Extended Abstract

Motivation Creative tasks often involve implicit, user-specific aesthetic preferences; a policy optimal for one user may not be optimal for another user. Learning these preferences is an open problem, as these preferences are often highly subjective, based on intuition, and poorly specified. Existing approaches like RLHF employ human preferences between pairs of trajectory segments to solve complex RL tasks but aggregate preferences into a single reward model (Christiano et al., 2023), while methods like Variational Preference Learning (VPL) require extensive datasets collected over several users (Poddar et al., 2024). Within the slide editing domain, we investigate how a system can learn to edit a slide to match a user's preference based on only a few examples and how the quality of these examples affects performance.

Method Our approach decomposes the problem into inference (estimating slide element target layouts from a small set of examples, termed context slides) and control (sequentially editing a new slide to match these implied preference targets). We evaluate three control methods on a discrete environment: tabular Q-learning with a learned attention encoder for inference, behavior cloning from an oracle expert with access to the ground-truth target layouts, and PPO with GAE using mean-pooled inference targets. To evaluate the robustness of these methods, we introduce a context degradation framework that corrupts slides via four regimes: Drop, Noise, Mismatch, and their composition (All), and vary the severity (i.e., the proportion of context slides degraded in an experiment). We also conduct preliminary experiments with an end-to-end PPO implementation on a continuous environment that jointly learns inference and control.

Implementation We represent slides with four indexed slots (two images and two textboxes) with position and size coordinates. In the discrete case, we utilize a 9×16 two-dimensional grid and eight discrete actions that operate in increments of exactly one grid unit. For the continuous case, we use normalized coordinates for position $(x_i, y_i) \in [0, 1]^2$ and size $(w_i, h_i) \in [0, 1]^2$. The Q-learning controller uses two Q-tables (one for navigation, and one for resizing), while the Behavior Cloning Oracle is an MLP with an 8-way categorical output head that learns from the oracle expert. The discrete PPO policy uses an MLP with an 8-way categorical output head with a degradation curriculum that gates and anneals context degradation during training. The preliminary continuous PPO implementation uses an LSTM-based actor-critic with a reward containing alignment, overlap penalty, and progress terms.

Results Context quality dominates performance: under the composed degradation regime, the upper-bound Behavior Cloning Oracle method drops from 0.741 to 0.507 macro IoU at just $s = 0.2$. Q-learning overfits to inferred targets, with Q-learning's performance degrading severely from 0.714 macro IoU at $s = 0$ to 0.319 at $s = 0.2$ and merely 0.085 at $s = 0.4$. Discrete PPO flatlines at near-zero IoU across all regimes despite accurate mean-pooled targets at $s = 0$, indicating an inability to learn effective control. Continuous PPO exhibited degenerate behavior and reward hacking, despite attempts to mitigate exploits.

Discussion Not all context degradation is equal; mismatch (misleading context) is far more harmful than drop (missing context). Even with a supervised attention encoder for inference, context degradation is difficult to overcome. Reward specification is the primary bottleneck for PPO-based approaches, suggesting that learned reward models (which require large amounts of data) could be a potential solution, enabling us to better investigate PPO's theoretical advantage in recovering from degraded content.

Conclusion Our findings highlight the challenge of robust inference under degraded and/or limited context, as well as designing expressive, robust, and exploit-resistant reward functions. We found decomposing the slide editing task into inference and control stages invaluable in accurately diagnosing the limitations of our three discrete approaches. Future work could include utilizing learned reward models, incorporating universal aesthetic standards to supplement learned user preferences, and simulating a highly-realistic PowerPoint-like environment.

Personalizing Slide Layouts: A Case Study in RL Reward and Context Bottlenecks

Ryan Le
Department of Computer Science
Stanford University
ryanmtle@stanford.edu

Elijah Song
Department of Computer Science
Stanford University
elijahs2@stanford.edu

Abstract

Everyday creative tasks (such as slide editing, front-end UI/UX design, and financial modeling) often involve sequential editing decisions, where the deemed "quality" of the final output is dependent upon the aesthetic preferences of the evaluator. To this end, we study slide layout personalization, decomposing the problem into inference (estimating preferences from a small set of example slides) and control (editing a new slide to match the inferred style preferences). We evaluate three control methods (tabular Q-learning, behavior cloning from an oracle expert, and PPO) under four context degradation regimes (Drop, Noise, Mismatch, and their composition) at varying severities, finding that context quality is the dominant factor in task performance. Reward specification emerges as a second bottleneck: PPO fails to learn an effective control policy in continuous and discrete settings, despite its theoretical advantage in recovering from degraded context.

1 Introduction

Learning subjective user-specific preferences is an open problem in RL, given that a policy optimal for one user may be suboptimal for another user (Casper et al., 2023). Furthermore, a given user's aesthetic preferences are often implicit–it is not uncommon for users to deem a creative output as "satisfactory" or "unsatisfactory" without fully understanding the reasoning behind their evaluation. Even in domains where style preferences are grounded in rule-based principles (e.g. aspect ratio, relative font sizing, etc. in the slide editing domain), these rules are rarely made explicit by users and often need to be inferred.

We explore this learning challenge in the context of slide editing: given a small context consisting of a user's example slides, a policy must learn to edit a new slide to match that user's style preferences. To this end, we decompose the problem into two sub-problems, namely *inference* and *control*. The objective of the inference stage is to learn user preferences from context (example slides), whereas in the control stage, a policy executes sequential edits to bring the current (new) slide into alignment with the inferred user preferences, roughly simulating the human editing process. The inclusion of a control stage is motivated in part by the broader goal of building interactive creative assistants that can actively help users accomplish creative tasks.

Our primary objective is to investigate how context quality affects performance across a variety of RL and non-RL approaches, and to explore when a pure end-to-end reinforcement learning approach (where inference and control are controlled by a single policy) may perform better than approaches that explicitly separate inference and control.

2 Related Work

Reinforcement through human feedback (RLHF) Christiano et al. (2023) provides a natural framework for learning user-specific preferences without access to the true reward by employing human preferences between pairs of trajectory segments. However, vanilla RLHF aggregates preferences across users into a single modeled reward and lacks a mechanism for user-level personalization. To this end, Poddar et al. (2024) introduced Variational Preference Learning (VPL), which infers a "novel user-specific latent and [learns] reward models and policies conditioned on this latent without additional user-specific data." This method, however, requires a dataset of preferences collected from a large set of known users at training time, a luxury not often afforded in many real-world creative settings, where a given user may have only a few examples to provide. In the language modeling space, few-shot preference optimization by Singh et al. (2026) enables an LLM to quickly model a personalized reward function based on just a few preferences. Our domain, however, involves an action space of presentation slides that consists of a layout with text and image elements.

Within the slide generation space, several non-RL methods currently exist. In particular, SlideTailor by Zeng et al. (2025) takes in a paper-slides sample pair and a `.pptx` slide template to infer a user's aesthetic preferences. This method, however, is primarily focused on generating slides for scientific papers, and the user preferences are inferred statically and cannot be updated based on user corrections or evolving user preferences. AeSlides by Pan et al. (2026) is an RL framework that evaluates and improves slide quality using verifiable rewards based on fixed, non-user-specific metrics (namely, aspect ratio compliance, whitespace, element collision, and visual imbalance). AeSlides, however, is unable to learn user-specific aesthetic preferences as it trains on a static dataset.

3 Methods

3.1 Problem Formulation

Given a small set of example slides from a user (denoted as context slides), our goal is to edit a new slide to match that user's implicit style preferences. We approach few-shot slide layout personalization via a two-stage process: first, **inference**, which estimates a user's implicit preferences from a set of context slides and **control**, which executes a sequence of edits to bring a new slide into alignment with the user's inferred preferences.

Formally, let a two-dimensional slide S consist of K indexed slots which each contain an element $e_i = (t_i, x_i, y_i, w_i, h_i)$ where t_i denotes the type of the element ($t_i \in \{\texttt{image}, \texttt{textbox}\}$) and (x_i, y_i, w_i, h_i) describe the position and size of each element. Each episode provides a set of context slides $C = (C_1, \ldots, C_N)$ (used in the inference stage to generate targets) and a current slide S to edit. The ground-truth user layout is hidden from the policy and only used for reward computation and evaluation.

At each timestep in the control stage, an action applies one of $\mathcal{A} = \{\texttt{move\_x}, \texttt{move\_y}, \texttt{resize\_w}, \texttt{resize\_h}\}$ to element e_i with magnitude chosen such that the edited element remains in-bounds within the slide. Furthermore, the context slides are optionally degraded prior to the inference stage, in order to investigate how context quality affects performance across methods.

We consider two environments that instantiate this formulation: a continuous-coordinate environment used in preliminary experiments, and a discrete grid environment that forms the basis of our main results and our investigation into how context degradation affects different methods.

3.2 Continuous Environment

In the continuous environment, for an element $e_i = (t_i, x_i, y_i, w_i, h_i)$, let $(x_i, y_i) \in [0, 1]^2$ denote the normalized coordinates of the top-left corner of the element e_i's rectangular, axis-aligned bounding box and $(w_i, h_i) \in [0, 1]^2$ denotes the normalized width and height of e_i's bounding box, respectively. At each timestep in the control stage, the policy selects a slot i_t, and an operation $o_t \in \{\texttt{move\_x}, \texttt{move\_y}, \texttt{resize\_w}, \texttt{resize\_h}\}$, and a magnitude $\delta_t \in \{-0.1, -0.05, -0.02, 0, 0.02, 0.05, 0.1\}$, and applies this edit (which is clipped to ensure that the box remains in-bounds).

3.3 Discrete Environment

In the discrete environment, we define a rectangular grid of size $n \times m$. For an element $e_i = (t_i, x_i, y_i, w_i, h_i)$, let (x_i, y_i) denote the integer tile coordinates of the top-left corner of the element e_i's rectangular, axis-aligned bounding box and (w_i, h_i) denote the integral width and height of e_i's bounding box, respectively. At each timestep in the control stage, the policy selects a slot i_t, and an operation $o_t \in \{\texttt{UP}, \texttt{DOWN}, \texttt{LEFT}, \texttt{RIGHT}, \texttt{GROW\_H}, \texttt{SHRINK\_H}, \texttt{GROW\_W}, \texttt{SHRINK\_W}\}$; note that in the discrete case, each operation o_t has magnitude of exactly one tile and the action names reflect this framing.

3.4 Preliminary Method: Continuous PPO Pilot

Initially, we focused solely on learning a user's layout preferences via an **End-to-End PPO** (Schulman et al., 2017) actor-critic policy that jointly learns to use context and execute edits from a reward function alone, without conducting inference separately from control. The policy receives as input context slides $C = (C_1, \ldots, C_K)$ and the slide to edit S and executes one edit per timestep in the continuous action space.

The actor and critic share an LSTM over a sequence of length $N + 1$: the N context slides followed by the current slide S_t. Each frame encodes the slide layout (slot types and normalized box coordinates) together with per-slot target boxes inferred from context slides. The LSTM hidden state is concatenated with a one-hot indicator of the slot i_t being edited; the actor then outputs two categorical distributions over (i) the operation $o_t \in \{\texttt{move\_x}, \texttt{move\_y}, \texttt{resize\_w}, \texttt{resize\_h}\}$ and (ii) a binned magnitude δ_t.

At each timestep exactly one slot is edited: the environment applies the chosen operation and magnitude with clipping so boxes remain in-bounds. Training uses on-policy PPO with GAE and a comprehensive reward function that takes into account each element's alignment with the inferred target boxes, overlap, and progress shaping.

Reward Function: The step reward is:

$$r_t = \bar{s}(S_t) - \lambda_{\text{ov}} \,\text{Overlap}(S_t) + w_p \,\Delta\bar{s}_t + w_e \,\Delta s_{i_t,t} - \text{Over-Alignment}(S_t),$$

with a terminal bonus β added at the end of an episode, where $\bar{s}(S_t)$ is a slide-level alignment score, Overlap(S_t) is a slide overlap penalty, $\Delta\bar{s}_t$ and $\Delta\bar{s}_{i,t}$ are progress terms, and Over-Alignment(S_t) is a penalty such that editing a slot already above an alignment threshold τ is penalized to discourage overfitting. More details are available in the appendix.

This approach consistently failed to produce meaningful editing behavior across multiple reward iterations, including an initial run that used negative ℓ_2 distance to the target boxes, zone-coverage terms, and all-or-nothing multipliers. We discuss the specific failure modes in our Results and Discussion sections, and these negative results motivated our pivot to the discrete grid environment with explicit inference-control stages, which forms the basis of our main investigation.

3.5 Discrete Methods: Two-Stage Pipeline

Given the limitations of the continuous PPO-based approach, we pivoted to a discrete grid environment with a two-stage pipeline that separates inference from control. We also introduce context degradation and investigate how context quality affects downstream performance across a variety of methods.

3.5.1 Context Degradation

Prior to the inference stage, the context is optionally degraded via the following methods, with $s \in [0, 1]$ denoting the *severity* of degradation in a given method:

- **Drop**: Remove a fraction s of context slides, reducing the number of demonstrations available for inference.
- **Noise**: Apply Gaussian jitter to context element positions and sizes scaled by s, which randomly corrupts the signal within each context slide.
- **Mismatch**: Replace a fraction s of context slides with slides drawn from a different random user, injecting irrelevant preference signal into the context.

- **All**: Compose drop $\rightarrow$ noise $\rightarrow$ mismatch simultaneously. This represents the worst-case and most difficult degradation regime.

Degradation is applied only to context slides C; the current slide S and ground-truth user preference layout z^* remain unchanged, thus isolating the effect of context quality on inference and downstream control. Examples of context degradation can be found in Appendix C.

3.5.2 Inference

The inference stage produces a per-element target $\hat{z}_i = (\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i)$ from (potentially degraded) context slides C via the following methods:

- **Mean Pooling**: The inferred target for each element is the average position and size of the corresponding element over all context slides.
- **Supervised Attention Encoder**: The inferred target for each element is the output from a learned, per-element attention encoder that attends over the context slides conditioned on the current slide S. This encoder is frozen at evaluation time.
- **Oracle**: The ground-truth target layout is provided directly to the control stage, isolating control performance from inference error and serving as a diagnostic upper bound.

3.5.3 Control

The control stage executes a sequence of edits to transform the current slide S given the targets (essentially, estimated user preferences) from the inference stage, via the following methods:

- **Q-Learning**: We use a two-phase split Q-learning approach on the discrete environment, where for each element, we first navigate to the inferred target position, and then resize the element to the inferred target size. We use two separate Q-tables: Q_{nav} for navigation and Q_{resize} for resizing.

  We update Q-values via one-step tabular Q-learning:

  $$Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big].$$

  1. **Navigation**: For the navigation stage, the state at a given slot i is defined as the coordinate-wise deltas between the current and the inferred target location, namely: $(\Delta x_{i,t}, \Delta y_{i,t}) = (\hat{x}_i - x_{i,t}, \hat{y}_i - y_{i,t})$. Q_{nav} selects an action among $\{\texttt{UP}, \texttt{DOWN}, \texttt{LEFT}, \texttt{RIGHT}\}$ using a reward function based on the true targets: $r^{\text{nav}}_{i,t} = -(|x_{i,t} - x^*_i| + |y_{i,t} - y^*_i|)$.
  2. **Resizing**: Similarly, for the resizing stage, the state at a given slot i is defined as the coordinate-wise deltas between the current and inferred target size, namely: $(\Delta w_{i,t}, \Delta h_{i,t}) = (\hat{w}_i - w_{i,t}, \hat{h}_i - h_{i,t})$. Q_{resize} selects an action among $\{\texttt{GROW\_H}, \texttt{SHRINK\_H}, \texttt{GROW\_W}, \texttt{SHRINK\_W}\}$ using a reward function based on the true targets: $r^{\text{resize}}_{i,t} = -(|w_{i,t} - w^*_i| + |h_{i,t} - h^*_i|)$.

  At evaluation time, both Q-tables are frozen with greedy action selection.
- **Behavior Cloning with Oracle (Upper Bound)**: We train a controller that clones a greedy oracle expert that acts towards the ground-truth user preferences $\{z^*_i\}$; note that the ground-truth is hidden from the controller and the controller only has access to the context targets $\{\hat{z}_i\}$ based on possibly degraded context. This serves as an upper bound on control performance, demonstrating how effective a controller can be at approaching the true layout given an oracle expert.
- **Discrete PPO**: We also train a discrete PPO actor-critic that uses mean-pooled inference targets rather than the learned encoder, to isolate the contribution of learned inference from learned control. The policy receives mean-pooled inferred targets $\hat{z}_i$ and the current slide S as input, and executes one discrete edit per slot in parallel at each timestep.

  The actor and critic share an MLP over a concatenated per-slot observation: for each slot i, the input encodes normalized deltas to inferred targets $\hat{z}_i$, the inferred target coordinates $\hat{z}_i$ themselves, and two phase indicators (`resize_mode`, `done`) marking whether the slot is navigating, resizing, or complete. Like our Q-Learning method, editing proceeds in two

phases: *navigation* toward the target position, then *resizing* toward the target size. The policy outputs one action per slot in parallel at each timestep via factorized categorical distributions over actions.

Training uses PPO with GAE and a per-step reward against true user targets:

$$r_t = -\sum_{i=1}^{K} \left\| z_i^{t+1} - z_i^* \right\|_{\text{sq}}^2,$$

where $z_i = (x_i, y_i, w_i, h_i)$ and $\| \cdot \|_{\text{sq}}^2$ sums squared errors over the four tile coordinates. As with the Q-learning controller, rewards use true goals z_i^* during training only.

During training, we use a two-stage degradation curriculum over context quality:

1. **Gate phase.** Context is non-degraded (i.e., $s = 0$) until the policy's recent training error on such context falls below a threshold. Specifically, we transition to the annealing phase when the rolling mean of the control error $\bar{d}_{\text{ctrl}}$ falls below a fixed threshold, where we define
$$\bar{d}_{\text{ctrl}} = \tfrac{1}{2}\big(d_{\text{image}} + d_{\text{text}}\big)$$
and d_{type} is the mean $\|z_i - z_i^*\|_2$ over slots of that type (image, text).
2. **Annealing phase.** Once the gate opens, degradation severity is increased linearly as a function of the global training-episode index, up to a maximum cap. The purpose of this is to gradually shift training from easy (less degraded) targets to harder (degraded) targets.

4 Experimental Setup

4.1 Dataset

All episodes are procedurally generated. Each episode samples a random $K = 4$ slot goal layout (with exactly two image slots and two textbox slots) and generates $N = 10$ context slides by randomly jittering element positions and sizes around the true goals.

4.2 Continuous PPO Hyperparameters

The continuous policy was trained for $10,000$ episodes with PPO and GAE ($\gamma = 0.99$, $\lambda = 0.95$, $\epsilon = 0.2$). Each episode runs for $T = 100$ steps with slots edited in round-robin order ($i_t = t \bmod K$). The actor-critic uses an LSTM with hidden size $H = 128$, with categorical heads over operation o_t and the binned magnitude $\delta_t \in \{-0.1, -0.05, -0.02, 0, 0.02, 0.05, 0.1\}$.

We employed the Adam optimizer (lr $= 3 \times 10^{-4}$), value-loss coefficient $c_v = 0.5$, entropy coefficients 0.05 on both action components (operation and magnitude), and gradient clipping at 0.5.

4.3 Discrete Environment

All discrete experiments utilize a 9×16 tile grid, simulating a standard widescreen presentation slide.

4.3.1 Q-Learning

Training runs for $10{,}000$ episodes with a learning rate $\alpha = 0.5$ and discount $\gamma = 0.99$. We use an ϵ-greedy exploration strategy with linear annealing, with $\epsilon : 1.0 \to 0.05$ over a single training run. The maximum episode length is 500 steps and the Q-tables are frozen during evaluation.

For the inference stage, the Q-learning approach uses inferred targets generated by the Supervised Attention Encoder.

4.3.2 Behavior Cloning Oracle

The controller is modeled as a two-layer MLP with hidden size $H = 128$ and $\tanh$ activations, followed by an 8-way categorical output head (corresponding to the discrete actions $\{\texttt{UP}, \texttt{DOWN}, \texttt{LEFT}, \texttt{RIGHT}, \texttt{GROW\_H}, \texttt{SHRINK\_H}, \texttt{GROW\_W}, \texttt{SHRINK\_W}\}$).

We collect $2{,}500$ expert rollouts (with a maximum episode length of 500 steps each). During training, the controller policy's context slides are degraded via only the **Drop** method at severity sampled uniformly from $\{0\}$, $\mathrm{Unif}(0.25, 0.85)$, or $\mathrm{Unif}(0.4, 0.9)$ per rollout. The policy is optimized over 15 epochs using the Adam optimizer (lr $= 3 \times 10^{-4}$) with batch size 1024 and gradient clipping at 1.0.

4.3.3 Discrete PPO

The discrete PPO experiment was trained for $10{,}000$ episodes with GAE ($\gamma = 0.99$, $\lambda = 0.95$, $\epsilon = 0.2$). Each episode runs for up to $T = 500$ steps. The actor-critic uses an MLP with hidden size $H = 128$, $\tanh$ activations, and an 8-way categorical output head, like the controller in Behavior Cloning Oracle.

We employed the Adam optimizer (lr $= 3 \times 10^{-4}$), value-loss coefficient $c_v = 0.5$, entropy coefficients 0.05, and gradient clipping at 0.5. Inference at both train and eval time is mean pooling over context slots (no learned encoder). We transition to the annealing stage from the gate stage when $\bar{d}_{\mathrm{ctrl}} \geq 2.0$. In the annealing stage, we increase the training severity s linearly over up to $8{,}000$ episodes from 0.0 to 1.0.

4.4 Evaluation Metrics

At evaluation, we apply degradation to the context slides at severities $s \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. We report means over 50 evaluation episodes per (method, s) pair and use random seed 42.

4.4.1 Intersection Over Union (IoU)

Note that each element occupies an axis-aligned, rectangular bounding box on a two-dimensional Cartesian plane. For boxes A and B, let

$$\mathrm{IoU}(A, B) = \frac{\text{area}(A \cap B)}{\text{area}(A \cup B)}.$$

We define the **Macro IoU** for a given slide S as:

$$\mathrm{IoU}_{\mathrm{Macro}} = \frac{1}{K} \sum_{i=1}^{K} \mathrm{IoU}_i.$$

where $\mathrm{IoU}_i = \mathrm{IoU}(e_i, z_i^*)$, or in other words, the IoU of element i's bounding box with its corresponding target's bounding box.

5 Results

5.1 Preliminary Continuous PPO Results

Our preliminary experiments with a continuous PPO setup yielded generally unsatisfactory results; even as we iteratively refined the reward function to prevent reward hacking and degenerate behavior, the policy always found an unintended shortcut for the reward functions we tried.

For example, a basic reward function that used negative ℓ_2 distance of the bounding boxes coordinates (location and size) to the corresponding targets' bounding boxes coordinates did not converge; upon the addition of the overlap penalty and progress terms, elements grow maximally large and often covered the entire presentation screen (see Figure 3). Adding in zone coverage to the reward function caused elements to stack and the agent to move only one or two elements instead of all four elements on a slide.

These diverse failure modes pointed to a fundamental reward specification problem and a need to rethink our approach to the problem, and led us to explore context degradation on a discrete grid instead.

5.2 Context Degradation Results

Our context degradation experiments show that context quality is one of the most important factors for performance on the preference-learning task; an increase in the severity of context degradation

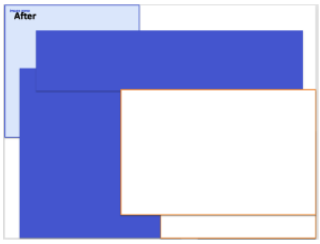


Figure 1: Degenerate behavior when using a reward function that penalizes overlap and includes progress shaping terms; note that the dark blue boxes represent images and the white boxes with an orange border represent textboxes–here the inferred targets (green boxes) are not visible.

leads to a decrease in performance versus the true target, which is to be expected. For example, in the degradation regime **All**, we achieve 0.741 macro IoU on a non-degraded context for the BC Oracle method, which drops to 0.507 macro IoU when just 20% of the context is degraded (i.e. $s = 0.2$).

Interestingly, the dropoff in performance due to degraded context is quite severe for the Q-Learning control approach, where in the same regime, a 0.714 macro IoU at $s = 0$ drops to 0.319 at $s = 0.2$ and to just 0.085 at $s = 0.4$. We hypothesize, based on the ℓ_2 distances to the *inferred* target layouts in the Q-Learning case being much smaller than with the other methods across various context degradation regimes, that Q-learning control often devolves into an essentially "greedy" approach, overfitting to inferred targets and not exploring further.

Looking at the various context degradation regimes that only applied one source of degradation, we see that **Drop** was the least harmful form of degradation and **Mismatch** was by far the most harmful form of degradation. This may be because missing context is less harmful to performance than actively misleading context; simply missing a few slides still enables the inference to be accurate by interpolating data from existing non-degraded slides, whereas mismatch mixes multiple user preferences into the same context, leading to confusion even with a supervised attention encoder for inference.

Notably, the Discrete PPO method (using mean-pooling for inference) flatlines with a performance close to zero across all context degradation regimes and even with a non-degraded context. At $s = 0$, the mean-pool targets are already relatively accurate (with macro IoU with respect to true targets equal to 0.91), so the poor outcome implies control learning failure during PPO. We hypothesize that the per-step reward (being the *summed* negative ℓ_2 distances to inferred targets across the four coordinates (x_i, y_i, w_i, h_i)) was too noisy and provided an unreliable signal for the policy in PPO. As demonstrated by the continuous PPO experiment, designing a reliable reward function for this task is a difficult challenge. Furthermore, without a warm-start (perhaps with behavior cloning) and poor PPO performance in initial episodes, the annealing phase during training was never triggered, so the policy never learned how to deal with context degradation.

Table 1: Evaluation by degradation type (mean over $s \in \{0, 0.2, \ldots, 1.0\}$)

Degradation	Method	Macro IoU	Mean ℓ_2 to True	Mean ℓ_2 to Inferred
Drop	BC Oracle	0.571	2.487	2.626
	Q-Learning	0.524	1.358	0.489
	Discrete PPO	0.018	5.532	5.878
Noise	BC Oracle	0.440	2.437	2.646
	Q-Learning	0.376	1.214	0.929
	Discrete PPO	0.018	5.532	5.193
Mismatch	BC Oracle	0.323	4.146	4.198
	Q-Learning	0.193	2.683	0.941
	Discrete PPO	0.018	5.532	4.170
All	BC Oracle	0.283	4.407	4.321
	Q-Learning	0.201	2.710	0.793
	Discrete PPO	0.018	5.532	5.256

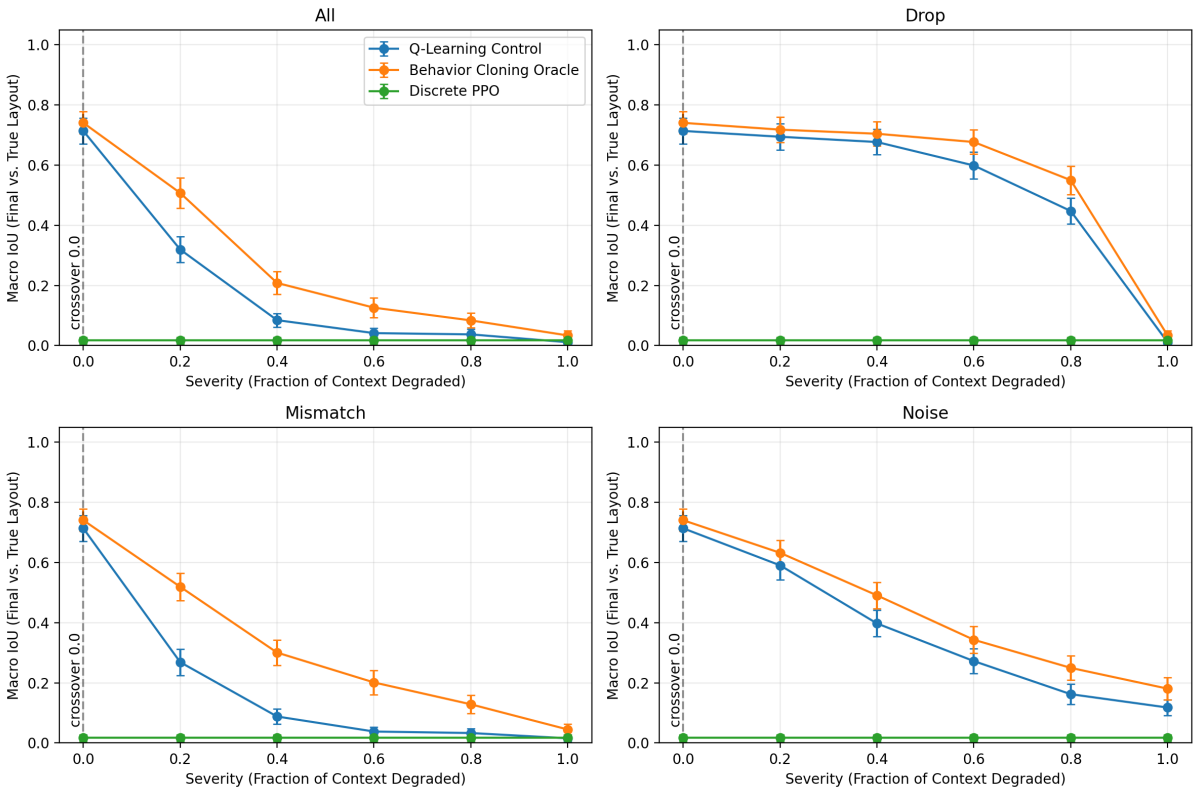


Figure 2: Macro IoU (Final vs. True Layout) vs. Context Degradation Severity on various context degradation modes by method.

6 Discussion

The quality and expressiveness of our reward function served as a major bottleneck in both the continuous and discrete PPO experiments. Defining an expressive, dense reward function in an underexplored domain that is also resistant to reward hacking is a challenging task, particularly in creative domains. To that end, learned reward models present an enormous potential in fixing the reward bottleneck, but require an extensive data collection pipeline. However, this contradicts the original objective of learning user preferences in the slide editing domain with few-shot learning. Further experiments could involve warm-starts with behavior cloning or other methods, as well as potentially building a lightweight learned reward model that does not necessarily require an extensive data collection pipeline. In theory, approaches like PPO should be more robust when the inferred targets are corrupted compared to approaches like Q-Learning.

Furthermore, context quality and by extension, inference, was a major bottleneck in performance. Even a well-performing, upper bound control method like Behavior Cloning Oracle degraded sharply when context is corrupted, particularly under the Mismatch degradation regime. The use of a supervised attention encoder during the inference stage in both the BC Oracle and Q-Learning methods revealed the limits of a learned attention when training and test distributions diverge significantly due to context degradation.

7 Conclusion

We investigated learning user aesthetic preferences via the slide editing domain, decomposing the problem into an inference stage (estimating user preferences from context) and a control stage (editing slides to match inferred user preferences). Our primary finding is that context quality (and by extension, inference) is the dominant factor in task performance: all working methods perform significantly worse as context degrades, with misleading information (Mismatch) being the most

harmful. Furthermore, reward function quality bottlenecks PPO performance in both the continuous and discrete domains, despite the algorithm's theoretical ability to overcome degraded targets better than other control methods. Successful future implementations are likely to combine universal aesthetic standards with user-specific preferences. Future work may include building a realistic slide environment that better simulates real PowerPoint content, possibly using LLMs or VLMs to analyze the semantic content of text and images to help guide the editing process.

8 Team Contributions

- **Ryan Le:** Set up the environment and experiments. Developed the end-to-end continuous RL approach, along with Q-Learning and Behavior Cloning Oracle. Contributed to Discrete PPO. Wrote Introduction, Methods, Experimental Setup, and Discussion.
- **Elijah Song:** Conducted experiments and refined the continuous RL approach and contributed to Discrete PPO. Wrote Related Work, Results, and Conclusion, and conducted visualizations and analysis.

Changes from Proposal Our project's focus shifted from effectively learning user preferences via PPO in the slide editing task to focusing on how end-to-end RL methods compare to factored inference and control stages, as well as how context quality affects different methods.

AI Usage We used AI tools for boilerplate, visualizing/plotting, and debugging. We personally owned and tested the core RL components, implementing everything and only resorting to AI for debugging and quick refactoring: continuous PPO actor-critic and reward implementation and reward design, discrete grid environment, Q-learning updates, BC oracle setup, discrete PPO/GAE training, and the context degradation framework.

References

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Bıyık, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. 2023. Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback. arXiv:2307.15217 [cs.AI] `https://arxiv.org/abs/2307.15217`

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep reinforcement learning from human preferences. arXiv:1706.03741 [stat.ML] `https://arxiv.org/abs/1706.03741`

Yiming Pan, Chengwei Hu, Xuancheng Huang, Can Huang, Mingming Zhao, Yuean Bi, Xiaohan Zhang, Aohan Zeng, and Linmei Hu. 2026. AeSlides: Incentivizing Aesthetic Layout in LLM-Based Slide Generation via Verifiable Rewards. arXiv:2604.22840 [cs.CV] `https://arxiv.org/abs/2604.22840`

Sriyash Poddar, Yanming Wan, Hamish Ivison, Abhishek Gupta, and Natasha Jaques. 2024. Personalizing Reinforcement Learning from Human Feedback with Variational Preference Learning. arXiv:2408.10075 [cs.LG] `https://arxiv.org/abs/2408.10075`

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 `http://arxiv.org/abs/1707.06347`

Anikait Singh, Sheryl Hsu, Kyle Hsu, Eric Mitchell, Stefano Ermon, Tatsunori Hashimoto, Archit Sharma, and Chelsea Finn. 2026. FSPO: Few-Shot Optimization of Synthetic Preferences Personalizes to Real Users. arXiv:2502.19312 [cs.LG] `https://arxiv.org/abs/2502.19312`

Wenzheng Zeng, Mingyu Ouyang, Langyuan Cui, and Hwee Tou Ng. 2025. SlideTailor: Personalized Presentation Slide Generation for Scientific Papers. arXiv:2512.20292 [cs.CL] `https://arxiv.org/abs/2512.20292`

A Context Degradation Results

Table 2: Evaluation on **All** degradation regime

Method	Severity	Macro IoU	L2 to True	L2 to Inferred Targets
BC Oracle	0.0	0.741	1.044	1.280
BC Oracle	0.2	0.507	2.170	2.558
BC Oracle	0.4	0.208	3.994	4.508
BC Oracle	0.6	0.126	5.091	5.084
BC Oracle	0.8	0.084	5.920	4.596
BC Oracle	1.0	0.034	8.222	7.899
Q-Learning	0.0	0.714	0.516	0.486
Q-Learning	0.2	0.319	1.201	0.828
Q-Learning	0.4	0.085	2.399	1.088
Q-Learning	0.6	0.041	3.355	1.147
Q-Learning	0.8	0.037	3.959	1.201
Q-Learning	1.0	0.011	4.832	0.010
Discrete PPO	0.0	0.018	5.532	5.518
Discrete PPO	0.2	0.018	5.532	5.059
Discrete PPO	0.4	0.018	5.532	4.372
Discrete PPO	0.6	0.018	5.532	4.182
Discrete PPO	0.8	0.018	5.532	4.717
Discrete PPO	1.0	0.018	5.532	7.685

B Continuous PPO Implementation Details

For the reward function in the continuous PPO experiment, we have:

$$r_t = \bar{s}(S_t) - \lambda_{\text{ov}}\,\text{Overlap}(S_t) + w_p\,\Delta\bar{s}_t + w_e\,\Delta s_{i_t,t} - \text{Over-Alignment}(S_t),$$

We define each term as follows:

- **Slide-Level Alignment Score**: $\bar{s}(S_t) = \frac{1}{K}\sum_i s_i(t)$, where the per-slot alignment score against fixed corner targets z_i^* is:

$$s_i(t) = \max\left(0, 1 - \frac{d_i^{\text{pos}}(t)}{\sigma_{\text{pos}}}\right) \cdot \max\left(0, 1 - \frac{d_i^{\text{size}}(t)}{\sigma_{\text{size}}}\right),$$

  with d_i^{pos} and d_i^{size} measuring the ℓ_2 center-point distance and size deviation from the target z_i^*, respectively.
- **Overlap Penalty**:

$$\text{Overlap}(S_t) = \sum_{i<j} \frac{\text{Area}(e_i^{(t)} \cap e_j^{(t)})}{\min(\text{area}(e_i^{(t)}),\, \text{area}(e_j^{(t)}))}$$

  which sums the normalized pairwise intersection area over all slot pairs.
- **Progress Terms**: $\Delta\bar{s}_t = \max(0, \bar{s}(S_t) - \bar{s}(S_{t-1}))$ and $\Delta s_{i_t,t} = \max(0, s_{i_t}(t) - s_{i_t}(t-1))$ reward improvement in slide-level and slot-level alignment, respectively.
- **Over-Alignment Term**: editing a slot already above an alignment threshold τ is penalized to discourage overfitting (e.g, unnecessary edits to well-placed elements).

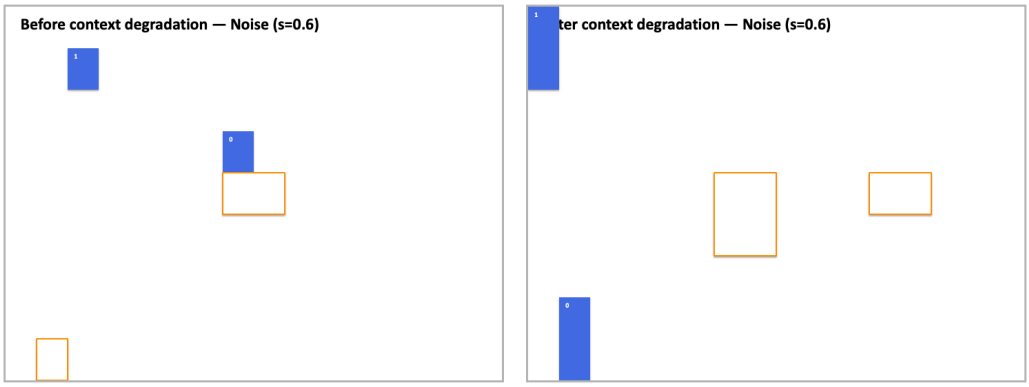
In our continuous PPO run, we used reward coefficients of $\lambda_{\text{ov}} = 0.6$, $w_p = 0.1$, $w_e = 0.4$, terminal bonus $\beta = 0.5$, alignment thresholds $\sigma_{\text{pos}} = 0.35$, $\sigma_{\text{size}} = 0.4$, and penalty threshold $\tau = 0.85$.

C Context Degradation Examples

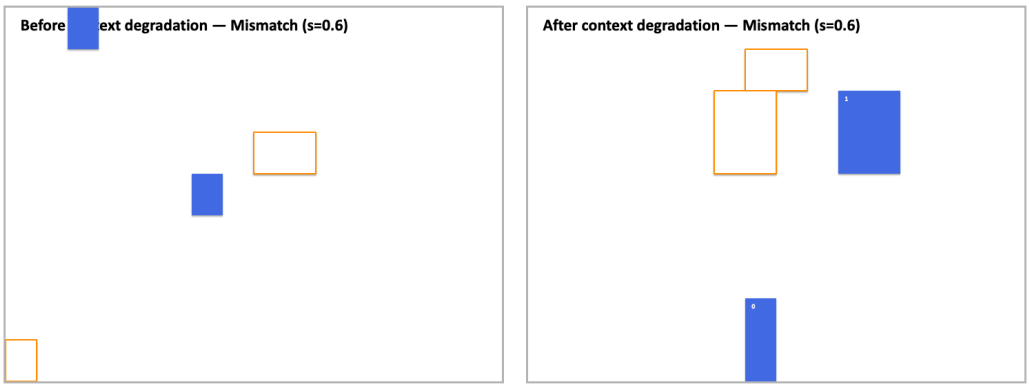
(a) All

(b) Drop

Figure 3: Context degradation examples at severity $s = 0.6$ across various modes.



(c) Noise



(d) Mismatch

Figure 3: Context degradation examples at severity $s = 0.6$ across various modes (continued).