Personalizing Reinforcement Learning from Human Feedback with Temporal Variational Preference Learning

Ishaan Singh
Department of Computer Science
University of Illinois Urbana-Champaign
Urbana, IL, USA
is14@illinois.edu

Vishal Moorjani
Department of Computer Engineering
University of Illinois Urbana-Champaign
Urbana, IL, USA
vishal6@illinois.edu

Aum Wadhwani
Department of Computer Science
University of Illinois Urbana-Champaign
Urbana, IL, USA
aumw2@illinois.edu

Abstract

Traditional Reinforcement Learning from Human Feedback (RLHF) methods often employ a single reward model that averages diverse user preferences, potentially leading to inaccuracies and misalignment for individual users, especially those with minority viewpoints. Variational Preference Learning (VPL) [Poddar et al., 2024] addresses this by inferring a user-specific latent variable to effectively model multi-modal preferences. However, VPL's reliance on a static latent variable limits its applicability in long-horizon tasks or scenarios where user preferences evolve over time. We introduce Temporal VPL (T-VPL), an extension that models user preferences as a sequence of latent states $\{z_t\}$ that dynamically change. T-VPL integrates a transition model $p(z_t|z_{t-1})$ and optimizes a sequential Evidence Lower Bound (ELBO) to capture these preference dynamics. Our experimental evaluations on modified versions of the Simple-Pets dataset demonstrate T-VPL's enhanced ability to adapt to shifts in preferences, and understand rather than memorise patterns when compared to static VPL.

1 Introduction

Reinforcement Learning from Human Feedback (RLHF) has emerged as a powerful paradigm for aligning large-scale models, particularly Large Language Models (LLMs), with human values and intentions [Poddar et al., 2024]. Conventional RLHF techniques typically learn a single, unimodal reward model from a diverse pool of human preference data. While effective in capturing general tendencies, this averaging approach can dilute individual nuances and fail to cater to users with specific or minority preferences, potentially leading to generic or misaligned model behaviors.

To address the challenge of preference heterogeneity, Poddar et al. [2024] proposed Variational Preference Learning (VPL). VPL learns a multi-modal reward model by conditioning on a user-specific latent variable z, inferred from a small set of that user's preference annotations. This approach has demonstrated improved reward prediction accuracy and downstream task performance in both robotics and LLM applications.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

However, the static nature of the latent preference variable z in VPL presents a limitation. Human preferences are often not fixed; they can evolve due to changing goals, tastes, learning, or varying contextual factors over longer interaction horizons. A static z may fail to capture such temporal dynamics, leading to outdated or inaccurate personalization.

In this paper, we introduce Temporal Variational Preference Learning (T-VPL), an extension of VPL designed to explicitly model and adapt to dynamic user preferences. T-VPL represents user preferences as a sequence of latent states $\{z_t\}$, where each z_t captures the preference profile at a given timestep t. The evolution of these states is governed by a learned transition model, enabling the system to continuously refine its understanding of user preferences as new data arrives. Our primary contributions are:

- The formulation of T-VPL, a novel framework for learning personalized and temporally evolving reward models.
- An adapted Variational Autoencoder (VAE) architecture incorporating a transition model and a sequential ELBO objective for training (detailed in Section 3.1, based on `vpl_llm/hidden_context/temporal_vae_utils.py`).
- Experimental validation demonstrating T-VPL's ability to track and adapt to preference shifts, outperforming static VPL on Sequential Simple-Pets.

2 Background: Variational Preference Learning (VPL)

VPL [Poddar et al., 2024] addresses the limitations of unimodal reward models in RLHF by learning to infer a user-specific latent preference vector z. This vector z is intended to capture the underlying factors that differentiate one user's preferences from another's (e.g., helpfulness vs. harmlessness priorities). The reward for a given response x is then modeled as $R(x|z)$, conditioned on this latent user representation.

The VPL framework typically uses a VAE (Variational Autoencoder) structure. An encoder network, $q_\phi(z|D_{\text{user}})$, takes a set of preference pairs $D_{\text{user}} = \{(c_i, r_i)\}_{i=1}^N$ (chosen and rejected responses) from a specific user and outputs the parameters (mean and variance) of an approximate posterior distribution over z. A prior, usually a standard Gaussian $p(z) = \mathcal{N}(0, I)$, is assumed for z. A decoder network, $p_\theta(D_{\text{user}}|z)$, then predicts the likelihood of the observed preferences given z. For binary preferences, this often involves predicting rewards r_c and r_r for the chosen and rejected options, with the probability of preference being $\sigma(r_c - r_r)$.

3 Temporal Variational Preference Learning (T-VPL)

The core assumption of VPL is that a user's preference latent z is static once inferred from a batch of their data. T-VPL relaxes this assumption by modeling z as a time-varying quantity z_t, allowing for adaptation to evolving preferences.

3.1 Framework

In T-VPL, we consider a sequence of preference data arriving over time, where $D_t = \{(c_i, r_i)\}_t$ represents the preference pairs observed at timestep t. The user's preference at time t is captured by a latent state z_t. The evolution of this state is modeled by a **transition model** $p_\pi(z_t|z_{t-1})$, which defines a prior for z_t based on the previous state z_{t-1}. This transition model can be a simple Multi-Layer Perceptron (MLP) or a more complex recurrent structure like a GRU or LSTM, as implemented in `TransitionModel` within `vpl_llm/hidden_context/temporal_vae_utils.py`.

The generative process at each timestep t involves:

1. Sampling the current latent state $z_t \sim p(z_t|z_{t-1})$ (or from an initial prior $p(z_0)$ for $t = 0$).
2. Generating preference outcomes D_t based on z_t.

For inference, we aim to estimate the sequence of latent states. The T-VPL model employs the following components at each timestep t:

- **Pair Encoder**: Encodes the current preference evidence D_t (specifically, embeddings of c_i and r_i) into an evidence vector e_t. This is analogous to the input processing for the encoder in static VPL but operates on the current timestep's data. Implemented as `PairEncoder` in `vpl_llm/hidden_context/temporal_vae_utils.py`. The input embeddings themselves might come from pre-computed LLM embeddings of text sequences.
- **Posterior Network**: Approximates the true posterior $p(z_t|z_{t-1}, D_t)$ with $q_\psi(z_t|\text{prior\_params}(z_{t-1}), e_t)$. It combines the prior information propagated by the transition model with the current evidence e_t. Implemented as `PosteriorNet` in `vpl_llm/hidden_context/temporal_vae_utils.py`.
- **Decoder**: Predicts the rewards $R(c_i|z_t)$ and $R(r_i|z_t)$ (and thus the likelihood of D_t) conditioned on the inferred latent state z_t. Implemented as `Decoder` in `vpl_llm/hidden_context/temporal_vae_utils.py`.

The entire `TemporalVAEModel` is trained by maximizing a sequential ELBO, summed over the timesteps T of an interaction sequence:

$$\mathcal{L}_{\text{T-VPL}} = \sum_{t=1}^{T} \left(\mathbb{E}_{q_\psi(z_t|...)}[\log p_\theta(D_t|z_t)] - \beta \cdot \text{KL}(q_\psi(z_t|...)||p_\pi(z_t|z_{t-1})) \right)$$

where $q_\psi(z_t|...)$ denotes $q_\psi(z_t|\text{prior\_params}(z_{t-1}), e_t)$. The KL divergence term now penalizes deviation from the transition prior, encouraging smooth temporal evolution while allowing updates based on new evidence. This sequential processing is handled within the forward pass of the `TemporalVAEModel`.

3.2 Data Handling and Training

Input data for T-VPL consists of sequences of preference pairs. Scripts like `vpl_llm/hidden_context/data_utils/create_temporal_pets.ipynb` and `vpl_llm/hidden_context/data_utils/create_4_class_pets.ipynb` are used to transform flat data sets into sequential data sets, where each sample contains a sequence of chosen and rejected responses and their embeddings for a defined 'seq_len'. The `TemporalSequenceCollator` (`vpl_llm/hidden_context/temporal_collator.py`) is responsible for batching these sequences and padding them to uniform lengths for model input. Training is managed by `TemporalVAETrainer`, which adapts the standard Hugging Face Trainer for this sequential VAE model, incorporating KL annealing and logging relevant temporal metrics. The original VPL model required the data set to store one pair of accepted/rejected responses. In the script `vpl_llm/hidden_context/data_utils/data_processing.py` the context (previous preference annotations) is randomly sampled as the context for the current responses. The script `vpl_llm/train_vae_pets.py` uses the `TemporalSequenceCollator` and then adapts the data into the format expected by the VAEModel using the `SequenceVAECollator`.

4 Experimental Setup

We conducted experiments to evaluate T-VPL's ability to model dynamic user preferences and to compare its performance against static VPL and other established baselines. Our setup focused on sequential preference data to simulate evolving user interactions.

4.1 Datasets

The Pets dataset used in the original paper consists of chosen/rejected pairs sampled from 100 synthetically generated sentences describing four types of pets: birds, dogs, cats, and rabbits. Two user groups are defined, both preferring birds the most and rabbits the least, but differing on the relative preference for dogs versus cats (Group 1 prefers dogs to cats, while Group 2 prefers cats to dogs). Each user is assigned several pairs of different pets with corresponding sentences, labeled as "chosen" or "rejected" according to their group's preference order.

While this dataset does capture divergent preferences, it exhibits several limitations. First, there are only two user groups and a single axis of divergence (dogs versus cats), meaning a model can simply

memorize the rule to accurately predict all dog/cat pairs. Second, the dataset never features users whose preferences change over time, so it is not possible to assess model behavior in the face of shifting preferences (e.g., a user who initially prefers dogs to cats but later reverses this preference). Third, both groups always prefer birds and rabbits as their most and least favored pets, respectively, which reduces the total number of divergent preference cases from six to two and primarily tests memorization rather than more flexible reasoning.

To address these shortcomings, we adapted the Pets dataset to better evaluate models on preference change and a broader set of divergent scenarios.

- **Sequential Simple Pets:** This variant greatly expands the space of divergent preferences by eliminating fixed user groups and not constraining birds and rabbits to always be the most and least preferred pets. Each user's initial preference order is chosen at random. For each subsequent chosen/rejected pair (i.e., timestep), a probability parameter α determines whether the user's preferences change. Additionally, at each timestep, we check if the user's current chosen/rejected sequence matches any other user's history. If so, the probability of preference change is reduced, encouraging the emergence of clusters of similar but not rigidly defined preferences. This dataset challenges models to recognize evolving preference patterns and discourages simple memorization, due to both greater randomness and the increased number of possible permutations.

4.2 Models and Baselines

All reward models, including T-VPL variants and baselines, were trained using `gpt2` for generating the initial preference pair embeddings.

We compared our proposed T-VPL, with different transition model architectures, against several baselines:

- **T-VPL (MLP)**: This version of our temporal variational preference learning model, detailed in Section 3.1, employs a Multi-Layer Perceptron (MLP) as the transition model $p_\pi(z_t|z_{t-1})$. The MLP takes the previous latent state z_{t-1} and outputs the parameters (mean and log-variance) for the prior distribution of the current latent state z_t. This architecture is the default in our `TransitionModel` implementation within `vpl_llm/hidden_context/temporal_vae_utils.py` when no specific `transition_type` is passed during `TemporalVAEModel` initialization in `vpl_llm/train_temporal_pets.py`.
- **T-VPL (GRU)**: This variant utilizes a Gated Recurrent Unit (GRU) cell for the `TransitionModel`. The GRU is designed to capture sequential dependencies more effectively than a simple MLP by maintaining a hidden state that is updated based on z_{t-1}. Specifically, the GRU cell takes z_{t-1} and its previous hidden state h_{t-1} to produce h_t, which is then linearly transformed to output the parameters for the prior distribution of z_t. This allows the model to potentially capture longer-range temporal patterns in preference evolution. To test this, the `transition_type` parameter of the `TemporalVAEModel` was set to `"gru"`.
- **T-VPL (LSTM)**: Similar to the GRU variant, we implemented a Long Short-Term Memory (LSTM) cell for the `TransitionModel`. The LSTM cell utilizes a more complex gating mechanism involving cell state and hidden state, offering another way to model temporal dependencies in the latent preference sequence z_t. This can be called by setting the `transition_type` to `"lstm"`.
- **Static VPL**: The original VPL model by Poddar et al. [2024], which infers a single, static latent preference vector z. For fair comparison with T-VPL on sequential datasets, Static VPL was trained by adapting the sequential dataset, to use timesteps [0, seq_len - 1] as the context (accepted / rejected pairs) for the sample. This baseline uses the architecture from `vpl_llm/hidden_context/vae_utils.py`.

All T-VPL variants share the same overall architecture described in Section 3.1 (Pair Encoder, Posterior Network, Decoder), differing only in the internal mechanism of the `TransitionModel`. Key hyperparameters for T-VPL, such as `latent_dim`, `hidden_dim`, `evidence_dim`, and `kl_loss_weight`, were tuned based on validation performance, with values specified in train-

ing scripts like `vpl_llm/train_temporal_pets.py` and `vpl_llm/train_vae_pets.py`. For instance, `evidence_dim` is a newly added script argument for the temporal model configuration.

4.3 Evaluation Metrics

The performance of T-VPL and baselines was evaluated on the following metrics:

- **Reward Model Accuracy**: The percentage of correctly predicted preferences on held-out sequential test data. A correctly predicted preference is determined by a "chosen" sequence being predicted at a higher probability than the "rejected" sequence.

5 Results and Discussion

5.1 Quantitative Analysis

We present a comparative analysis of T-VPL against static VPL and other baselines across our evaluation metrics.

Table 1: Reward Prediction Accuracy on Sequential Datasets.

MODEL	PETS ($\alpha = 1$)	PETS ($\alpha = 0.8$)	PETS ($\alpha = 0.5$)
STATIC VPL	49.5%	54.0%	47.5%
T-VPL (MLP)	96.5%	95.5%	97.0%
T-VPL (GRU)	96.0%	98.0%	96.0%
T-VPL (LSTM)	95.0%	95.5%	98.0%

5.2 Discussion of Results

Table 1 presents the reward prediction accuracy for both static and temporal VPL models across the Pets dataset with varying rates of preference shifts (as controlled by the α parameter). Across all conditions, the temporal VPL models (T-VPL MLP, GRU, LSTM) consistently outperform the static VPL baseline, with the largest improvements observed on datasets where user preferences change over time. Notably, while the static VPL model achieves only 49.5%–54.0% accuracy on these datasets, the temporal models reach accuracies as high as 98.0%.

Interestingly, the choice of temporal model architecture—whether MLP, GRU, or LSTM—appears to have only a marginal impact on overall performance, with all T-VPL variants achieving similarly high accuracy across the different datasets. This suggests that the key performance gains arise primarily from modeling temporal dynamics, rather than from specific recurrent or feedforward mechanisms. Additionally, there is no clear monotonic relationship between the α parameter (which controls the rate of preference change) and prediction accuracy. While all temporal models perform substantially better than the static baseline for all values of α, their relative performance remains stable, and accuracy does not consistently increase or decrease as α varies. This indicates that once temporal modeling is incorporated, the models are robust to different patterns and frequencies of preference evolution.

While we might expect the static VPL model to perform best when preferences are fixed ($\alpha = 1$), its slightly lower accuracy at $\alpha = 1$ compared to $\alpha = 0.8$ is likely due to minor statistical fluctuations or sampling effects. In principle, the static model is best suited to scenarios with stable user preferences, and its performance in these settings generally reflects this expectation.

6 Related Work

Our work builds upon several lines of research. **Personalized RLHF:** Beyond VPL [Poddar et al., 2024], other works have explored personalization in RLHF by methods such as multi-head reward models or conditioning on explicit user profiles. However, most do not explicitly model temporal preference dynamics. **Modeling User Preference Dynamics:** Fields like recommender systems [Koren et al., 2009] and user modeling in dialogue systems have a rich history of tracking evolving

user interests, often using recurrent neural networks or state-space models. T-VPL brings similar concepts to the preference modeling stage within RLHF. **Continual Learning and RL:** Adapting to non-stationary reward functions is a known challenge in RL. While T-VPL focuses on the reward model, it shares goals with continual learning approaches that aim for robust adaptation in changing environments.

7 Conclusion and Future Work

We have introduced Temporal Variational Preference Learning (T-VPL), an extension to VPL that explicitly models and adapts to dynamic user preferences by representing them as an evolving sequence of latent states $\{z_t\}$. Our proposed framework, incorporating a transition model and a sequential ELBO, demonstrates [reiterate your key positive findings, e.g., improved accuracy in dynamic settings and better alignment with evolving preferences on tasks like X and Y].

Future research directions include exploring more sophisticated transition dynamics, such as Bayesian filtering mechanisms as initially proposed, to provide better uncertainty estimates for preference evolution. Investigating the integration of an active learning component, where the system queries for feedback upon detecting significant divergence in latent states, could further enhance T-VPL's adaptability. Applying T-VPL to more complex, real-world, long-horizon interaction scenarios and extending it to handle richer, non-binary forms of human feedback are also promising avenues.

We would also like to test models on another dataset that introduces some changing preferences but doesn't include as many permutations or randomness. The dataset would be a direct extension of Pets, with four user groups instead of two. Each user is given a context of eight chosen/rejected response pairs. The first two groups are the same as in the original Pets dataset (Group 1 prefers dogs to cats; Group 2 prefers cats to dogs). The remaining two groups feature a change in preference: Group 3 starts with preferences identical to Group 1 (preferring dogs to cats), but randomly between the third and seventh chosen/rejected pair, flips to preferring cats to dogs. Group 4 follows a similar pattern as Group 3, but flips from preferring cats to dogs to preferring dogs to cats. The purpose of this dataset would be to mimic basic preference changes and test a model's ability to detect and adapt to them.

Individual Contributions

Ishaan: I worked on designing the core `TemporalVAEModel`, including the `TransitionModel` (incorporating MLP, GRU, and LSTM architectures) to capture dynamic user preferences via latent states z_t. I also worked on the initial data pipeline for T-VPL, developing scripts like `vpl_llm/hidden_context/data_utils/generate_complex_data.py` to transform standard preference datasets into the sequential format required for temporal analysis and generating initial sequence embedding datasets (e.g., "simple_pets_seq" via `vpl_llm/generate_llm_seq_embeddings_pets.sh`). To do this, I needed to build the `TemporalSequenceCollator` for batching sequential data and adapting the training loop into the `TemporalVAETrainer` to optimize the sequential ELBO objective.

Vishal: I worked on creating the data collators to use the sequential datasets with the SequenceVAECollator. I also worked on generating the preference datasets (`vpl_llm/hidden_context/data_utils/create_temporal_pets`). I worked on the scripts to train and evaluate the models (`vpl_llm/train_temporal_pets.py`, `vpl_llm/train_vae_pets.py`, `vpl_llm/evaluate_models.py`). I had also worked on a script to show sample sequences to the user, ask them for their preference annotations, and train a VPL model on these preference annotations to score a set of responses. This gave us a qualatative understanding of the VPL models strengths and weaknesses while working on our extension.

Aum: I worked on designing and generating the synthetic preference datasets used in our experiments. I adapted and implemented the temporal training scripts to ensure compatibility with our new dataset structure. I also worked on modifying the original paper's evaluation pipeline to work for our temporal model and mesh together with a new data format.

All authors contributed significantly to the experimental design, analysis, and the primary writing of this report.

References

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263.

Sriyash Poddar, Yanming Wan, Hamish Ivison, Abhishek Gupta, and Natasha Jaques. Personalizing reinforcement learning from human feedback with variational preference learning. *arXiv preprint arXiv:2408.10075*, 2024. URL `https://arxiv.org/abs/2408.10075`.

A Appendix: Implementation Details

A.1 Model Hyperparameters

Key hyperparameters used for training T-VPL and baselines are detailed in Table 2. These were primarily sourced from the configurations in our execution scripts (e.g., `vpl_llm/submit_job_pets.sh`, `vpl_llm/train_llm_vae_preference_model.py`).

Table 2: Key Hyperparameters for T-VPL Training.

PARAMETER	VALUE
POLICY MODEL NAME (`MODEL_NAME`)	`GPT2`
LLM ENCODER FOR VAE	`GPT2`
LATENT DIMENSION (`LATENT_DIM`)	64
HIDDEN DIMENSION (`HIDDEN_DIM`)	256
ENCODER EMBEDDING DIM (`ENCODER_EMBED_DIM`)	768
DECODER EMBEDDING DIM (`DECODER_EMBED_DIM`)	768
EVIDENCE DIMENSION (`EVIDENCE_DIM`)	128
TRANSITION MODEL TYPE	MLP (DEFAULT)
KL LOSS WEIGHT (`KL_LOSS_WEIGHT`)	1×10^{-2}
NUMBER OF EPOCHS (`NUM_TRAIN_EPOCHS`)	3
SEQUENCE LENGTH (`SEQ_LEN`) FOR PETS	8
KL ANNEALING (`USE_ANNEALING`)	TRUE

A.2 Dataset Generation for Temporal Analysis

Sequential datasets were primarily generated using the `vpl_llm/hidden_context/data_utils/create_temporal_pets.ipynb` python notebook. This notebook loads in the sample sentences in `vpl_llm/hidden_context/data_utils/simple_templates.py` and generates the sequential dataset. It loads in the GPT-2 model for sequence classification (however, since we are using the last token's last-hidden states, the head, be it classification or language modelling, is immaterial), applies the chat template, and creates the embedding for each prompt-response pair. It saves the created dataset to the specified output path.

A.3 Software

The implementation relies on PyTorch, Hugging Face Transformers, PEFT, and TRL libraries, as listed in `vpl_llm/requirements.txt`. All experiments were logged using Weights & Biases (`WANDB_MODE=offline` mode).