in \mathbb{R}^{B \times T \times 128}$.

Decoder Skip Connection

To give each decoder head access to both fine-grained per-token information and the coarser turn-level strategy, the raw encoder output ze and the broadcast quantized embedding z_{qtok} are concatenated before decoding:

Equation 3-1

$$z_{dec} = [z_e \,|z_{qtok}] \in \mathbb{R}^{B \times T \times 256}$$

This skip connection resolves the information bottleneck in which all tokens within a turn would otherwise share identical decoder input.

Decoder Heads

Every decoder head (except the variant head) follows:

$$\texttt{Linear}(256, 256) \rightarrow \texttt{GELU} \rightarrow \texttt{Linear}(256, C)$$

where the input dimension is always 256 ($2 \times d_{\text{model}} = 2 \times 128$) and C is the number of output classes for that head. Heads that are disabled (commented out of the loss) are still present as trained parameters, they just receive no gradient signal. The variant classification head differs from all others. It receives only the mean quantized turn embedding z_q (dimension 128, not 256), so:

$$\texttt{Linear}(128, 256) \rightarrow \texttt{GELU} \rightarrow \texttt{Linear}(256, 5)$$

It is smaller on the input side because it deliberately excludes the z_e skip connection. Only the quantized (strategy-level) representation feeds it, so gradients flow directly back through the codebook rather than through the per-token encoder output.

Loss Calculation

All three model variants share a commitment loss that trains the encoder to produce turn embeddings close to their assigned codebook entries, while the straight-through estimator passes gradients back through the quantization step. Denoting the turn-level encoder output as $\bar{z}_e$ and the quantized codebook vector as z_q, the commitment loss is

Equation 3-2

$$\mathcal{L}_{commit} = \beta \cdot \left|\bar{z}_e - sg\left[z_q\right]\right|^2 \quad (\beta = 0.25)$$

where sg[·] denotes the stop-gradient operator. The models differ in their task loss and, in the case of Model 2.3, an additional auxiliary loss.

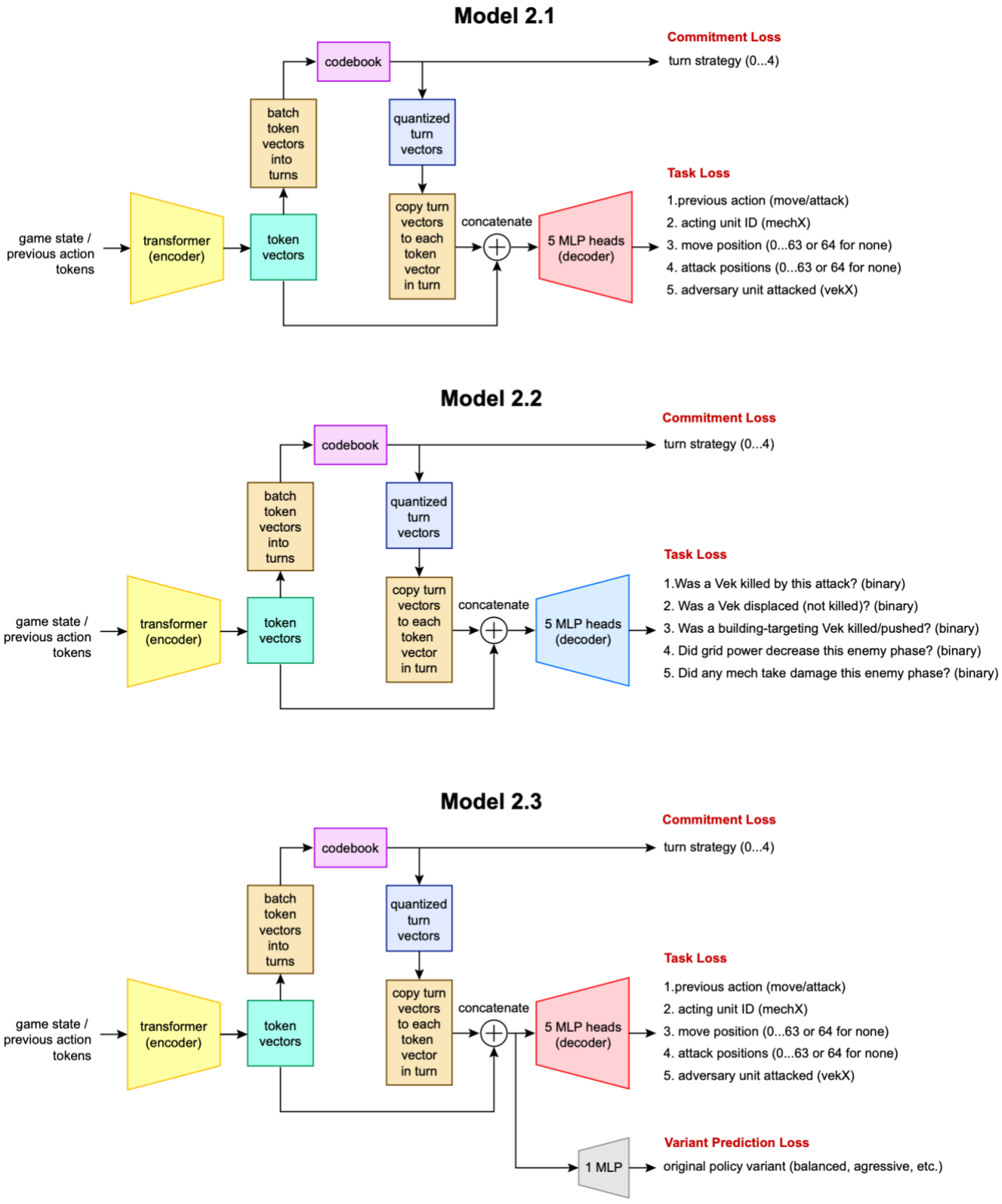


Figure 3-6. The three variants of the VQ-VAE (Model 2) decoder configurations.

Models 2.1 and 2.3 - Action Reconstruction Task Loss

Models 2.1 and 2.3 share an action reconstruction objective. Five independent MLP decoder heads operate on the concatenated encoder and quantized turn embedding z_{dec}, each predicting a distinct component of the action taken at each token: (1) action type (move or attack), (2) acting mech identity, (3) move tile (0–63, or 64 indicating no move), (4) attack tile (0–63, or 64 indicating no attack), and (5) the Vek unit targeted (Figure 3-6). Each head produces a discrete prediction over a fixed vocabulary, so each is trained with cross-entropy loss. The aggregate task loss is the unweighted sum across all five heads:

Equation 3-3

$$\mathcal{L}_{task} = \sum_{k=1}^{5} CE\,(\hat{y}_k, y_k)$$

where $\hat{y}_k$ and y_k are the predicted logits and ground-truth label for head k respectively. The total training objective for Models 2.1 and 2.3 (excluding the variant prediction term described below) is:

Equation 3-4

$$\mathcal{L}_{2.1} = \mathcal{L}_{task} + \mathcal{L}_{commit}$$

Model 2.2 - Outcome Prediction Task Loss

Model 2.2 replaces action reconstruction with an outcome prediction objective. Rather than recovering what action was taken, the five decoder heads each predict a binary outcome consequence of that action: (1) whether a Vek was killed, (2) whether a Vek was displaced but not killed, (3) whether a building-targeting Vek was killed or pushed, (4) whether grid power decreased during the enemy phase, and (5) whether any mech sustained damage during the enemy phase. Each head is trained with binary cross-entropy loss:

Equation 3-5

$$\mathcal{L}_{task} = \sum_{k=1}^{5} BCE\,(\hat{p}_k, p_k)$$

where $\hat{p}_k$ is the predicted probability and $p_k \in \{0, 1\}$ is the ground-truth binary outcome for head *k*. This objective forces the codebook to organize turns by their strategic consequences rather than by the literal actions taken. The total training objective for Model 2.2 is:

Equation 3-6

$$\mathcal{L}_{2.2} = \mathcal{L}_{task} + \mathcal{L}_{commit}$$

Model 2.3 - Additional Variant Prediction Loss

Model 2.3 extends the Model 2.1 objective with a supervised auxiliary loss. A separate single-layer MLP operating on z_{dec} predicts the identity of the behavioral variant (e.g. balanced, aggressive) that generated the trajectory. This is a five-class classification problem trained with cross-entropy:

Equation 3-7

$$\mathcal{L}_{variant} = CE(\hat{v}, v)$$

where $\hat{v}$ and *v* are the predicted logits and ground-truth variant label respectively. The total training objective for Model 2.3 is therefore:

Equation 3-8

$$\mathcal{L}_{2.3} = \mathcal{L}_{task} + \mathcal{L}_{commit} + \mathcal{L}_{variant}$$

The variant prediction loss is used to help provide an explicit supervision signal encouraging the codebook to align its discrete indices with the known behavioral categories, in contrast to Models 2.1 and 2.2 where strategy structure must emerge unsupervised from the reconstruction objective alone.

Input Features

The input features of the VQ-VAE model consist of two primary components: previous action, and the resulting game state from that action. These two segments comprise a single token. Tokens are sequenced in the order they occur in the game with a maximum of 52 possible tokens per game. Games where not all token slots are utilized, such as when either the player or adversary opt to not move or attack with one or more units, use a PAD token to fill the remaining token slots.

Table 3-8. Summary input features detailed in Table 3-9 through Table 3-14.

Field	Shape	Dimensions
token_ids	(52,)	52 tokens (see Table 3-9)
prev_action	(52, 11)	52 tokens × 11 fields (see Table 3-10)
global_scalar	(52,)	52 tokens (see Table 3-11)
unit_table	(52, 9, 13)	52 tokens × 9 slots × 13 channels (see Table 3-12)
tile_sensor	(52, 17, 8, 8)	52 × 17 channels × 8×8 board (see Table 3-13)
move_masks	(52, 9, 8, 8)	52 × 9 slots × 8×8 board (see Table 3-14)
attack_masks	(52, 9, 8, 8)	52 × 9 slots × 8×8 board (see Table 3-14)
vek_intents	(52, 6, 8, 8)	52 × 6 Vek slots × 8×8 board (see Table 3-14)
pad_mask	(52,) bool	52 tokens (see Table 3-14)

Table 3-9. Token ID feature – shape (52,)

ID	Meaning
0	PAD
1	START
2	MOVE
3	ATTACK
4	REPAIR (not used - placeholder for future feature)
5	END_TURN
6	ENEMY_PHASE
7	END_MISSION

Table 3-10. Previous action features – shape (52,11)

Index	Field	Valid Values
[0]	unit_slot	0–8 (0 = N/A, 1–3 = mech slots, 4–9 = vek slots)
[1]	action_kind	0 = none, 1 = mech action, 2 = vek resolve
[2]	move_tile	0–63 = tile index, 64 = N/A
[3]	attack_tile	0–63 = tile index, 64 = N/A
[4]	weapon_id	0 = N/A, 1 = punch, 2 = artillery, 3 = cannon
[5]	delta_grid_power	0, 1, 2, ... (buildings lost this enemy phase)
[6]	buildings_damaged	0, 1 (per attack token)
[7]	mechs_damaged	0, 1, 2, 3 (mechs damaged this enemy phase)
[8]	vek_killed	0, 1, 2, ... (veks killed this token)
[9]	Spawns	0 (always - placeholder)
[10]	threats_remaining	0 (always - placeholder)

Table 3-11. Grid Power feature – shape (52,)

Grid Power	0 to 7 (scalar)

Table 3-12. Unit table features – shape (52, 9, 13)

Channel	Field	Valid Values
0	Unit ID	1–9 (1-indexed slot number, always set)
1	Active?	0 = inactive/dead, 1 = alive
2	Team	1 = mech, 2 = vek
3	Class index	0=CombatMech, 1=ArtilleryMech, 2=CannonMech, 3=Firefly, 4=Hornet; 0 if inactive
4	Current HP	0–5 (varies by unit type); 0 if inactive
5	Max HP	CombatMech=3, ArtilleryMech=2, CannonMech=5, Firefly=2, Hornet=2; 0 if inactive
6	Move range	0–N tiles; 0 if inactive
7	Flying?	0 = grounded, 1 = flying (Hornet only)
8	Armored?	0 (always - placeholder)
9	Web Producing?	0 (always - placeholder)
10	Has Attack Order	0 = mech (no standing order), 1 = vek (always has order); 0 if inactive
11	Attack Rank	0 = mech; vek: slot_idx − 2 (1–6); 0 if inactive
12	Board position	0–63 = row×8+col; 64 = off-board/inactive

Table 3-13. Tile sensor features – shape (52, 17, 8, 8)

Channel	Field	Valid Values
0	Plains	0 or 1 (1 = Grass, Unknown, or Building base)
1	Water	0 or 1
2	Mountain	0 or 1
3	Forest	0 (always - placeholder)
4	On Fire	0 (always - placeholder)
5	Cracked	0 (always - placeholder)
6	Unit present?	0 or 1
7	Unit ID	0 = no unit, 1–9 (slot index + 1)
8	Unit team	0 = no unit, 1 = mech, 2 = vek
9	Unit HP	0–5
10	Unit max HP	0–5 (CombatMech=3, ArtilleryMech=2, CannonMech=5, Firefly=2, Hornet=2)
11	Flying?	0 = grounded, 1 = flying (Hornet only)
12	Armored?	0 (always - placeholder)
13	Web Producing?	0 (always - placeholder)
14	Building Present	0 or 1
15	Building HP	0 = no building, 1 = building at 1 HP
16	Instant kill hazard	0 or 1 (1 = Water tile, lethal to non-flying veks)

Table 3-14. Masks and intents features

Field	Shape	Description	Valid Values
move_masks	(52, 9, 8, 8)	Legal moves per unit slot	0 or 1 (1 = legal)
attack_masks	(52, 9, 8, 8)	Legal attacks per unit slot	0 or 1 (1 = legal)
vek_intents	(52, 6, 8, 8)	Planned attack tile per Vek slot	0 or 1 (1 = vek intent)
pad_mask	(52,)	True = PAD token	0 or 1 (1 = pad token)

Model 3 – Multi-Head PPO Adversary Agent

Overview

Model 3 is a multi-headed PPO adversary that controls the Vek (enemy units) against a frozen player policy (balanced) from Model 1. Instead of learning a single monolithic counter-strategy, it maintains one independent actor–critic policy per VQ-VAE codebook entry, five

heads in total, matching the codebook size K=5 (consistent across variants). At each turn, the frozen transformer (Model 2) classifies the player's most recently completed turn into one of the five codebook strategies, and the adversary routes its experience and action selection to the corresponding head. This allows each head to specialize in countering one particular behavioral cluster of the player.

Episode Structure and Action Decomposition

Each episode begins with a randomly sampled map restored to the initial game state with Vek at their original spawn positions. The initial_vek and initial_intents fields are stripped from the map data before loading, so the adversary takes full control of all Vek decisions from scratch instead of inheriting the heuristic's pre-computed moves.

The adversary's action space is decomposed into sequential sub-steps, one per Vek unit per phase. For each active Vek, the adversary first issues a move decision (actions 0–63 encoding a flat tile index, or 64 to stay in place), then an attack decision (0–63 for a target tile, or 64 to skip). This cycles through all active Vek in list order, move0, attack0, move1, attack1, move2, attack2, giving 2N sub-steps per turn for N active Vek units. Each sub-step is one environment step from the PPO perspective, so reward is sparse and only computed and assigned after all Vek have acted and the full turn cycle has resolved.

Once all Vek decisions have been issued, the environment sends a VEK_DONE command, which triggers the frozen player policy to run its full turn deterministically (with its own GRU hidden state reset each turn), followed by automatic resolution of the ENVIRONMENT, ENEMY_EXECUTE, and CLEANUP game phases. The adversary reward is then computed over the complete cycle.

Strategy Classification and Experience Routing

After each player turn completes, the environment classifies the player's strategy using the frozen VQ-VAE transformer (Model 2.2). The full token sequence accumulated across all player turns so far is packed into a padded (1, 52) batch and passed through the transformer in inference mode. The transformer returns the VQ codebook index assigned to each player turn, which equates to the strategy classification (e.g., VQ index 0 = strategy 0). The strategy for the most recent turn (the last entry in turn_strategy with a valid index ≥ 0) is extracted and used to select the adversary head for the next adversary turn. Routing uses the raw VQ codebook argmin instead of the variant classification head, directly reflecting which codebook cluster the player's MOVE and ATTACK tokens fell into.

This routing decision governs both action selection and experience storage. Each environment sub-step transition (observation, action, reward, value, log-probability, done flag, and action mask) is written to the rollout buffer of the head corresponding to the current strategy. When a particular strategy has not appeared during a rollout window, its buffer accumulates no transitions and its weights are not updated.

Network Architecture

Each of the five strategy heads is an independent policy/value network with a two-layer MLP trunk (hidden sizes 512, 512, Tanh activations) projecting to separate actor and critic heads. Importantly, no GRU is used in the adversary heads. Strategy routing via the transformer replaces episodic memory. Each head has its own Adam optimizer. All five heads' weights and optimizer states are saved together in a single checkpoint file.

Observation Space (2,372 features)

The adversary observation vector follows the same layout as the player (Model 1) but omits vek_intents (384 elements), since the adversary controls the Vek and therefore sets its own intents instead of observing them. The phase field encodes the current Vek sub-decision context (0.0 = vek_move, 1.0 = vek_attack) and the slot field encodes the index of the current Vek unit in the queue (0.0, 1.0, 2.0). The prev_action field carries the encoding of the player's most recent action, giving the adversary awareness of what the player just did.

Reward Function

The reward is computed from the adversary's perspective after each full turn cycle using the reward components shown in Table 3-15.

Table 3-15. Model 3 reward components.

Event	Weight
Per HP lost by any player mech	+0.4
Per player mech killed	+0.4
Per Vek killed by player	−0.1
Per building destroyed	+0.5
Player loses (all mechs dead or grid power = 0)	+5.0
Player wins (survives 5 turns)	−1.0

Training Configuration

The adversary is trained for 4,000,000 total adversary sub-steps, with rollout windows of 2,048 sub-steps. PPO updates are applied independently per head; a head is only updated when its buffer has accumulated at least 256 transitions. Up to 4 update epochs are run per head per rollout window, with early exit if the mean KL divergence within an epoch exceeds 0.05. The learning rate starts at 3×10^{-4} and decays linearly to zero. GAE is used for advantage estimation ($\gamma = 0.99$, $\lambda = 0.95$), with clipping parameter $\varepsilon = 0.2$ and gradient clipping at 0.5.

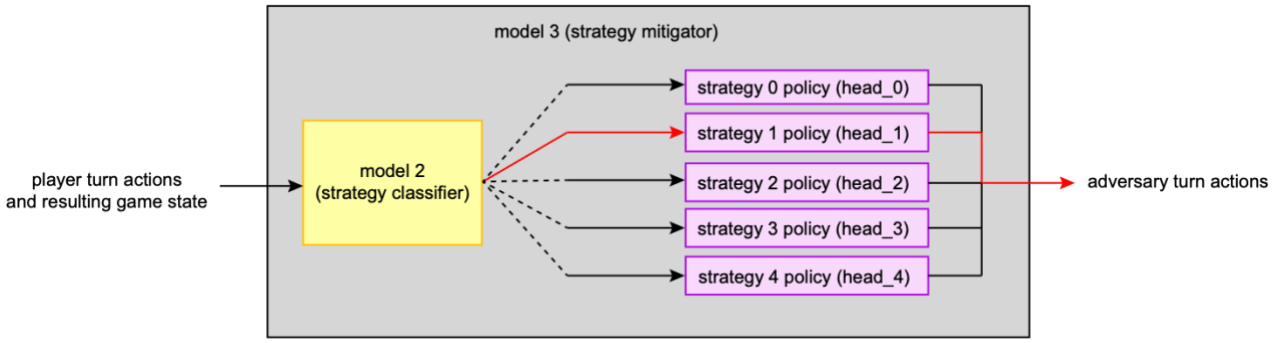


Figure 3-7. Model 3 with policy routing for each turn using Model 2.

CHAPTER 4. RESULTS

Results by Model 2 Variant

Model 2.1 – Spatial Strategy Discovery via Action Reconstruction

Model 2.1's codebook converged on five distinct and fully populated strategy clusters (S0–S4) (Figure 4-1), each capturing a spatially differentiated behavioral pattern. Attack target density (Figure 4-2) and move destination density (Figure 4-3) heatmaps reveal consistent spatial signatures across clusters. S0 concentrates activity in the upper board, S1 favors the right edge, S2 exhibits a tightly compressed lower-center pattern with reduced mech spread and mobility, S3 demonstrates a wide left-flank distribution, and S4 presents a center-anchored generalist spread. These spatial signatures are consistent across both move and attack heatmaps, indicating that the clusters capture genuine positional tendencies instead of artifacts of a single action type.

The spatial differentiation observed across clusters in Model 2.1 is consistent with the demands of the action reconstruction objective, which requires the encoder to preserve tile-level positional information to minimize reconstruction loss. However, whether this objective is the primary driver of the spatial organization, as opposed to spatial regularities inherent in the training data, cannot be determined from a single training run. No other differentiating patterns emerged in other metrics applied including mission length, ending grid power, total Vek kills per mission, and turn of first attack, which suggests the VQ-VAE codebook for Model 2.1 organizes turns by where actions occur on the board, instead of by their tactical consequences. The five clusters seem to represent five distinct spatial operating zones that the player's policy alternates between across different turns and missions.

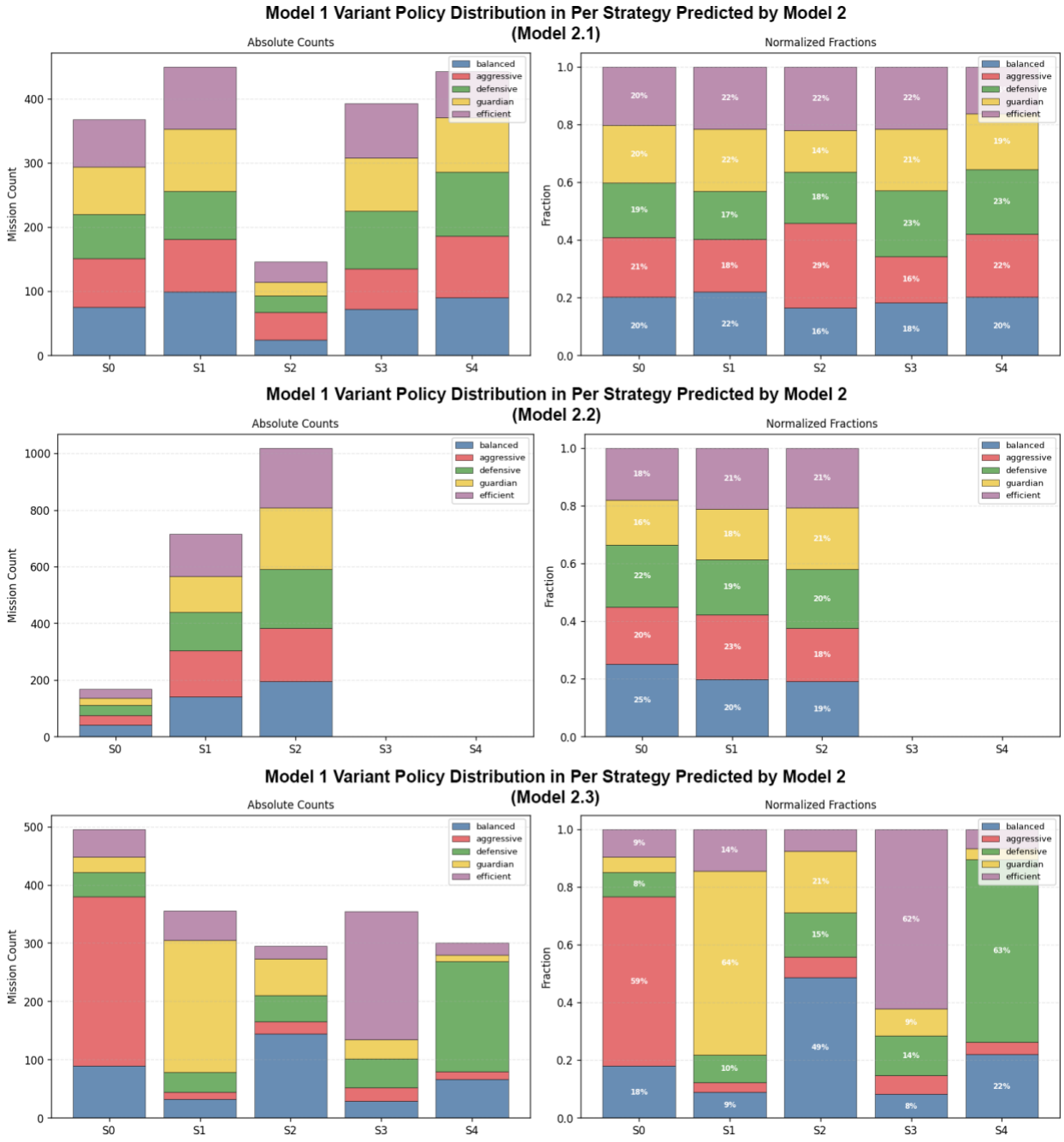


Figure 4-1. The policy variant distribution per strategy across Models 2.1, 2.2, and 2.3

The distribution of injected policy variants across all five clusters for Model 2.1 is near-uniform, with each of the five variants (balanced, aggressive, defensive, guardian, and efficient) contributing 14–29% of missions to each strategy cluster. This demonstrates that the spatial clusters cut across policy identity. The same named policy variant adopts all five spatial styles

across different map configurations, and the codebook captures this variation instead of recovering the hand-specified behavioral labels. This property is desirable for a strategy classifier intended to generalize to real-world players whose strategies are not known in advance.

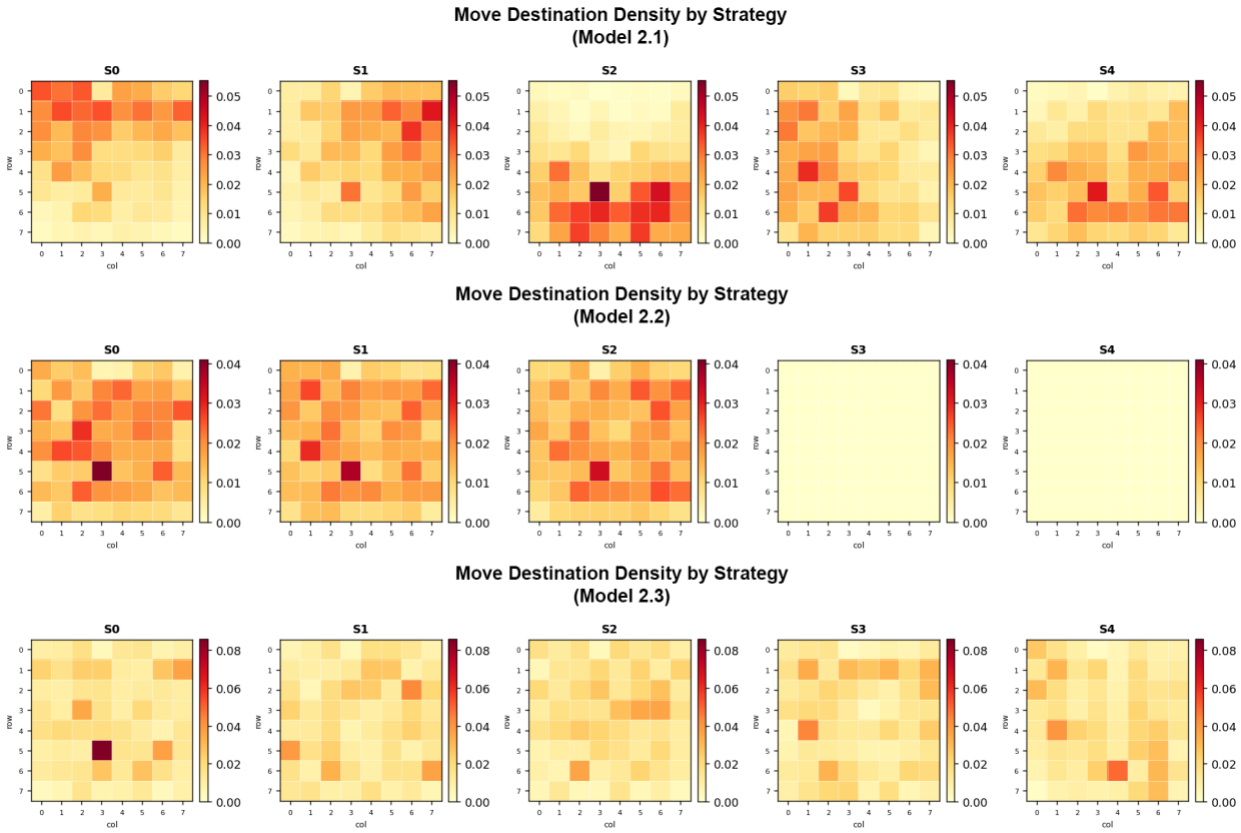


Figure 4-2. Gameboard attack target density by strategy across Models 2.1, 2.2, and 2.3

Model 2.2 – Outcome-Defined Strategy Discovery via Tactical Prediction

Codebook Utilization

Model 2.2's codebook converged on only three of five available strategy entries (Figure 4-1). Clusters S3 and S4 accumulated zero transitions across the full training and evaluation corpus (n = 0), indicating that the outcome-prediction objective produced a sparser representation than action reconstruction. The three active clusters vary substantially in size: S2 is the largest (n = 1,018 attack tokens), S1 is intermediate (n = 715), and S0 is the smallest (n =

167). This asymmetry suggests that the high-performance tactical turns captured by S0 are comparatively rare events within the training data.

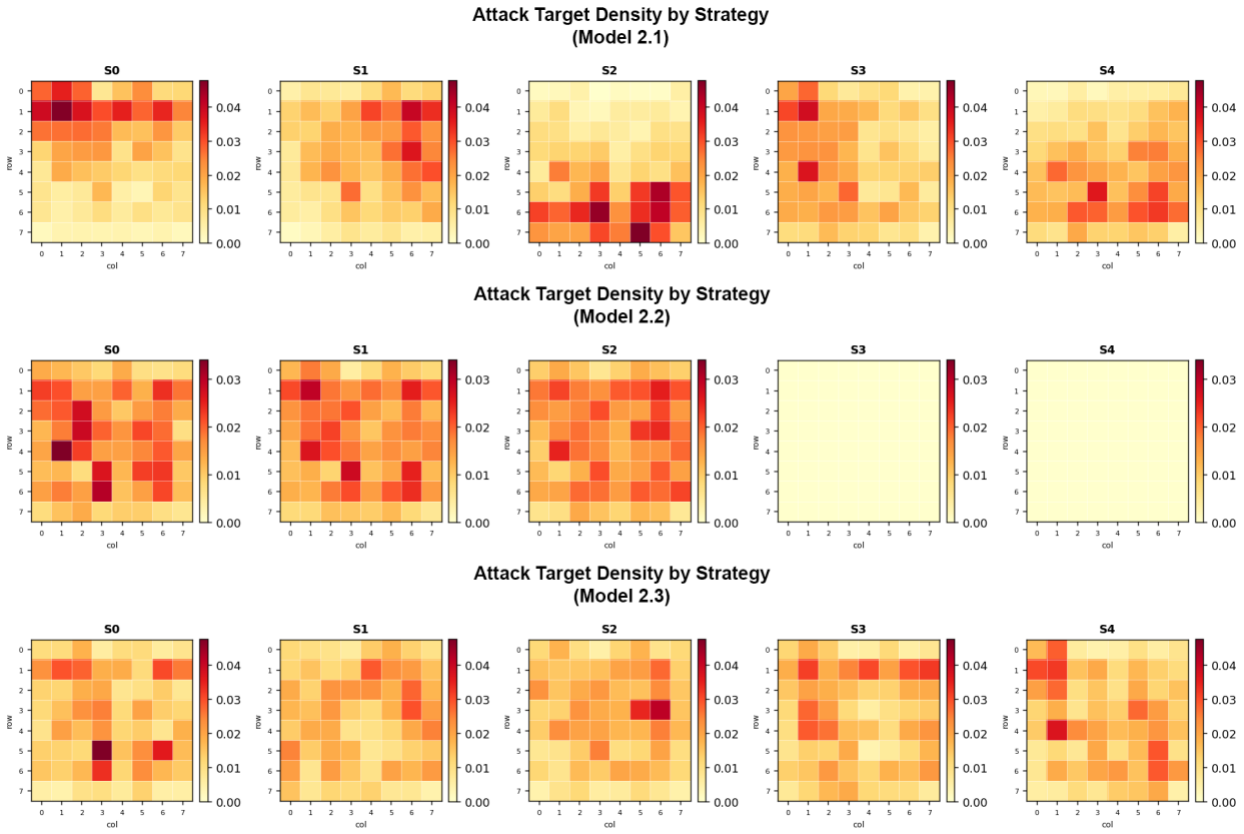


Figure 4-3. Gameboard move destination density by strategy across Models 2.1, 2.2, and 2.3

Differentiating Metrics Across Clusters

The three active clusters are differentiated along several combat outcome dimensions (Figure 4-4). Kill rate, the fraction of attacks resulting in at least one Vek eliminated, shows a strong monotonic gradient: S0 achieves the highest kill rate (0.255 ± SEM), followed by S1 (0.105), and S2 (0.022). Building-save rate, measuring turns on which a building-targeting Vek was killed or displaced, follows the same ordering: S0 (0.183), S1 (0.077), S2 (0.028). These two metrics together suggest that S0 captures high-lethality turns in which the player both eliminates threats and protects objectives.

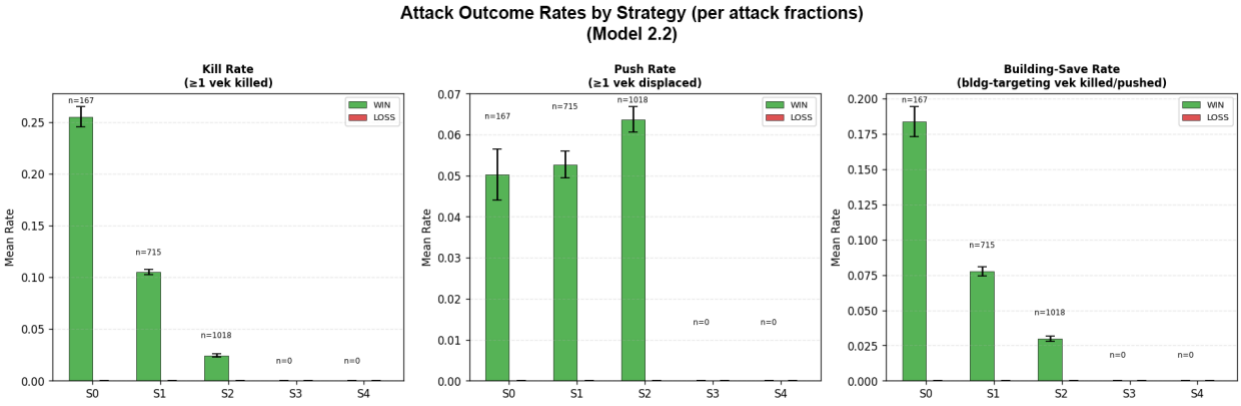


Figure 4-4. Attack outcomes by strategy per attack fractions for Model 2.2

Push rate, the fraction of attacks displacing at least one Vek without killing it, shows a reversed pattern: S2 achieves the highest push rate (0.064), followed by S1 (0.053) and S0 (0.050). This inversion suggests that S2 turns are characterized by displacement instead of elimination, consistent with a positioning or knockback-dominant style.

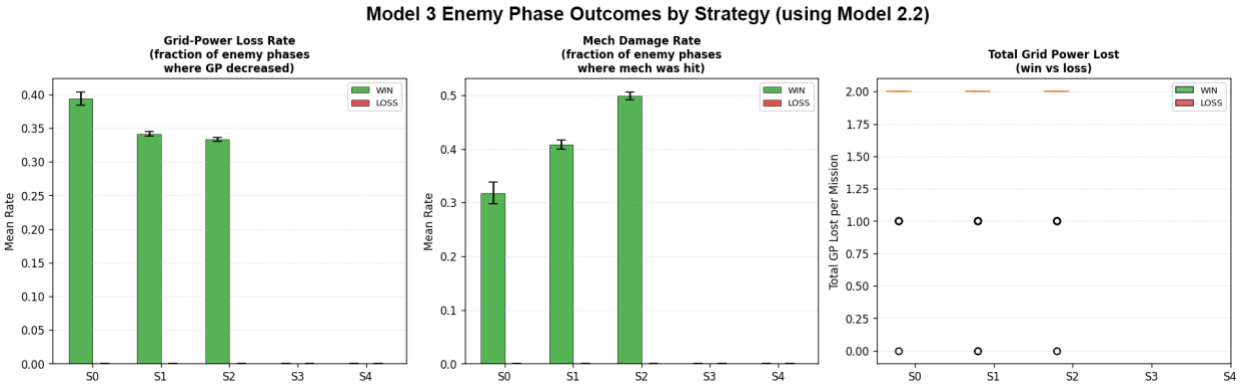


Figure 4-5. Enemy phase outcomes by strategy for Model 2.2

Enemy phase outcome metrics further reinforce this picture (Figure 4-5). Grid power loss rate, the fraction of enemy phases in which grid power decreases, is highest for S0 (0.395) and lower for S1 (0.340) and S2 (0.333). The difference between S1 and S2 on this metric is small, suggesting both clusters experience similar rates of objective damage despite their other differences. Mech damage rate, the fraction of enemy phases in which any mech sustains damage, follows the opposite ordering: S2 is highest (0.500), S1 intermediate (0.410), and S0

lowest (0.315). This indicates that high-lethality play in S0 is associated with better mech protection, while the low-kill S2 style leaves mechs more exposed.

Total grid power lost per mission shows no significant differentiation across the three clusters. In all cases the distribution is concentrated at 0 or 1 unit of grid power lost, with a ceiling at 2, and the spread is similar across S0, S1, and S2. This indicates that while the clusters differ substantially in their turn-level tactical profiles, they do not differ in the final quantity of grid power surrendered by the end of the mission, suggesting all three cluster styles are viable paths to mission success.

Variant Distribution

As with Model 2.1, the distribution of injected policy variants across the three active clusters remains near-uniform (Figure 4-1). Each of the five named variants contributes 16–25% of missions to each cluster, and no variant is meaningfully over-represented or under-represented in any strategy bucket. This supports the notion that the outcome-prediction objective, like action reconstruction, discovers structure that is orthogonal to the injected behavioral labels.

Model 2.3 – Supervised Strategy Alignment via Variant Prediction Loss

Model 2.3 is the only variant whose codebook aligns with the injected policy variant labels (Figure 4-1). All five clusters are populated, and the variant prediction loss produces a strong, non-uniform variant distribution across strategy slots. S0 is dominated by the aggressive policy (59%), S1 is dominated by the guardian policy (64%), S2 is dominated by the balanced policy (49%), S3 is dominated by the efficient policy (62%), and S4 is dominated by the defensive policy (63%). The remaining variants contribute minor fractions within each cluster. This policy-strategy alignment is a direct result of the auxiliary variant prediction loss (Equation 3-7), which provides an explicit supervision signal that encourages the codebook to partition the

latent space along variant identity instead of along spatial or outcome-based dimensions. Because gradients from the variant classification head flow back through the quantization step, specifically, because the variant head receives only the quantized embedding zq and not the skip-connected encoder output ze, the codebook entries themselves are directly shaped by the need to discriminate between variant styles. The result is a codebook that functions as a supervised variant classifier instead of an unsupervised strategy discoverer.

The spatial heatmaps for Model 2.3 are not well differentiated across clusters (Figure 4-2 and Figure 4-3), indicating that the action reconstruction task loss (Equation 3-3) imposes less positional structure on the learned representations.

Model 3 – Player Strategy Mitigation with Policy Routing

Model 3 was evaluated by playing against Model 1's balanced policy (Table 3-4) such that 30 episodes per condition per map were evaluated across all 40 winnable maps (20 easy, 20 difficult), for a total of 1,200 episodes per condition and 7,200 episodes overall. Six conditions were tested: five individual policy heads (head_0 through head_4), each used exclusively across all maps regardless of the player's observed strategy, and one routed condition in which the frozen VQ-VAE transformer (Model 2.2) classified the player's most recent completed turn each episode and dynamically selected the matching adversary head for that turn's enemy turn phase. The primary evaluation metric is adversary win rate, the fraction of episodes in which the adversary wins by either eliminating all three player mechs or reducing grid power to zero before the player survives five turns. Secondary metrics include mean grid power remaining at episode end, mean buildings destroyed per turn, mean mech HP lost by the player per episode, and mean Veks killed by the player per turn.

Win Rate

To assess statistical significance, a one-sided sign test was applied to the paired per-map win rate differences between the routed condition and each individual head across all 40 maps. The sign test evaluates whether the routed condition wins more often on a map-by-map basis than the comparison head, under the null hypothesis that neither condition is systematically favored ($p = 0.5$). The routed condition outperformed head_0 on 16 of 40 maps ($p = 0.006$), head_1 on 15 of 40 maps ($p = 0.010$), head_2 on 17 of 40 maps ($p = 0.004$), head_3 on 16 of 40 maps ($p = 0.002$), and head_4 on 18 of 40 maps ($p = 0.0002$). In no case did an individual head outperform the routed condition on more than four maps (Figure 4-6). All five comparisons are statistically significant at $\alpha = 0.01$, and four of five are significant at $\alpha = 0.005$, providing strong evidence that the routed condition's advantage is not attributable to chance.

A per-map comparison between the routed condition and the best-performing individual head on each map reveals a more nuanced picture (Figure 4-7). On 12 of 40 maps, the routed condition strictly exceeded the best single head, on 17 maps the conditions were approximately equal (most of these being maps where both conditions achieved zero wins), and on 11 maps, a single head achieved a higher win rate than routing. The mean win rate for the best-possible individual head selected post-hoc per map is 0.309, marginally above the routed mean of 0.280, indicating that routing does not yet reliably identify and apply the optimal head on every map. The routed condition achieved this performance without any knowledge of which head was best suited to each map.

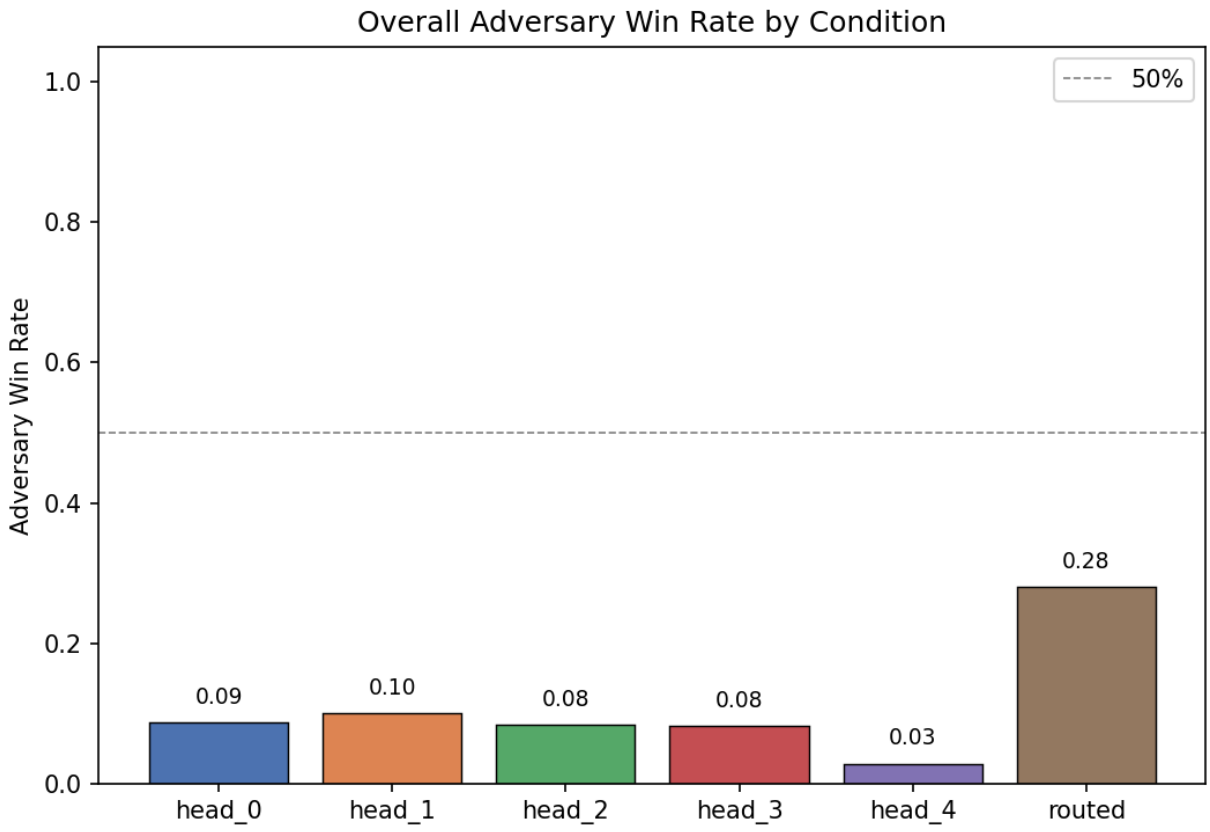


Figure 4-6. Overall adversary win rate by Model 3 policy head.

At the individual map level, the per-map adversary win rate heatmap (Figure 4-7) reveals notable map-specific patterns. The routed condition achieved nonzero win rates on 18 of 40 maps (10 easy, 8 difficult), compared to an aggregate of 22 maps with at least one nonzero individual head win rate. On three maps (easy_map_016, difficult_map_004, difficult_map_006), the routed condition was the only condition to achieve any wins. No individual head achieved a nonzero win rate on these maps. Conversely, on seven maps (easy_map_002, easy_map_009, easy_map_010, easy_map_014, easy_map_015, easy_map_018, and difficult_map_008), at least one individual head achieved positive win rates while the routed condition did not. This asymmetry suggests that routing occasionally selects a head poorly matched to the map's configuration, suppressing wins that a consistently applied head would have achieved.

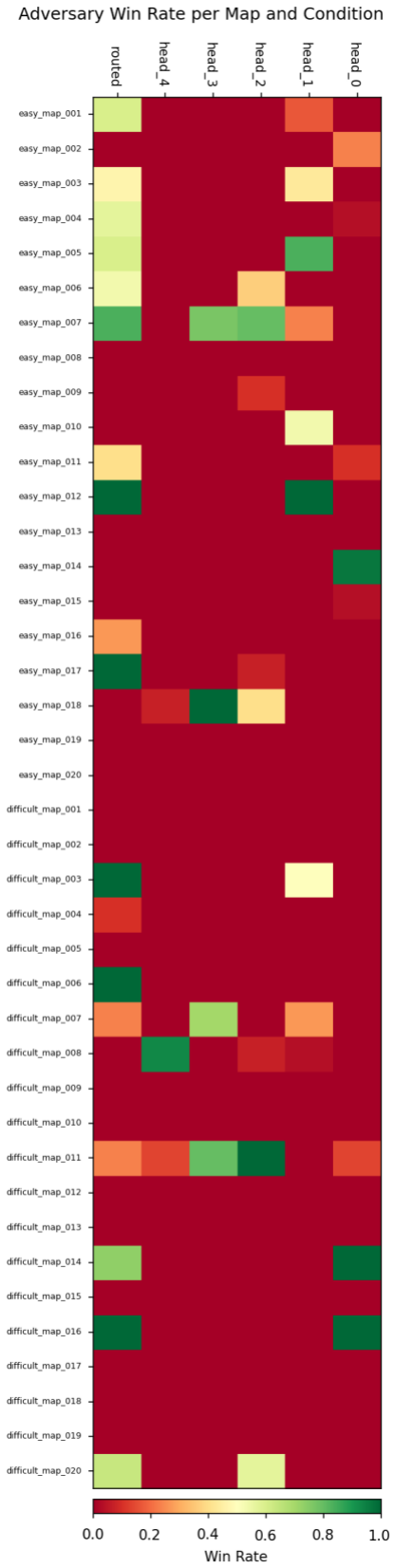


Figure 4-7. Win rate per map and condition by Model 3 policy head.

Secondary Metrics

Secondary metrics confirm that the routed condition applies substantially greater adversarial pressure on the player across multiple dimensions, not solely in outright wins. The mean grid power remaining at episode end (Figure 4-8) was 1.290 for the routed condition, the lowest of all conditions and 0.499 to 1.040 below the individual heads (head_0: 1.889, head_1: 1.788, head_2: 1.851, head_3: 2.329, head_4: 2.241). Lower residual grid power indicates that the routed adversary more consistently threatens the grid objective throughout the episode, even when not achieving an outright win. The mean grid power over turns plot (Figure 4-9) shows the routed condition's mean grid power declining most steeply across all five turns, reaching a mean of approximately 1.57 by turn five, compared to approximately 1.92 to 2.52 for the individual heads.

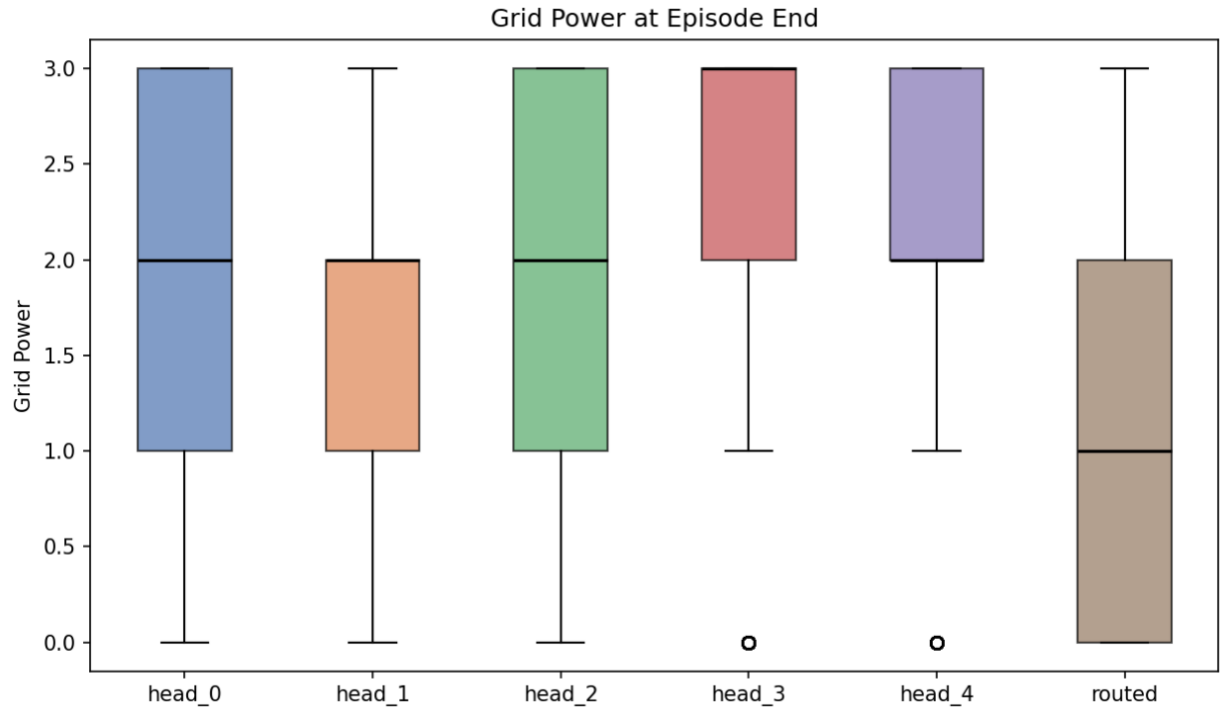


Figure 4-8. Grid Power at episode (game) end by Model 3 policy head.

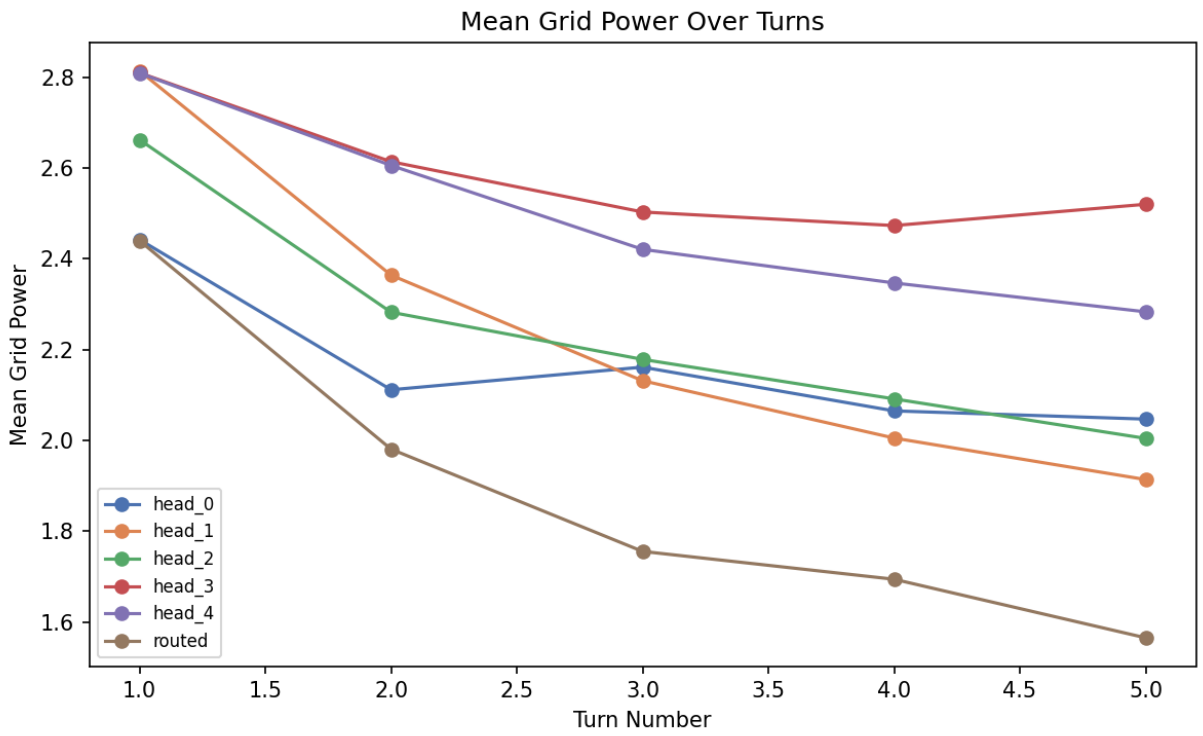


Figure 4-9. Mean grid power over turns by Model 3 policy head.

The mean number of buildings destroyed per turn (Figure 4-10) was 0.409 for the routed condition, the highest across all conditions and substantially above the individual heads (head_0: 0.273, head_1: 0.265, head_2: 0.274, head_3: 0.152, head_4: 0.162). This indicates that routing selects heads that more aggressively target buildings, the mechanism by which grid power is reduced, instead of prioritizing Vek survival or mech damage alone. The mean Mech HP lost by the player per episode (Figure 4-11) was 3.029 for the routed condition, the highest value observed and exceeding every individual head (ranging from 2.032 for head_4 to 2.920 for head_0). The elevated building destruction rate and mech damage suggest that the routed adversary maintains broader simultaneous pressure across both of the player's loss conditions (grid power depletion and mech elimination) instead of concentrating effort on one vector.

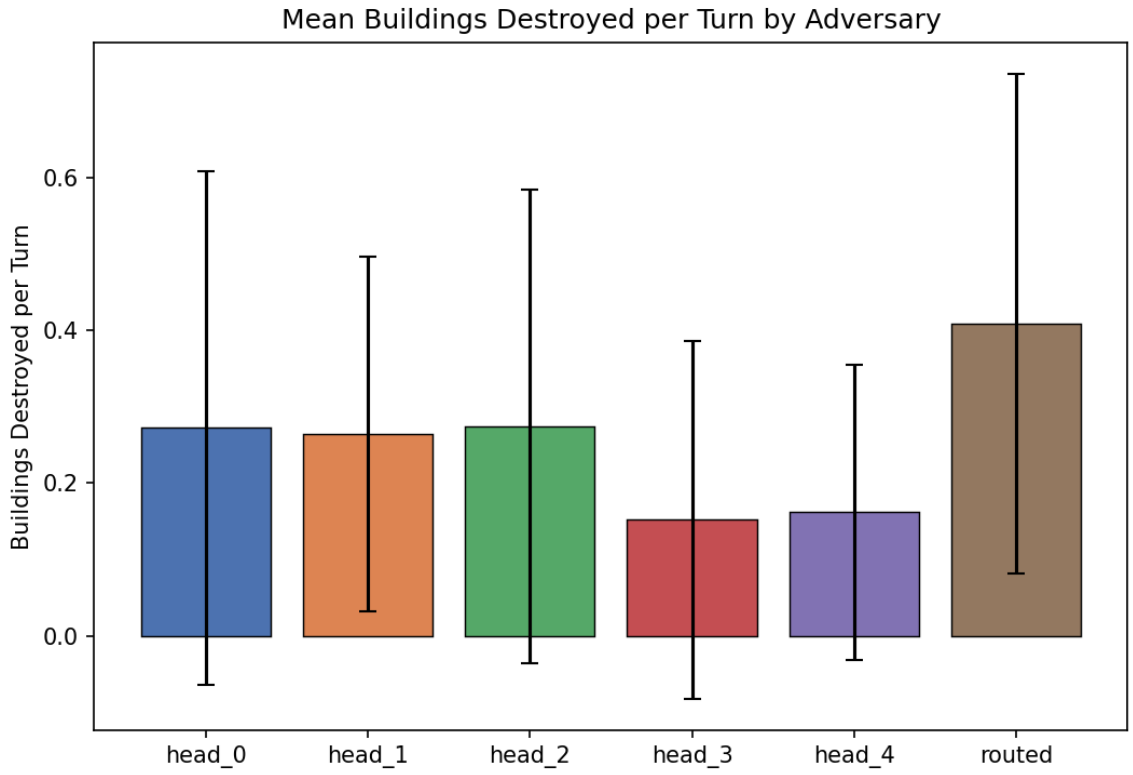


Figure 4-10. Mean buildings destroyed per turn by the adversary by Model 3 policy head.

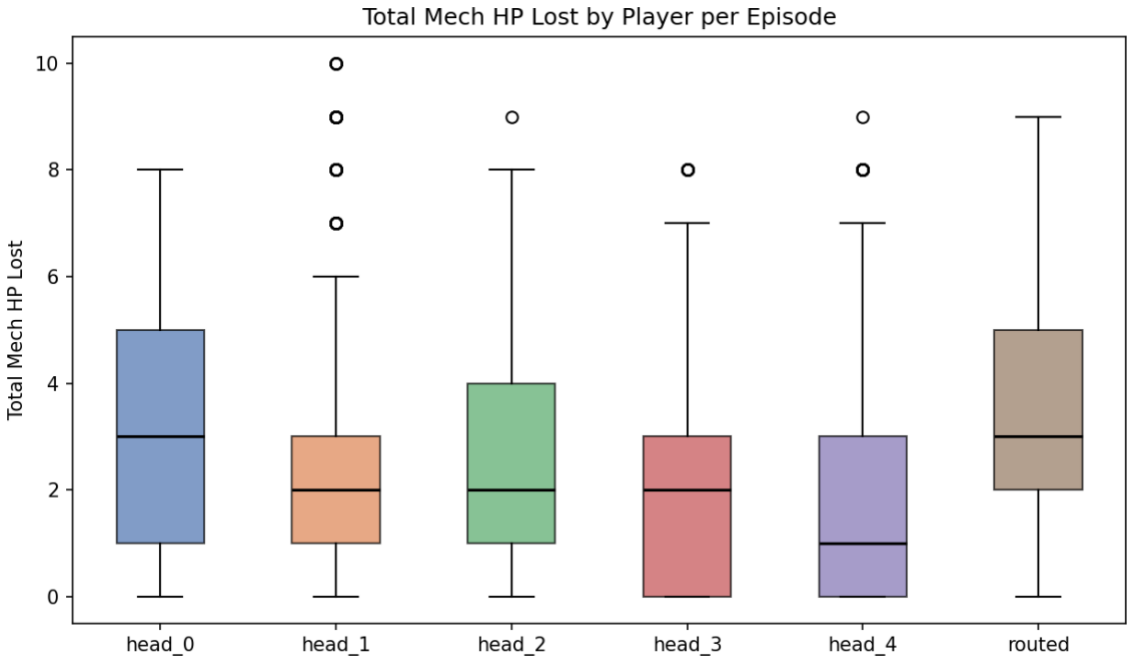


Figure 4-11. Total Mech HP lost by player per episode (game) by Model 3 policy head.

Mean Veks killed per turn (Figure 4-12) was 0.132 for the routed condition, approximately equal to or below several individual heads (head_1: 0.135, head_2: 0.166, head_3: 0.132, head_4: 0.147). The routed adversary therefore does not sacrifice Vek survival for increased aggression. Its Vek kill rate is comparable to individual heads, while its damage output is higher. This combination is consistent with the hypothesis that routing selects heads that place Vek in more dangerous but strategically productive positions, generating greater expected damage without exposing units to disproportionate countermeasure risk.

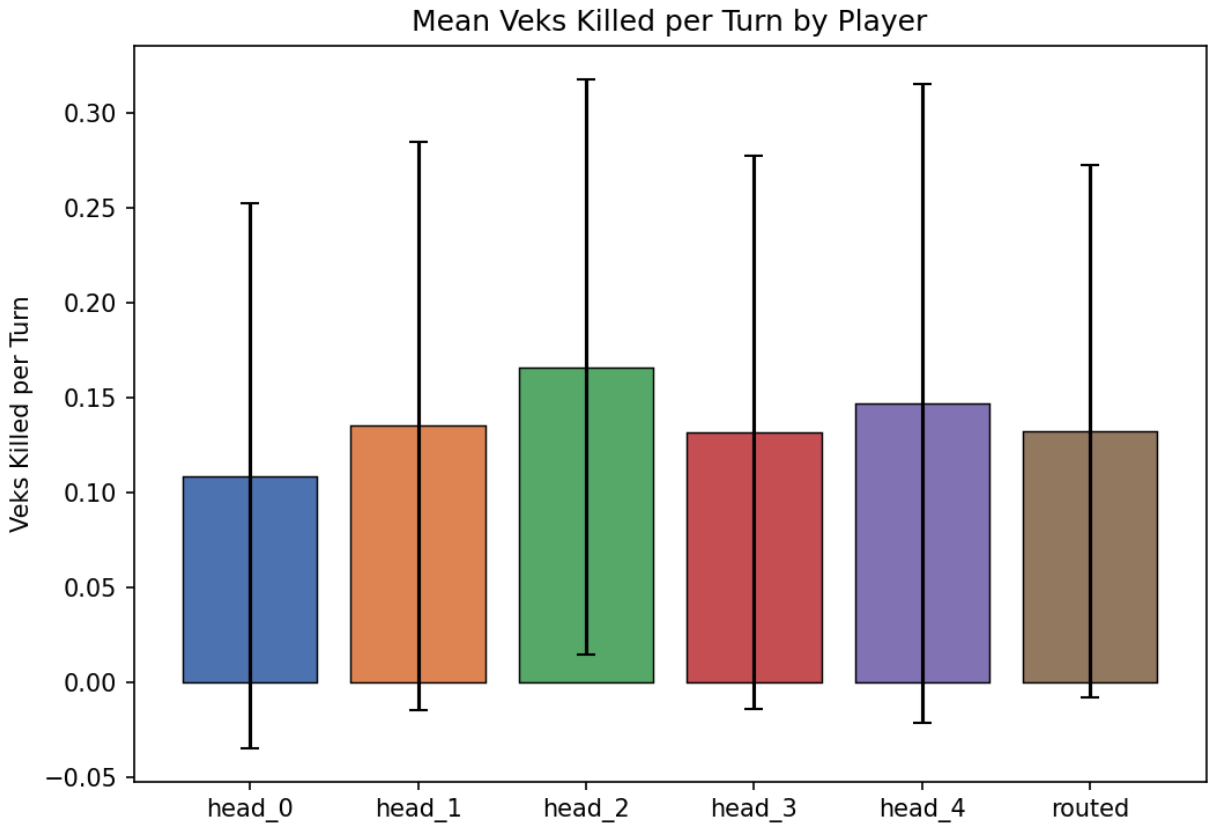


Figure 4-12. Mean number of Vek killed per turn by the player by Model 3 policy head.

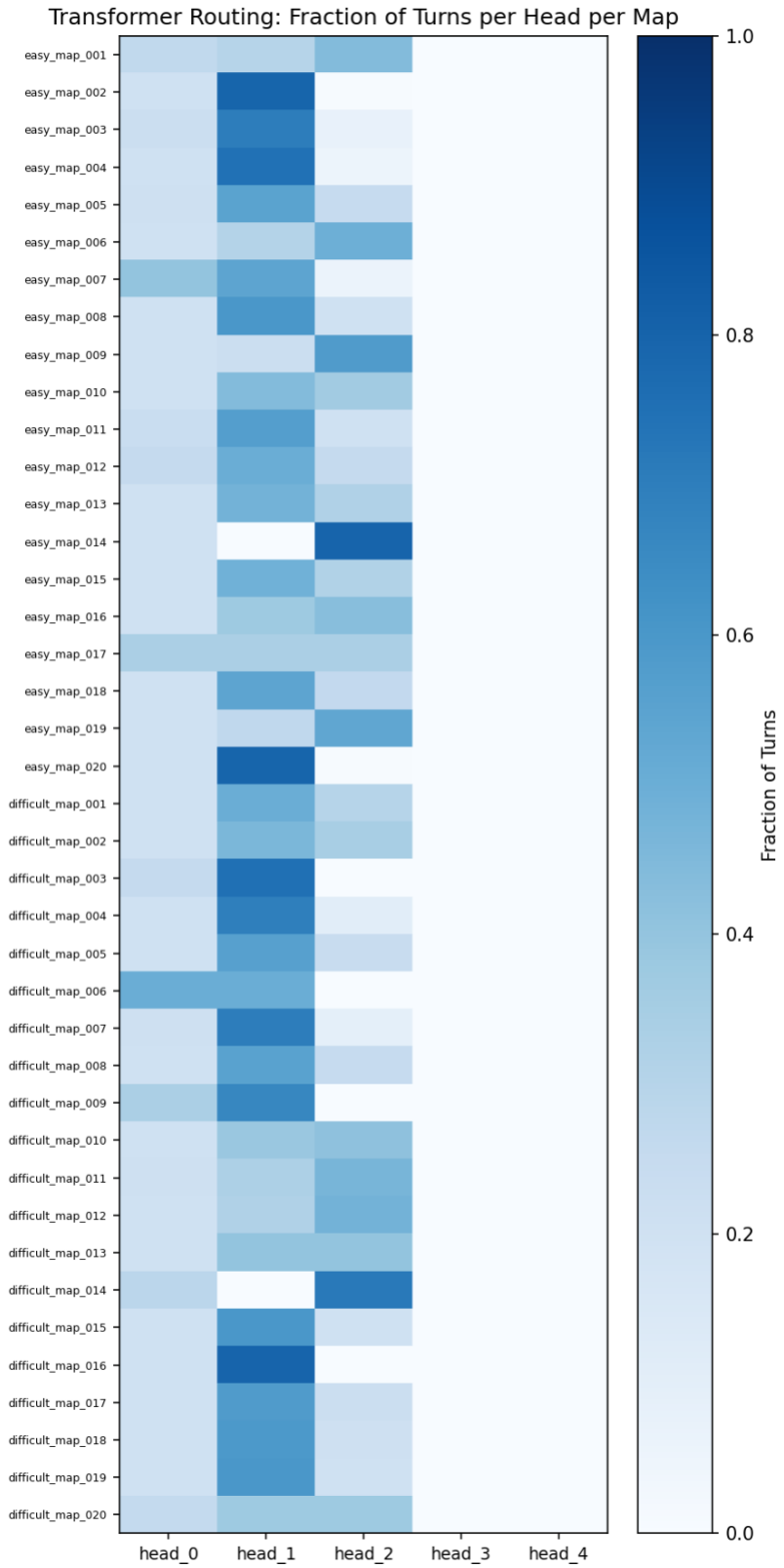


Figure 4-13. Model 2.2 routing distribution in fraction of turns per head per map.

Routing Distribution

The Model 2.2 routing distribution heatmap (Figure 4-13) reveals that the VQ-VAE classifier assigns the player's turns predominantly to heads 0, 1, and 2 across nearly all maps. Heads 3 and 4 receive near-zero routing fractions on the majority of maps, with only a small number of exceptions. This concentration is consistent with the codebook utilization patterns observed during VQ-VAE training, where certain codebook entries consistently attracted the majority of the player's turn-level embeddings (Figure 4-1). Within the maps where routing does occur, the allocation between heads 0, 1, and 2 varies substantially by map, with some maps dominated by a single head and others showing roughly equal contribution across two or three heads. On maps where head_1 receives a high routing fraction, the routed adversary tends to achieve win rates closer to head_1's individual performance, and similarly for heads 0 and 2. The low occurrence of routing to heads 3 and 4 is consistent with those heads' lower individual win rates, and suggests that the behavioral clusters they were assigned to represent player strategies that either appear infrequently in the evaluation data or were insufficiently represented in the adversary's training distribution to develop effective counter-policies.

Generalizing Over Other Policies

To evaluate whether the advantage of the routed condition extends beyond the training distribution (Model 3 was trained against the balanced opponent policy), Model 3 was also evaluated against four held-out policies, aggressive, defensive, efficient, and guardian, which were not seen during training. Each of the five individual heads (head_0 through head_4), along with the routed condition, was tested against each of these four policies across the full map set. The resulting adversary win rates are reported in Figure 4-14.

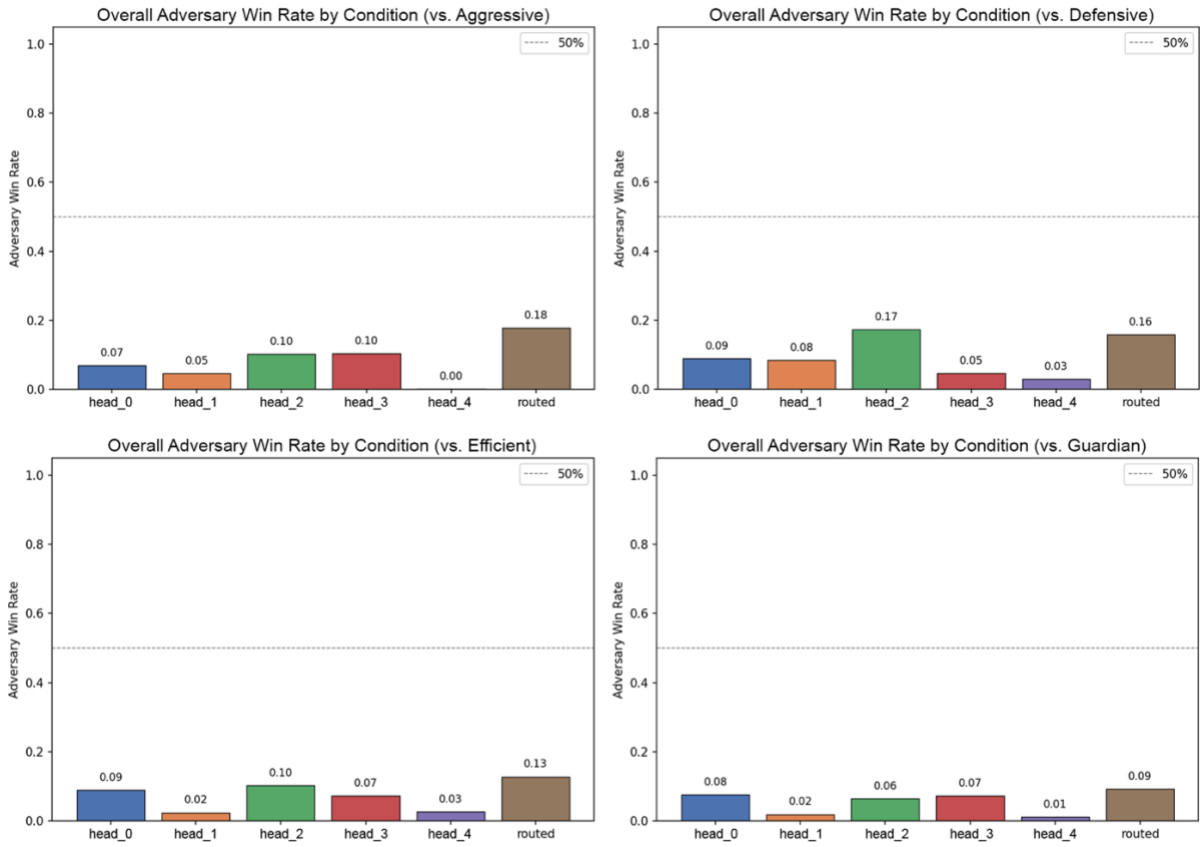


Figure 4-14. Overall adversary (Model 3) win rate by Model 3 policy head when playing against other player (Model 1) policies.

Across all four held-out policies, the routed condition achieved the highest or statistically comparable adversary win rate among the six conditions, indicating that the gains observed against the training policy are not an artifact of overfitting to the balanced opponent. Specifically, the routed condition reached an adversary win rate of 0.18 against aggressive, 0.16 against defensive, 0.13 against efficient, and 0.09 against guardian. In three of the four evaluations (aggressive, efficient, and guardian), the routed condition exceeded every individual head by a meaningful margin; against aggressive, for example, the routed rate of 0.18 was nearly double the best-performing individual head (head_2 and head_3, both at 0.10), and against efficient the routed rate of 0.13 exceeded the best individual head (head_2 at 0.10) by 30%. Against the guardian policy, the most difficult of the four, on which all conditions performed poorly, the

routed condition still matched or exceeded every individual head, reaching 0.09 compared to a maximum individual-head rate of 0.08 (head_0).

The one partial exception is the defensive policy, where head_2 achieved a win rate of 0.17, marginally exceeding the routed condition's 0.16. This single case suggests that for certain opponent types a specialized head may occasionally edge out the router, but the routed condition remained within 0.01 of the best individual head and substantially outperformed the remaining four heads (which ranged from 0.03 to 0.09). Importantly, no individual head dominated across all four held-out policies: head_2 was strongest against defensive and efficient but weak against guardian, while head_0 was strongest against guardian but middling elsewhere. Head_4, the weakest generalizer, failed to exceed a 0.03 win rate against any of the four held-out policies and achieved 0.00 against aggressive.

These results provide evidence that the routed condition's performance advantage generalizes beyond the training policy. Rather than collapsing to the strategy of any single head, the router appears to select effectively among specialized behaviors in response to opponent type, producing consistently strong, and in most cases best-in-class, performance against policies that were never encountered during training.

CHAPTER 5. DISCUSSION

The central finding of this work is that transformer-routed multi-head adversarial policy learning achieves substantially higher adversary win rates than any single policy trained in isolation, with statistical significance confirmed against all five individual heads. This result validates the three-model pipeline's core premise: the behavioral structure discovered unsupervised by Model 2.2 is actionable for adversary training in Model 3, enabling a single deployed system to respond to a wider range of player behaviors than any monolithic policy could. The routed adversary achieves this without access to ground-truth strategy labels during either the classifier's training or the adversary's training, relying entirely on the emergent codebook structure to partition experience and select heads at inference time.

Model 2 as a Strategy-Specific Mitigation Engine for Model 3

An important distinction between the routed system and a naive ensemble is that Model 2.2 does not just select a head at random or rotate through them uniformly, it identifies the player's active behavioral cluster in real time and routes to the head trained exclusively against that cluster. Each adversary head in Model 3 accumulated experience only from turns the classifier assigned to its corresponding codebook index during training, meaning its policy was shaped by counter-play against a specific behavioral regime, not against the full distribution of player behaviors. When Model 2.2 correctly identifies that the player is operating in, for example, the spatially distributed high-activity cluster associated with head_0's codebook entry, Model 3 deploys a head whose training distribution was that cluster's behavioral footprint. The result is targeted mitigation instead of averaged response. The adversary produces a policy specialized in countering the specific strategy the player is currently exhibiting, instead of a generalist policy tuned for the average across all strategies.

The three maps on which only the routed condition achieved positive win rates, while all five individual heads achieved zero, provide the clearest evidence of this emergent mitigation capability. On these maps, no single head's training distribution was sufficient to develop a winning counter in isolation, but the routing system's ability to dynamically shift which head is active as the player's turn-by-turn behavior evolves produced conditions under which at least one head's specialized knowledge could be applied at the right moment. This is exactly the scenario the pipeline was designed for: a player whose strategy varies across turns or across episodes, such that no fixed-head adversary can consistently exploit it, but whose moment-to-moment behavioral cluster can be detected in real time and met with a targeted response. The results support that Model 2's strategy classification provides genuine signal that translates into measurable adversarial gain for Model 3.

Feature Prediction Fidelity and Codebook Adherence

An additional observation that emerged from the training and analysis of Model 2 concerns the relationship between the accuracy of the decoder's feature predictions and the strength of encoder-to-codebook commitment for individual turns. In Models 2.1 and 2.3, the decoder predicts five specific features of each player action: action type (move or attack), acting mech identity, move tile, attack tile, and targeted Vek unit. Turns on which the decoder achieves low cross-entropy loss across these heads, meaning the player's action is behaviorally consistent and predictable from the surrounding game state context, tend to produce turn-level embeddings $\overline{z}_e$ that sit close to their assigned codebook entry z_q in L2 distance, yielding a

low commitment loss (Equation 3-2). Conversely, turns on which feature prediction accuracy is poor, such as atypical moves that do not conform to any stable spatial pattern,

produce embeddings that drift toward the boundary between codebook entries, weakening the assignment and increasing quantization distance.

This relationship is mechanistically implied by the joint optimization of the reconstruction and commitment losses during training. Both losses reward the encoder for producing embeddings that are simultaneously close to their assigned codebook entry and informative of the action features that the decoder must recover. However, to the best of our knowledge, the explicit correspondence between per-turn decoder reconstruction fidelity and per-turn codebook commitment strength has not been characterized in the VQ-VAE literature as a diagnostic tool. Prior work on VQ-VAE optimization has addressed codebook utilization and collapse at the population level (van den Oord et al., 2017; Razavi et al., 2019; Huh et al., 2023), but has not examined individual-sample reconstruction quality as a proxy for the confidence of the resulting codebook assignment. In the context of this system, this relationship is significant and implies that the reliability of Model 2's routing decisions for a given player turn is not uniform, but could be estimated from the decoder's reconstruction loss on that turn, without requiring access to ground-truth strategy labels.

Secondary Metric Interpretation and Head Utilization

The secondary metrics provide additional evidence that the routed condition generates qualitatively different adversarial pressure than individual heads. The routed adversary's lower residual grid power, higher buildings-per-turn rate, and higher mech HP damage collectively indicate that routing selects heads that pursue both win conditions more aggressively. Individual heads 3 and 4, which show higher residual grid power (2.329 and 2.241) and lower building destruction rates (0.152 and 0.162), appear to have converged on conservative strategies that prioritize Vek survival over tactical aggression, likely a consequence of their limited training

exposure to the player's behavioral clusters. The routing mechanism effectively bypasses these conservative heads on most maps, favoring heads 0, 1, and 2, which exhibit both higher win rates and more aggressive secondary metric profiles. This suggests that routing does not just ensemble the heads uniformly but implicitly selects for the more capable and better-trained policies by way of how the player's behavioral patterns align with the codebook.

The near-total absence of routing to heads 3 and 4 merits specific discussion. It is plausible that the codebook may not have fully differentiated all five strategy slots during training, as reflected in the lower decoder reconstruction fidelity for those clusters, leaving heads 3 and 4 without sufficient specialized training signal. The low individual win rates for heads 3 and 4 (0.082 and 0.028) and their near-zero routing fractions are consistent with this interpretation.

The overall adversary win rate of 0.280, while substantially above individual heads, remains well below 50%. The gap between adversarial pressure (as captured by secondary metrics) and outright wins points toward an adversary that imposes meaningful strategic stress but has not yet acquired the tactical precision to consistently deliver decisive outcomes. This is partly due to the game's design. The player's multi-phase turn structure, action masking, and three-mech formation provide robust defensive options that the three Vek units with limited move ranges must overcome. The reward shaping also incentivizes attrition (building destruction, mech damage) alongside winning, producing an adversary that may prefer sustained pressure over high-risk closing attacks. These limitations can be addressed through further training and reward tuning.

CHAPTER 6. FUTURE WORK

Several directions follow directly from the observations above. First, the commitment-fidelity relationship should be quantified empirically by correlating per-turn reconstruction cross-entropy with per-turn commitment loss across the evaluation dataset. If the correlation is strong, reconstruction loss can be used as a lightweight confidence score to implement soft routing. Instead of assigning each turn to a single head, the adversary would mix head outputs proportionally to the inverse of their commitment distance. This would reduce the cost of the misrouting events currently observed on eleven maps and is testable within the existing system architecture.

Second, model tuning could be explored starting with extended adversary training beyond 4,000,000 sub-steps with an adaptive schedule that re-weights experience toward underperforming heads. These updates could improve the tactical precision needed to convert accumulated pressure into wins. The current win rate of 0.280 suggests the adversary has learned to apply attrition but not to close out games consistently. Additional training under a reward schedule that more heavily penalizes failure to win after sustained damage may address this.

Finally, the planned human evaluation comparing the routed adversary against a fixed single-head baseline remains the most important validation of whether the quantitative performance improvements translate into perceived difficulty and strategic variety from a human player’s perspective, the ultimate criterion for an adaptive game opponent.

REFERENCES

Chollet, F. (2019). On the measure of intelligence. *arXiv*. https://arxiv.org/abs/1911.01547

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

Hou, B., Pan, G., & Chen, Y. (2026). Enhancing adversarial policy learning via value-based reward shaping. *Electronics, 15*, 463. https://doi.org/10.3390/electronics15020463

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)* (pp. 6379–6390). https://proceedings.neurips.cc/paper_files/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf

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

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

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

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv*. https://arxiv.org/abs/1707.06347

van den Oord, A., Vinyals, O., & Kavukcuoglu, K. (2017). Neural discrete representation learning. In *Advances in Neural Information Processing Systems (Vol. 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. In *Advances in Neural Information Processing Systems (Vol. 30)*. https://arxiv.org/abs/1706.03762

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., & 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)*. https://papers.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf

APPENDIX A.

Table A-6-1. Model 3 adversary evaluation results by condition and map.

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
head_0	easy_map_001	30	0.0	5.0	2.933	0.033	0.013	2.0
head_0	easy_map_002	30	0.2333	5.0	1.333	0.153	0.333	3.667
head_0	easy_map_003	30	0.0	5.0	2.567	0.12	0.087	3.767
head_0	easy_map_004	30	0.0333	4.9	1.067	0.047	0.417	0.033
head_0	easy_map_005	30	0.0	5.0	3.0	0.207	0.0	2.9
head_0	easy_map_006	30	0.0	5.0	1.4	0.153	0.32	2.367
head_0	easy_map_007	30	0.0	5.0	1.0	0.0	0.4	6.633
head_0	easy_map_008	30	0.0	5.0	2.0	0.4	0.2	1.467
head_0	easy_map_009	30	0.0	5.0	1.733	0.2	0.253	1.233
head_0	easy_map_010	30	0.0	5.0	2.567	0.2	0.087	2.833
head_0	easy_map_011	30	0.1	5.0	1.067	0.007	0.387	0.0
head_0	easy_map_012	30	0.0	5.0	2.967	0.0	0.007	5.0
head_0	easy_map_013	30	0.0	5.0	1.033	0.0	0.393	4.0
head_0	easy_map_014	30	0.9667	4.03	0.033	0.0	0.738	0.0
head_0	easy_map_015	30	0.0333	5.0	0.967	0.047	0.407	1.833
head_0	easy_map_016	30	0.0	5.0	2.0	0.0	0.2	4.567
head_0	easy_map_017	30	0.0	5.0	1.0	0.06	0.4	4.0
head_0	easy_map_018	30	0.0	5.0	1.967	0.0	0.207	4.367
head_0	easy_map_019	30	0.0	5.0	2.6	0.053	0.08	2.667

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
head_0	easy_map_020	30	0.0	5.0	1.567	0.327	0.287	1.233
head_0	difficult_map_001	30	0.0	4.97	2.667	0.112	0.067	2.5
head_0	difficult_map_002	30	0.0	5.0	3.0	0.0	0.0	6.3
head_0	difficult_map_003	30	0.0	5.0	2.0	0.253	0.2	4.0
head_0	difficult_map_004	30	0.0	5.0	2.067	0.0	0.187	0.6
head_0	difficult_map_005	30	0.0	5.0	2.967	0.0	0.007	3.267
head_0	difficult_map_006	30	0.0	5.0	1.0	0.4	0.4	0.0
head_0	difficult_map_007	30	0.0	5.0	2.367	0.287	0.127	1.533
head_0	difficult_map_008	30	0.0	5.0	1.967	0.0	0.207	5.133
head_0	difficult_map_009	30	0.0	5.0	2.0	0.32	0.2	5.333
head_0	difficult_map_010	30	0.0	5.0	2.9	0.033	0.02	2.133
head_0	difficult_map_011	30	0.1333	4.8	0.867	0.1	0.467	4.667
head_0	difficult_map_012	30	0.0	5.0	2.867	0.087	0.027	1.767
head_0	difficult_map_013	30	0.0	5.0	2.7	0.14	0.06	4.3
head_0	difficult_map_014	30	1.0	2.0	0.0	0.0	1.5	0.0
head_0	difficult_map_015	30	0.0	5.0	2.6	0.147	0.08	4.567
head_0	difficult_map_016	30	1.0	2.0	0.0	0.0	1.5	5.0
head_0	difficult_map_017	30	0.0	5.0	3.0	0.153	0.0	0.833
head_0	difficult_map_018	30	0.0	5.0	1.8	0.273	0.24	7.5
head_0	difficult_map_019	30	0.0	5.0	2.0	0.0	0.2	2.6
head_0	difficult_map_020	30	0.0	5.0	2.0	0.04	0.2	0.2

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
head_1	easy_map_001	30	0.1667	4.67	1.567	0.173	0.353	2.233
head_1	easy_map_002	30	0.0	5.0	1.7	0.147	0.26	1.333
head_1	easy_map_003	30	0.4333	4.57	0.567	0.228	0.552	0.0
head_1	easy_map_004	30	0.0	5.0	2.733	0.3	0.053	2.6
head_1	easy_map_005	30	0.8333	3.5	0.167	0.111	0.858	2.0
head_1	easy_map_006	30	0.0	5.0	2.367	0.033	0.127	2.867
head_1	easy_map_007	30	0.2333	4.77	1.033	0.0	0.417	7.9
head_1	easy_map_008	30	0.0	5.0	1.0	0.2	0.4	3.0
head_1	easy_map_009	30	0.0	5.0	1.867	0.047	0.227	0.0
head_1	easy_map_010	30	0.5333	5.0	0.567	0.127	0.487	1.067
head_1	easy_map_011	30	0.0	5.0	2.333	0.273	0.133	0.133
head_1	easy_map_012	30	1.0	3.0	0.0	0.011	1.0	2.0
head_1	easy_map_013	30	0.0	5.0	2.2	0.04	0.16	1.3
head_1	easy_map_014	30	0.0	5.0	2.0	0.16	0.2	3.0
head_1	easy_map_015	30	0.0	5.0	1.133	0.247	0.373	1.267
head_1	easy_map_016	30	0.0	5.0	2.733	0.353	0.053	0.0
head_1	easy_map_017	30	0.0	5.0	2.0	0.073	0.2	3.367
head_1	easy_map_018	30	0.0	5.0	2.0	0.0	0.2	1.467
head_1	easy_map_019	30	0.0	5.0	2.333	0.26	0.133	2.833
head_1	easy_map_020	30	0.0	5.0	2.367	0.027	0.127	2.5
head_1	difficult_map_001	30	0.0	5.0	2.467	0.007	0.107	1.6

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
head_1	difficult_map_002	30	0.0	5.0	2.0	0.4	0.2	6.0
head_1	difficult_map_003	30	0.5	5.0	0.967	0.093	0.407	2.033
head_1	difficult_map_004	30	0.0	5.0	1.767	0.1	0.247	3.133
head_1	difficult_map_005	30	0.0	5.0	3.0	0.113	0.0	0.867
head_1	difficult_map_006	30	0.0	5.0	2.0	0.2	0.2	1.167
head_1	difficult_map_007	30	0.2667	4.7	1.433	0.312	0.358	0.4
head_1	difficult_map_008	30	0.0333	5.0	1.8	0.0	0.24	2.4
head_1	difficult_map_009	30	0.0	5.0	2.233	0.127	0.153	2.7
head_1	difficult_map_010	30	0.0	5.0	2.067	0.24	0.187	1.133
head_1	difficult_map_011	30	0.0	5.0	1.967	0.107	0.207	3.233
head_1	difficult_map_012	30	0.0	5.0	1.633	0.333	0.273	2.2
head_1	difficult_map_013	30	0.0	5.0	2.333	0.2	0.133	5.667
head_1	difficult_map_014	30	0.0	5.0	1.067	0.0	0.387	1.867
head_1	difficult_map_015	30	0.0	5.0	3.0	0.06	0.0	2.433
head_1	difficult_map_016	30	0.0	5.0	1.0	0.027	0.4	9.0
head_1	difficult_map_017	30	0.0	5.0	2.233	0.013	0.153	0.633
head_1	difficult_map_018	30	0.0	5.0	2.267	0.147	0.147	7.2
head_1	difficult_map_019	30	0.0	5.0	1.633	0.113	0.273	0.067
head_1	difficult_map_020	30	0.0	5.0	1.967	0.0	0.207	0.8
head_2	easy_map_001	30	0.0	5.0	2.2	0.247	0.16	0.733
head_2	easy_map_002	30	0.0	4.93	2.7	0.297	0.06	3.633

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
head_2	easy_map_003	30	0.0	5.0	1.567	0.067	0.287	1.267
head_2	easy_map_004	30	0.0	5.0	3.0	0.34	0.0	1.533
head_2	easy_map_005	30	0.0	5.0	1.967	0.14	0.207	1.6
head_2	easy_map_006	30	0.3667	4.63	0.633	0.02	0.528	2.867
head_2	easy_map_007	30	0.8	2.6	0.2	0.0	1.28	3.4
head_2	easy_map_008	30	0.0	5.0	1.0	0.4	0.4	1.0
head_2	easy_map_009	30	0.1	5.0	1.3	0.2	0.34	5.867
head_2	easy_map_010	30	0.0	5.0	2.0	0.067	0.2	1.2
head_2	easy_map_011	30	0.0	5.0	1.067	0.253	0.387	0.133
head_2	easy_map_012	30	0.0	5.0	1.9	0.007	0.22	2.0
head_2	easy_map_013	30	0.0	5.0	2.333	0.013	0.133	2.4
head_2	easy_map_014	30	0.0	5.0	1.8	0.42	0.24	1.0
head_2	easy_map_015	30	0.0	5.0	1.833	0.0	0.233	1.333
head_2	easy_map_016	30	0.0	5.0	2.833	0.013	0.033	4.6
head_2	easy_map_017	30	0.0667	4.93	1.067	0.213	0.397	2.233
head_2	easy_map_018	30	0.4	4.57	0.6	0.007	0.558	0.2
head_2	easy_map_019	30	0.0	5.0	1.933	0.047	0.213	4.5
head_2	easy_map_020	30	0.0	5.0	1.0	0.173	0.4	1.767
head_2	difficult_map_001	30	0.0	5.0	1.4	0.213	0.32	1.767
head_2	difficult_map_002	30	0.0	5.0	3.0	0.353	0.0	4.333
head_2	difficult_map_003	30	0.0	5.0	2.0	0.36	0.2	4.367

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
head_2	difficult_map_004	30	0.0	5.0	2.367	0.2	0.127	4.267
head_2	difficult_map_005	30	0.0	5.0	2.733	0.18	0.053	0.033
head_2	difficult_map_006	30	0.0	5.0	1.733	0.2	0.253	0.067
head_2	difficult_map_007	30	0.0	5.0	2.1	0.3	0.18	1.567
head_2	difficult_map_008	30	0.0667	5.0	2.133	0.08	0.173	6.967
head_2	difficult_map_009	30	0.0	5.0	3.0	0.073	0.0	3.867
head_2	difficult_map_010	30	0.0	5.0	2.0	0.087	0.2	2.033
head_2	difficult_map_011	30	1.0	3.0	0.0	0.067	1.0	3.867
head_2	difficult_map_012	30	0.0	5.0	2.633	0.187	0.073	1.933
head_2	difficult_map_013	30	0.0	5.0	2.033	0.0	0.193	0.867
head_2	difficult_map_014	30	0.0	5.0	1.933	0.36	0.213	1.2
head_2	difficult_map_015	30	0.0	5.0	2.233	0.2	0.153	4.3
head_2	difficult_map_016	30	0.0	5.0	3.0	0.2	0.0	4.0
head_2	difficult_map_017	30	0.0	5.0	2.267	0.22	0.147	0.767
head_2	difficult_map_018	30	0.0	5.0	1.4	0.2	0.32	6.067
head_2	difficult_map_019	30	0.0	5.0	2.433	0.2	0.113	3.0
head_2	difficult_map_020	30	0.5667	3.3	0.7	0.04	0.97	2.2
head_3	easy_map_001	30	0.0	5.0	2.867	0.293	0.027	0.467
head_3	easy_map_002	30	0.0	5.0	2.433	0.133	0.113	0.133
head_3	easy_map_003	30	0.0	5.0	2.933	0.173	0.013	2.767
head_3	easy_map_004	30	0.0	5.0	2.167	0.327	0.167	4.333

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
head_3	easy_map_005	30	0.0	5.0	3.0	0.373	0.0	1.867
head_3	easy_map_006	30	0.0	5.0	2.0	0.153	0.2	0.6
head_3	easy_map_007	30	0.7667	3.47	0.233	0.0	0.86	7.233
head_3	easy_map_008	30	0.0	5.0	3.0	0.0	0.0	3.0
head_3	easy_map_009	30	0.0	5.0	1.8	0.067	0.24	0.4
head_3	easy_map_010	30	0.0	5.0	2.4	0.273	0.12	3.6
head_3	easy_map_011	30	0.0	5.0	2.733	0.1	0.053	1.133
head_3	easy_map_012	30	0.0	5.0	2.167	0.0	0.167	2.0
head_3	easy_map_013	30	0.0	5.0	1.833	0.027	0.233	0.733
head_3	easy_map_014	30	0.0	5.0	3.0	0.4	0.0	0.0
head_3	easy_map_015	30	0.0	5.0	2.467	0.02	0.107	2.7
head_3	easy_map_016	30	0.0	5.0	3.0	0.127	0.0	0.033
head_3	easy_map_017	30	0.0	5.0	3.0	0.147	0.0	3.0
head_3	easy_map_018	30	1.0	3.7	0.0	0.0	0.825	0.433
head_3	easy_map_019	30	0.0	5.0	3.0	0.06	0.0	4.5
head_3	easy_map_020	30	0.0	5.0	2.133	0.2	0.173	1.467
head_3	difficult_map_001	30	0.0	5.0	3.0	0.187	0.0	0.0
head_3	difficult_map_002	30	0.0	5.0	2.0	0.0	0.2	6.0
head_3	difficult_map_003	30	0.0	5.0	1.3	0.2	0.34	3.0
head_3	difficult_map_004	30	0.0	5.0	2.9	0.02	0.02	3.0
head_3	difficult_map_005	30	0.0	5.0	2.867	0.007	0.027	0.433

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
head_3	difficult_map_006	30	0.0	5.0	3.0	0.4	0.0	1.0
head_3	difficult_map_007	30	0.7	4.37	0.3	0.232	0.635	0.867
head_3	difficult_map_008	30	0.0	5.0	2.8	0.14	0.04	3.033
head_3	difficult_map_009	30	0.0	5.0	2.0	0.08	0.2	4.467
head_3	difficult_map_010	30	0.0	5.0	3.0	0.0	0.0	2.5
head_3	difficult_map_011	30	0.8	4.37	0.2	0.04	0.655	3.767
head_3	difficult_map_012	30	0.0	5.0	2.8	0.24	0.04	2.033
head_3	difficult_map_013	30	0.0	5.0	2.433	0.013	0.113	6.1
head_3	difficult_map_014	30	0.0	5.0	3.0	0.28	0.0	0.067
head_3	difficult_map_015	30	0.0	5.0	2.0	0.0	0.2	3.1
head_3	difficult_map_016	30	0.0	5.0	3.0	0.0	0.0	4.0
head_3	difficult_map_017	30	0.0	5.0	3.0	0.073	0.0	0.4
head_3	difficult_map_018	30	0.0	5.0	2.7	0.253	0.06	7.1
head_3	difficult_map_019	30	0.0	5.0	1.7	0.187	0.26	1.133
head_3	difficult_map_020	30	0.0	5.0	3.0	0.04	0.0	2.033
head_4	easy_map_001	30	0.0	5.0	2.8	0.013	0.04	1.3
head_4	easy_map_002	30	0.0	5.0	1.533	0.207	0.293	5.933
head_4	easy_map_003	30	0.0	5.0	3.0	0.0	0.0	2.0
head_4	easy_map_004	30	0.0	5.0	2.667	0.067	0.067	1.033
head_4	easy_map_005	30	0.0	5.0	1.9	0.2	0.22	2.6
head_4	easy_map_006	30	0.0	5.0	3.0	0.207	0.0	0.1

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
head_4	easy_map_007	30	0.0	5.0	2.0	0.0	0.2	7.0
head_4	easy_map_008	30	0.0	4.23	3.0	0.678	0.0	0.8
head_4	easy_map_009	30	0.0	5.0	3.0	0.053	0.0	0.9
head_4	easy_map_010	30	0.0	5.0	2.067	0.173	0.187	2.433
head_4	easy_map_011	30	0.0	5.0	2.267	0.44	0.147	0.0
head_4	easy_map_012	30	0.0	5.0	2.133	0.04	0.173	0.333
head_4	easy_map_013	30	0.0	5.0	2.367	0.033	0.127	0.0
head_4	easy_map_014	30	0.0	5.0	1.733	0.073	0.253	2.3
head_4	easy_map_015	30	0.0	5.0	3.0	0.027	0.0	0.167
head_4	easy_map_016	30	0.0	5.0	2.0	0.013	0.2	1.1
head_4	easy_map_017	30	0.0	5.0	2.0	0.053	0.2	0.7
head_4	easy_map_018	30	0.0667	5.0	2.033	0.053	0.193	0.0
head_4	easy_map_019	30	0.0	5.0	2.1	0.3	0.18	1.333
head_4	easy_map_020	30	0.0	5.0	2.5	0.107	0.1	0.767
head_4	difficult_map_001	30	0.0	5.0	2.233	0.1	0.153	1.7
head_4	difficult_map_002	30	0.0	5.0	3.0	0.413	0.0	6.0
head_4	difficult_map_003	30	0.0	5.0	2.0	0.2	0.2	2.0
head_4	difficult_map_004	30	0.0	5.0	1.133	0.193	0.373	0.3
head_4	difficult_map_005	30	0.0	5.0	2.267	0.207	0.147	0.0
head_4	difficult_map_006	30	0.0	5.0	3.0	0.2	0.0	0.233
head_4	difficult_map_007	30	0.0	5.0	1.433	0.213	0.313	0.467

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
head_4	difficult_map_008	30	0.9333	3.13	0.067	0.194	0.985	5.733
head_4	difficult_map_009	30	0.0	5.0	1.8	0.36	0.24	4.8
head_4	difficult_map_010	30	0.0	5.0	3.0	0.02	0.0	2.867
head_4	difficult_map_011	30	0.1333	5.0	1.133	0.0	0.373	4.033
head_4	difficult_map_012	30	0.0	5.0	1.467	0.06	0.307	1.633
head_4	difficult_map_013	30	0.0	5.0	2.4	0.013	0.12	3.9
head_4	difficult_map_014	30	0.0	5.0	3.0	0.347	0.0	0.0
head_4	difficult_map_015	30	0.0	5.0	2.533	0.027	0.093	1.8
head_4	difficult_map_016	30	0.0	5.0	1.833	0.14	0.233	4.567
head_4	difficult_map_017	30	0.0	5.0	3.0	0.233	0.0	0.133
head_4	difficult_map_018	30	0.0	5.0	2.0	0.227	0.2	7.133
head_4	difficult_map_019	30	0.0	5.0	3.0	0.0	0.0	2.867
head_4	difficult_map_020	30	0.0	5.0	2.233	0.0	0.153	0.333
routed	easy_map_001	30	0.6	3.8	0.4	0.007	0.76	3.667
routed	easy_map_002	30	0.0	5.0	2.433	0.12	0.113	2.1
routed	easy_map_003	30	0.4667	4.47	0.6	0.167	0.587	2.1
routed	easy_map_004	30	0.5667	5.0	0.533	0.207	0.493	4.3
routed	easy_map_005	30	0.6	4.87	0.4	0.173	0.54	3.0
routed	easy_map_006	30	0.5333	5.0	0.8	0.053	0.44	3.8
routed	easy_map_007	30	0.8333	5.0	0.167	0.0	0.567	8.033
routed	easy_map_008	30	0.0	5.0	2.0	0.4	0.2	3.0

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
routed	easy_map_009	30	0.0	5.0	1.9	0.34	0.22	0.1
routed	easy_map_010	30	0.0	5.0	2.2	0.087	0.16	2.233
routed	easy_map_011	30	0.4	4.33	1.5	0.133	0.433	0.567
routed	easy_map_012	30	1.0	4.0	0.0	0.0	0.75	3.0
routed	easy_map_013	30	0.0	5.0	1.233	0.047	0.353	2.233
routed	easy_map_014	30	0.0	5.0	1.0	0.033	0.4	2.367
routed	easy_map_015	30	0.0	5.0	1.133	0.2	0.373	0.433
routed	easy_map_016	30	0.2667	5.0	0.733	0.0	0.453	3.133
routed	easy_map_017	30	1.0	3.0	0.0	0.0	1.0	4.0
routed	easy_map_018	30	0.0	5.0	1.3	0.02	0.34	4.267
routed	easy_map_019	30	0.0	5.0	2.1	0.06	0.18	3.967
routed	easy_map_020	30	0.0	5.0	1.533	0.247	0.293	0.367
routed	difficult_map_001	30	0.0	5.0	2.233	0.227	0.153	3.667
routed	difficult_map_002	30	0.0	5.0	2.3	0.187	0.14	5.0
routed	difficult_map_003	30	1.0	4.0	0.0	0.25	0.75	3.0
routed	difficult_map_004	30	0.1	5.0	1.433	0.02	0.313	0.867
routed	difficult_map_005	30	0.0	5.0	2.467	0.053	0.107	3.067
routed	difficult_map_006	30	1.0	2.0	0.0	0.5	1.5	0.0
routed	difficult_map_007	30	0.2333	4.77	1.267	0.16	0.382	2.233
routed	difficult_map_008	30	0.0	5.0	2.833	0.06	0.033	3.0
routed	difficult_map_009	30	0.0	5.0	2.0	0.207	0.2	5.2

Condition	Map	n	Win Rate	Mean Turns	Mean GP End	Mean Veks/Turn	Mean Blds/Turn	Mean Mech HP Lost
routed	difficult_map_010	30	0.0	5.0	1.233	0.2	0.353	2.0
routed	difficult_map_011	30	0.2333	4.8	0.767	0.0	0.477	4.633
routed	difficult_map_012	30	0.0	5.0	1.733	0.147	0.253	2.367
routed	difficult_map_013	30	0.0	5.0	2.1	0.167	0.18	6.667
routed	difficult_map_014	30	0.7333	3.53	0.267	0.0	0.84	0.0
routed	difficult_map_015	30	0.0	5.0	3.0	0.2	0.0	5.0
routed	difficult_map_016	30	1.0	5.0	0.0	0.067	0.6	8.0
routed	difficult_map_017	30	0.0	5.0	2.733	0.007	0.053	0.533
routed	difficult_map_018	30	0.0	5.0	1.667	0.24	0.267	7.167
routed	difficult_map_019	30	0.0	5.0	1.233	0.233	0.353	2.0
routed	difficult_map_020	30	0.6333	3.93	0.367	0.073	0.74	0.1