Policy–Value Learning for Magic: The Gathering Arena Cube Draft

Anonymous Authors[1]

Abstract

We develop and evaluate a learning system for Magic: The Gathering Arena cube draft recommendations from public 17lands data. The method separates the problem into a pick policy trained from human draft logs and outcome/value signals trained from game results. At inference, the policy supplies calibrated in-pack priors while value-derived terms rerank or fine-tune recommendations. The architecture is cube-list-conditioned, represents cards by learned and Scryfall-derived features, and uses only information observable to the drafting player: current pack, own pool, cards seen but not picked, pack index, pick index, and known cube list. On a held-out Powered Cube split, the strongest human-imitation model obtains top-1 accuracy 0.624 and top-3 accuracy 0.909. We further train game-data preference policies from learned card bonuses. A safer game-data policy improves generated preference accuracy from 0.549 to 0.641 while preserving top-1 0.616; an aggressive preference-accuracy policy reaches preference accuracy 0.726 but drops top-1 human agreement to 0.509. We also implement an experimental MTGA Player.log follower that reconstructs quick-draft pack and pick state from live client logs. Finally, we add an inference-time Monte Carlo search layer; with 25 simulations it gives live search-based reranking at roughly one-second median CPU latency and qualitatively produces more lane-consistent draft traces.

1. Problem

A cube draft consists of $S = 8$ seats, $P = 3$ packs per seat, and $K = 15$ cards per pack. At each decision t, the acting player observes a state

$$s_t = (x_t, y_t, z_t, m, p_t, k_t), \tag{1}$$

where $x_t \in \{0,1\}^C$ is the current pack mask over a cube-local vocabulary of size C, $y_t \in \mathbb{N}^C$ is the player's drafted pool, $z_t \in \mathbb{N}^C$ counts cards previously seen by the player but not selected, $m \in \{0,1\}^C$ is the known cube-list mask, and (p_t, k_t) are pack and pick indices. The action set is the current pack,

$$\mathcal{A}(s_t) = \{i : x_{t,i} = 1\}. \tag{2}$$

The objective is to recommend a pick $a_t \in \mathcal{A}(s_t)$ that maximizes expected downstream deck performance, while remaining robust to partial cube-list knowledge and unseen cards.

2. Data

We use three public 17lands tables for a target format such as `Cube - Powered`: draft data, game data, and replay data (17Lands, 2026). Draft rows contain one human pick per row: the selected card, current pack card indicators, draft identifier, pack number, pick number, rank, and event metadata. Game rows contain one played game per row: draft identifier, build index, deck and sideboard card indicators, colors, rank, play/draw, mulligans, opponent metadata, and win/loss. Replay rows contain turn-level play traces and are deferred to future play-skill modeling.

The core supervised datasets are:

$$\mathcal{D}_\pi = \{(s_t, a_t^{\text{human}})\}_{t=1}^{N}, \tag{3}$$

$$\mathcal{D}_V = \{(d_j, b_j, c_j, r_j, w_j)\}_{j=1}^{M}, \tag{4}$$

where d_j is a main-deck vector, b_j is a sideboard vector, c_j are color features, r_j are rank/event covariates, and $w_j \in \{0,1\}$ is the game outcome. Game rows are also aggregated by $(\texttt{draft\_id}, \texttt{build\_index})$ into empirical deck win rates.

3. Card Representation

Each card i has a learned global embedding and a feature-derived embedding:

$$e_i = E_i + \alpha g(f_i), \tag{5}$$

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

where $E_i \in \mathbb{R}^d$ is a learned embedding, f_i contains Scryfall-derived features (Scryfall, 2026) such as mana value, colors, type bits, rarity, power/toughness, keywords, and oracle-text embeddings, and g is an MLP. The scalar α is annealed from 1 to a positive floor, e.g. 0.3, so that known cards may specialize while unseen cards retain meaningful feature-based representations. Out-of-vocabulary cards share an `UNK` learned embedding but keep their own $g(f_i)$ term when features are available.

4. Pick Policy Model

The pick policy is a cube-context contextual preference ranker. Given card representations $\{e_i\}_{i=1}^{C}$, we encode the pool, seen-but-unpicked cards, and cube list with DeepSets:

$$h_y = \rho_y\left(\frac{\sum_i y_i \phi_y(e_i)}{\max(1, \sum_i y_i)}\right), \tag{6}$$

$$h_z = \rho_z\left(\frac{\sum_i z_i \phi_z(e_i)}{\max(1, \sum_i z_i)}\right), \tag{7}$$

$$h_m = \rho_m\left(\frac{\sum_i m_i \phi_m(e_i)}{\max(1, \sum_i m_i)}\right). \tag{8}$$

We also compute a color-commitment vector

$$q = \frac{\sum_i y_i \,\mathrm{color}(i)}{\max(1, \sum_i y_i)} \in \mathbb{R}^5. \tag{9}$$

The context vector is

$$h(s) = \mathrm{MLP}_\pi\left([h_y, h_z, h_m, q, u_p, v_k]\right), \tag{10}$$

where u_p and v_k are learned pack-index and pick-index embeddings. Card logits are

$$\ell_i(s) = h(s)^\top e_i + b(e_i), \tag{11}$$

with b a scalar bias head. The masked policy is

$$\pi_\theta(i \mid s) = \frac{\exp \ell_i(s)}{\sum_{j \in \mathcal{A}(s)} \exp \ell_j(s)}, \quad i \in \mathcal{A}(s). \tag{12}$$

4.1. Policy Loss

The default loss is masked cross-entropy on human picks:

$$\mathcal{L}_{\mathrm{CE}}(\theta) = -\mathbb{E}_{(s,a)\sim\mathcal{D}_\pi} \log \pi_\theta(a \mid s). \tag{13}$$

We also use a contextual preference ranking loss over unpicked pack cards:

$$\mathcal{L}_{\mathrm{CPR}}(\theta) = \mathbb{E}_{(s,a)} \frac{1}{|\mathcal{A}(s)| - 1} \sum_{j \in \mathcal{A}(s)\setminus\{a\}} \left[\gamma - \ell_a(s) + \ell_j(s)\right]_+. \tag{14}$$

The combined pick-imitation loss is

$$\mathcal{L}_\pi = \mathcal{L}_{\mathrm{CE}} + \beta_{\mathrm{CPR}} \mathcal{L}_{\mathrm{CPR}}. \tag{15}$$

5. Outcome Value Model

The value model predicts game win probability from a built deck:

$$V_\psi(d, b, c, r) \in (0, 1). \tag{16}$$

Main deck and sideboard are encoded by DeepSets over the same card representations:

$$h_d = \rho_d\left(\frac{\sum_i d_i \phi_d(e_i)}{\max(1, \sum_i d_i)}\right), \tag{17}$$

$$h_b = \rho_b\left(\frac{\sum_i b_i \phi_b(e_i)}{\max(1, \sum_i b_i)}\right), \tag{18}$$

$$V_\psi(d, b, c, r) = \sigma\left(\mathrm{MLP}_V([h_d, h_b, c, r])\right). \tag{19}$$

For per-game training the loss is binary cross-entropy:

$$\mathcal{L}_V(\psi) = -\mathbb{E}_{(d,b,c,r,w)\sim\mathcal{D}_V}\left[w \log V_\psi + (1 - w)\log(1 - V_\psi)\right]. \tag{20}$$

For deck-aggregate training, if a deck has n games and empirical win rate $\hat{p}$, we use the binomial negative log-likelihood

$$\mathcal{L}_V^{\mathrm{agg}} = -\mathbb{E}\left[n\hat{p} \log V_\psi + n(1 - \hat{p})\log(1 - V_\psi)\right]. \tag{21}$$

Rank, event type, and play/draw covariates can be included or ablated. For deployment scoring, we report both rank-conditioned predictions and rank-marginalized predictions.

6. Pool-to-Deck Builder

Because V_ψ scores built decks, not raw draft pools, a deck-builder B maps a drafted pool y and cube context m to a main deck and sideboard:

$$B(y, m) = (d, b, c). \tag{22}$$

The minimum viable B is a constrained optimizer that selects approximately 23 nonland spells plus lands, penalizes excessive colors, enforces curve constraints, and rewards fixing. A learned deckbuilder may later be trained from 17lands build rows. In algorithms below, B can be either heuristic or learned.

7. Combining Policy and Value

7.1. One-Step Value Reranking

For each legal pick a, construct the hypothetical pool $y^a = y + \mathbf{1}_a$, build a deck $(d^a, b^a, c^a) = B(y^a, m)$, and score

$$S(a \mid s) = \log \pi_\theta(a \mid s) + \lambda \,\mathrm{logit}\left(V_\psi(d^a, b^a, c^a, r_0)\right), \tag{23}$$

where r_0 is a reference rank/event context. The recommended pick is $\arg\max_{a\in\mathcal{A}(s)} S(a \mid s)$. Small λ recovers imitation behavior; larger λ emphasizes outcome optimization.

Algorithm 1 One-step policy–value reranking

Require: State $s = (x, y, z, m, p, k)$, policy π_θ, value V_ψ, deckbuilder B, weight λ

for $a \in \mathcal{A}(s)$ **do**
  $y^a \leftarrow y + \mathbf{1}_a$
  $(d^a, b^a, c^a) \leftarrow B(y^a, m)$
  $S(a) \leftarrow \log \pi_\theta(a \mid s) + \lambda \operatorname{logit}(V_\psi(d^a, b^a, c^a, r_0))$
end for
return $\arg\max_a S(a)$

Algorithm 2 Rollout policy–value decision

Require: State s, simulations R, policy π_θ, value V_ψ, deckbuilder B

for $a \in \mathcal{A}(s)$ **do**
  $q \leftarrow 0$
  for $r = 1$ to R **do**
    $Y_T \leftarrow \mathrm{Rollout}(s, a, \pi_\theta, \eta_r)$
    $(d, b, c) \leftarrow B(Y_T, m)$
    $q \leftarrow q + V_\psi(d, b, c, r_0)$
  end for
  $Q(a) \leftarrow q/R$
  $S(a) \leftarrow \log \pi_\theta(a \mid s) + \lambda \operatorname{logit}(Q(a))$
end for
return $\arg\max_a S(a)$

7.2. Rollout Evaluation

One-step reranking is myopic. A stronger method evaluates each candidate by simulating the remainder of the draft. Let $\mathrm{Rollout}(s, a, \pi_\theta, \eta)$ force action a at s, then complete the draft using policy π_θ for the player and opponent models for other seats, under randomness η. The rollout value is

$$Q(a \mid s) = \mathbb{E}_\eta \left[V_\psi \left(B(Y_T^{s,a,\eta}, m), r_0 \right) \right], \tag{24}$$

where Y_T is the terminal pool. We combine prior and rollout value:

$$S_R(a \mid s) = \log \pi_\theta(a \mid s) + \lambda \operatorname{logit}(Q(a \mid s)). \tag{25}$$

7.3. MCTS with Policy Priors

Monte Carlo tree search uses π_θ as the prior and $V_\psi \circ B$ as the leaf evaluator. At node s, child a is selected by PUCT:

$$a^* = \arg \max_{a \in \mathcal{A}(s)} \left[\bar{Q}(s, a) + c_{\mathrm{puct}} \pi_\theta(a \mid s) \frac{\sqrt{N(s)}}{1 + N(s, a)} \right]. \tag{26}$$

Leaves are expanded with policy priors. A leaf state is evaluated by rolling out, deckbuilding, and applying V_ψ, or by a learned pool-value approximation. The final decision may use visit counts or posterior action value.

Algorithm 3 MCTS pick recommendation

Require: Root state s_0, simulations M, policy π_θ, value V_ψ, deckbuilder B

for $m = 1$ to M **do**
  $s \leftarrow s_0$; path $\leftarrow []$
  while s is expanded and nonterminal **do**
    choose a by PUCT
    append (s, a) to path
    $s \leftarrow \mathrm{Step}(s, a)$, including opponent-policy transitions as needed
  end while
  if s is terminal **then**
    $v \leftarrow V_\psi(B(Y_T, m), r_0)$
  else
    expand s with priors $\pi_\theta(\cdot \mid s)$
    complete or bootstrap to estimate v
  end if
  for each (s, a) in path **do**
    update $N(s)$, $N(s, a)$, and $\bar{Q}(s, a)$ with v
  end for
end for
return $\arg\max_{a \in \mathcal{A}(s_0)} N(s_0, a)$

8. Preference Fine-Tuning

The outcome model can create static preference pairs for direct preference optimization. For a logged state s, choose two legal actions a and b. Estimate $Q(a \mid s)$ and $Q(b \mid s)$ by one-step scoring or rollouts. If $Q(a) > Q(b) + \epsilon$, create preference $(s, a \succ b)$; if $Q(b) > Q(a) + \epsilon$, create $(s, b \succ a)$. Human disagreement audits can be added to the same preference set.

In addition to deck-quality preferences, we also train a lightweight game-card bonus model from aggregate outcome data. The best variant is a regularized card-main-effect binomial model,

$$\operatorname{logit}(\Pr(\mathrm{win})) = \beta_0 + \sum_i d_i \beta_i + \rho^\top r, \tag{27}$$

where d_i indicates card inclusion and r are covariates. Pair-interaction variants were tested but overfit relative to the card-only model. Game-card preference pairs are then generated by preferring the in-pack candidate with larger learned β_i, subject to a minimum gap. This produces both capped and uncapped DPO datasets; the uncapped dataset contains approximately 1.83M train pairs and 0.20M eval pairs.

Algorithm 4 Training pipeline

1: Build $\mathcal{D}_\pi$ from 17lands draft rows by reconstructing pack, pool, and seen-unpicked features per draft.
2: Train π_{θ_0} on $\mathcal{D}_\pi$ using $\mathcal{L}_\pi$.
3: Build $\mathcal{D}_V$ from 17lands game rows; aggregate by deck build when useful.
4: Train V_ψ using $\mathcal{L}_V$ or $\mathcal{L}_V^{\text{agg}}$.
5: Fit or implement deckbuilder B from draft pools to built decks.
6: Generate preference pairs with $V_\psi \circ B$ and optional human audits.
7: Initialize $\pi_{\text{ref}} \leftarrow \pi_{\theta_0}$ and fine-tune π_θ with $\mathcal{L}_{\text{DPO}}$.
8: Deploy with reranking, rollout search, or MCTS depending on latency budget.

Given a frozen reference policy π_{ref}, DPO optimizes

$$\Delta_\theta(s, a^+, a^-) = \log \frac{\pi_\theta(a^+ \mid s)}{\pi_{\text{ref}}(a^+ \mid s)} - \log \frac{\pi_\theta(a^- \mid s)}{\pi_{\text{ref}}(a^- \mid s)},$$
$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(s,a^+,a^-)} \log \sigma\left(\tau \Delta_\theta(s, a^+, a^-)\right). \tag{28}$$

where τ is an inverse-temperature hyperparameter. This moves the policy toward outcome-preferred picks while anchoring it to the human-imitation model.

9. Evaluation

We evaluate the policy component by top-1 accuracy, top-3 accuracy, and mean reciprocal rank on held-out human picks, with draft-level splits to avoid leakage. We evaluate the value model by held-out log loss, Brier score, calibration error, and rank-stratified AUC. The combined assistant is evaluated by offline counterfactual disagreement audits: positions where the assistant and logged human pick differ are sampled for expert review, and by simulated draft rollouts scored by the frozen value model. Synthetic out-of-vocabulary evaluation with held-out cards measures robustness to cube-list changes.

10. Empirical Results

Human-pick imitation and value reranking. Continuing human-pick training from an outcome-weighted checkpoint yields final held-out metrics

$$\text{top1} = 0.618, \quad \text{top3} = 0.908, \quad \text{MRR} = 0.767. \tag{29}$$

On a 20k sampled sweep, the strongest human-like continued model obtains top-1 0.6240, top-3 0.9093, and generated preference accuracy 0.5494. Adding value reranking to this model raises preference accuracy to 0.6062 with nearly unchanged human-pick metrics.

Table 1. Game-card bonus outcome model variants. Lower is better.

Variant	Log loss	Brier
Card-only	0.67798	0.05528
1k pairs	0.67943	0.05594
3k pairs	0.68319	0.05763
1k pairs, strong reg.	0.67929	0.05588

Table 2. 20k sampled policy sweep. $\lambda = 1$ denotes calibrated value reranking with temperature 1.2016 where applicable.

Config	Top-1	Top-3	Pref. acc.
Human continued, $\lambda = 0$	0.6240	0.9093	0.5494
Human continued, $\lambda = 1$	0.6234	0.9077	0.6062
Safer game-data, $\lambda = 1$	0.6161	0.9048	0.6406
Aggressive pref-acc, $\lambda = 1$	0.5092	0.8411	0.7262

Game-card bonus model. We trained card-only and pair-interaction binomial outcome models on aggregated 17lands Powered Cube outcomes. Table 1 shows that the card-only model gives the best validation loss; pair interactions are noisy in this data regime.

The largest learned bonuses are qualitatively plausible for Powered Cube, including *Ancestral Recall*, *Time Walk*, *Black Lotus*, *Sol Ring*, *Mana Crypt*, and the Moxen.

DPO from game-card preferences. Table 2 summarizes the main policy trade-off. The safer uncapped DPO model improves generated game-data preference behavior while retaining high human agreement. The aggressive preference-accuracy model substantially increases preference accuracy, but at a large cost to human-pick imitation. Extending aggressive training from 100k to 500k steps increased training margins but did not materially improve external preference accuracy, so the 100k checkpoint is used as the experimental aggressive preset.

Qualitative behavior. On an actual held-out 17lands draft sequence, the safer model matched the human on picks such as *Skyclave Apparition* and *Windswept Heath*, while the aggressive model selected higher game-bonus alternatives such as *Ouroboroid*, *Birds of Paradise*, *Demonic Tutor*, and *Zuran Orb*. This illustrates the intended behavior of the aggressive model: it is not a human-imitation policy, but a value/preference-seeking policy.

We also evaluated the search layer on real depleted packs from a held-out 17lands draft prefix. In one eight-pick trace, the logged human followed a white/fixing lane with picks such as *Skyclave Apparition*, *Windswept Heath*, *Scrubland*, *Sacred Foundry*, and *Winds of Abandon*. MCTS instead made an early *Blood Crypt* choice and then produced a co-

Table 3. MCTS inference latency on CPU with checkpoints preloaded.

Simulations	p50	p95	Mean
10	418 ms	574 ms	378 ms
25	942 ms	1370 ms	850 ms
50	1825 ms	2697 ms	1642 ms

herent black-red, reanimator-leaning line: *Hymn to Tourach*, *Demonic Tutor*, *Valki, God of Lies*, *Archon of Cruelty*, *Emperor of Bones*, and *Bone Shards*. This trace is not a counterfactual replay of hidden packs, but it suggests that search can improve internal strategic consistency: it may amplify a questionable early lane choice, yet subsequent picks are more coherent with that lane than static one-step policies.

MCTS latency. We benchmarked inference-time search with models loaded once on real held-out pick states. Table 3 reports wall-clock CPU latency. A 25-simulation budget is practical for live use as an optional search-based reranker.

Live-client integration. We implemented an experimental MTGA Player.log follower for macOS/Epic Games installs. In a Quick Draft log, the client emitted scene changes into `Draft`, outgoing `BotDraftDraftPick` messages, and nested payloads containing `DraftPack`, `PickedCards`, `PackNumber`, and `PickNumber`. This is sufficient to reconstruct live pack and pick state for formats whose Arena card ids can be mapped to names. Non-cube drafts require an up-to-date global Arena id map and are out-of-domain for the Powered Cube policy.

11. Discussion

The system intentionally avoids end-to-end reinforcement learning as a first step. Human pick imitation supplies stable local behavior, game outcomes supply a separate terminal signal, and search or preference fine-tuning combines them without high-variance credit assignment across an entire draft. This decomposition also cleanly separates three sources of error: policy imitation error, deckbuilder error, and value-model outcome error.

References

17Lands. 17lands public datasets. https://www.17lands.com/public_datasets, 2026. Accessed 2026-06-24.

Scryfall. Scryfall bulk data. https://scryfall.com/docs/api/bulk-data, 2026. Accessed 2026-06-24.