PROTORAIL: A RISK-COGNIZANT IMITATION AGENT FOR ADAPTIVE VCPU OVERSUBSCRIPTION IN THE CLOUD

Lu Wang [*1] **Mayukh Das** [*2] **Fangkai Yang** [1] **Bo Qiao** [1] **Hang Dong** [1] **Si Qin** [1] **Victor Ruehle** [3] **Chetan Bansal** [4] **Eli Cortez** [4] **Inigo Goiri** [4] **Saravan Rajmohan** [4] **Qingwei Lin** [1] **Dongmei Zhang** [1]

ABSTRACT

Safe optimization of operating costs is one of the holy grails of successful revenue-generating cloud systems and capacity/resource efficiency is a key factor in making that a reality. Among other strategies for resource efficiency across major cloud providers, Oversubscription is an extremely prevalent practice where more virtual resources are offered than actual physical capacity to minimize revenue loss against redundant capacity. While resources can be of any type, including compute, memory, power or network bandwidth, we highlight the scenario of virtual CPU (vCPU) oversubscription since vCPU cores are primarily the billable units for cloud services and has substantial impact on business as well as users. For a seamless cloud experience, while being cost-efficient for the provider, suitable policies for controlling oversubscription margins are crucial. Narrow margins lead to redundant expenditure on under-utilized resource capacity, and wider margins lead to under-provisioning where customer workloads may suffer from resource contention. Most oversubscription policies today are engineered either with tribal knowledge or with static heuristics about the system, which lead to catastrophic overloading or stranded/under-utilized resources. Smart oversubscription policies that can adapt to demand/utilization patterns across time and granularity to jointly optimize cost benefits and risks is a non-trivial, largely, unsolved problem. We address this challenge with our proposed novel Prototypical Risk-cognizant Active Imitation Learning (PROTORAIL) framework that exploits *approximate symmetries* in utilization patterns to learn suitable policies. The *active knowledge-in-the-loop* (KITL) module de-risks the learned policies. Our empirical investigations and real deployments on Microsoft's internal (1^{st} party) cloud service, show orders of magnitude reduction ($\approx\geq 90\times$) in risk and significant increase in benefits (saved stranded resources: in a range of ≈ 7 to 10%).

1 INTRODUCTION

Oversubscription is popular and widely used practice across the professional services (logistics, travel, cloud etc.) industry. It is the scenario where a system offers more resources or services, such as virtual compute resources, to users or entities than its available physical capacity, assuming not all users would simultaneously fully utilize the allocated capacity. In cloud services, Virtual Machines (VMs) having some quantum of virtual resources such as virtual cores/memory etc. are hosted on Physical Machines (PMs). Oversubscription here allocates fewer resources for a VM than the requested amount, assuming that a VM may use an extra margin beyond its allocated amount if needed (as shown in Fig 1). As a result the platform can leverage unused physical capacity by packing more VMs as an effective way to avoid unnecessary resource wastage and maximize profits (Baset et al., 2012; Breitgand & Epstein, 2012).

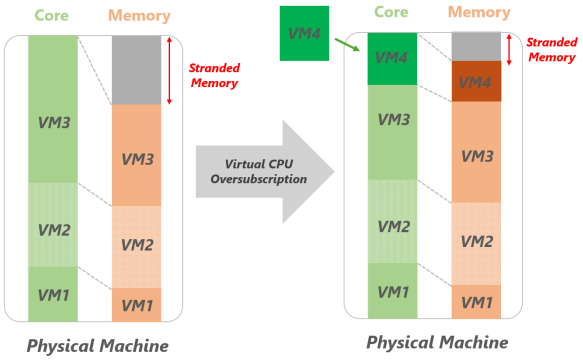


Figure 1. vCPU oversubscription example. After VCPU oversub. for VM3, one more VM (VM4) can be allocated in the physical machine and the stranded memory is reduced (Wang et al., 2024).

CPU bottlenecks are impactful in cloud (Mahapatra & Venkatrao, 1999) vCPUs are the sellable billing units. As illustrated in Fig. 1, three VMs are placed in the same node (PM), where we observe with unused memory, *i.e.*, stranded

[1]Microsoft, Beijing, China [2]Microsoft, Bangalore, India [3]Microsoft, Cambridge, UK [4]Microsoft, Redmond, USA. Correspondence to: Mayukh Das <mayukhdas@microsoft.com>, Lu Wang <wlu@microsoft.com>.

Proceedings of the 8^{th} MLSys Conference, Santa Clara, CA, USA, 2025.

memory (grey box) even when CPU dimension is fully packed. If oversubscribed, additional VMs can be allocated reducing stranded memory. Hence, we focus on vCPU oversubscription. However, designing a suitable oversubscription policy is both crucial and challenging— once a system is oversubscribed, overloading or under-utilization may happen (Baset et al., 2012) arbitrarily.

Challenges: **(1) Benefits vs Risk** - Aggressive policy leads to denser VM packing and may cause arbitrary *overloading* in cloud (Williams et al., 2011),leading to higher risk towards user experience and revenue losses through compensations (Wittman, 2014). Conservative policies result in redundant capacity and inefficient resource usage (wastage). **(2) Dynamic patterns** - Demand and utilization patterns vary across time (VMs, users, services/groups). Static policies cannot adapt to such dynamic load patterns (Cortez et al., 2017). Designing adaptive policies is also not trivial since forecasting just user demand pattern is not enough. **(3) Policy Granularity** - Demand may vary with granularity, i.e. at what level the oversubscription is enforced (per VM, subscriptions/users, or user groups).

The problem: We address the problem of vCPU oversubscription by alleviating overloading risks while maximizing utilization against temporally varying demand patterns across time and granularity. Existing research usually deals with specific scenarios instead of principled generalized formalism. Some approach oversubscription in cloud platforms as online constrained bin-packing problem (Baset et al., 2012; Breitgand & Epstein, 2012; Householder et al., 2014), focusing more on resource allocation rather than oversubscription policy. Others propose migration strategies to mitigate overload situations (Wang & Tianfield, 2018; Li, 2019). Yet, a generalized oversubscription policy, that addresses the above challenges is under-explored.

Oversubscription, resource allocation, scheduling, and optimal packing are all related concepts in resource management. However, they each address different aspects of the problem. Oversubscription occurs when the demand exceeds the available supply, whereas resource allocation and scheduling deal with management and mapping of demand to resources within a system. Optimal packing, on the other hand, is an approach that seeks to minimize fragmentation.

Proposed Solution: In this paper, we pose the adaptive oversubscription question as a sequential decision-making problem with resource limits. Predicting future utilization behaviors given historical observations through traditional supervised learning approaches is insufficient since they are unaware of the interactions between the users and the environments (De Haan et al., 2019). Alternatively, traditional online reinforcement learning (RL) with constraints (Garcıa & Fernández, 2015), it is challenging to optimize different competing objectives with convergence guarantees, due to non-convexity (Achiam et al., 2017; Paternain et al., 2019; Mazyavkina et al., 2021). *Also, it is not practically feasible to train RL policies online on deployed systems.* Imitation learning (IL), however, can solve MDP constraint problems (Hussein et al., 2017) from offline telemetry where the expert's policy induces the constraints by nature. In particular, we propose a prototypical imitation learning method (PROTORAIL) that learns to take actions by a set of learned *prototypes*, where prototype is a data instance that is representative of an equivalence class of expert trajectories (Kim et al., 2016; Molnar, 2020). Hence, our approach can leverage approximate symmetries in patterns, allowing us to learn adaptive policies for any granularity. One caveat is that the utilization data is usually noisy or sparse, resulting in sub-optimal prototypes and policies. Facilitated by the interpretability of prototypes, we leverage efficient *active* knowledge-in-the-loop infusion to de-risk the policies against overloading.

Contributions and Impact: We make the following contributions: **(1)** We propose a novel prototypical imitation learning approach to solve the vCPU oversubscription ratio prediction problem for cloud system to maximize utilization (COGS[1]) efficiency and minimizing risk; **(2)** Efficient active knowledge-in-the-loop (KITL) training technique mitigates overloading risk from systematic noise; **(3)** Extensive evaluations, both offline and in practice on Microsoft's internal cloud, show how PROTORAIL learns oversubscription policies with $\approx 0\%$ risk and ≈ 7 to 10% higher benefit; **(4)** We also show how our PROTORAIL seamlessly generalizes to other domains beyond cloud services. Airline ticket overbooking dataset is curated from static periodic reports of U.S. Department of Transportation (DOT).

2 PRELIMINARY AND BACKGROUND

2.1 Optimal oversubscription problem

We formally describe optimal oversubscription problem as identifying optimal oversubscription rate $\zeta_e^* | 0 \leq \zeta_e \leq 1$, where $\zeta_e = \mathcal{A}_e / \Re_e$ is the fraction of requested virtual resources ($\Re_e$) by an entity $e \in E$ (a VM, or a user) that is actually allocated ($\mathcal{A}_e$) by the cloud platform. We need to optimize on two competing objectives, "COGS benefit" β and "overloading risk" χ.

Semantically, benefit β is a function of denser VM packing and reduction in total provisioned nodes due to more harvestable stranded vCPUs made available better ζ. But that cannot be computed at offline training time. We present a simpler formulation tracking cost of physical CPU cores wrt harvested vCPUs, $\beta(\{\zeta_e\}_{e \in E}) = \sum_{e \in E} (1 - \zeta_e).\Re_e \times k \times \mathcal{C}$, where $k \in \mathbb{R}$ is physical cores per vCPU and $\mathcal{C}$ is the cost

[1] Cost Of Goods Sold a well-known concept in service/product industry (Franklin et al., 2019)

of a physical core. The risk of overloading arises if ζ is too low, causing the allocator to pack too many requests into a node, leaving no room for scaling up in case multiple VMs peak simultaneously. Thus risk $\chi(\{\zeta_e\}_{e\in E}) = \sum_{\eta\in\mathbb{N}} \mathbb{I}(\sum_{e\in E_\eta} \mathcal{U}_e.k \geq size(\eta))$ is the sum of node overload indicators, where $\mathcal{U}_e$ is the usage rate and $\mathbb{N}$ is the set of nodes. Hence, the optimization problem becomes,

$$\zeta^* = \arg\max_{\{\zeta_e\}} \beta(\{\zeta_e\}_{e\in E}) + \arg\min_{\{\zeta_e\}} \chi(\{\zeta_e\}_{e\in E}) \quad (1)$$

There is **no closed form solution** to this problem. Also, ζ^* is not one value but adaptive values across time based on demand patterns and granularity. We could learn a policy $\pi^* = P(\zeta^*|usage, time, \ldots)$ via online RL. But online RL methods (value-based or policy gradient) maximize some expected cumulative return $\max_{a\sim\pi} \mathbb{E}_{s\sim T(s|s,a)}[\sum_{t=t}^{T} R_t|S = s, A = a]$. In our context, we cannot reliably and efficiency compute/sample the return/reward (multi-objective combinatorial β vs χ) online. We also cannot safely train RL policies online on deployed live cloud. We leverage imitation learning based, PROTORAIL, to learn the policy from offline telemetry.

2.2 Prototype theory and its importance

Prototype theory emerged with the work of psychologist Eleanor Rosch (Rosch, 1973). In prototype theory, any given concept has a real-world example that best represents it. For instance, for the concept *fruits*, an apple is more frequently cited than a durian. Hence the presumed natural prototypes are central tendencies of the categories. Prototype theory has been applied in machine learning (Kim et al., 2016), where a prototype indicates a data instance that is representative of an equivalence class of samples (Molnar, 2020).

While there are many approaches to find prototypes, any clustering algorithm that returns actual data points as cluster centers is a good approximation. Harel *et al.* (Harel & Radinsky, 2018) uses generative model-based clustering to assign central points as prototypes. ProSeNet (Ming et al., 2019) leverages constraints to learn simple, diverse, and sparse prototypes with a sequence encoder which loosely inspires our work. However, to our best knowledge, leveraging prototypes in IL is a novel area and can help achieve ideal oversubscription granularity and better interpretability.

Why? In real cloud systems, there is no real ground truth on ideal oversubscription. Hence we exploit historical workload/VM telemetry as expert (training) trajectories, and CPU usage as proxy labels, assuming ($\zeta_e \propto \mathcal{U}_e$). *However, usage/demand patterns vary across granularity and impact policy choice*. Key idea is to smartly leverage different equivalence classes of approximately symmetric vCPU usage patterns via prototypes. Oversub. policy is probabilistic estimate based on nearest prototype's policy for given entity/context. IL with prototypes allow for interpretable and effective oversubscription.

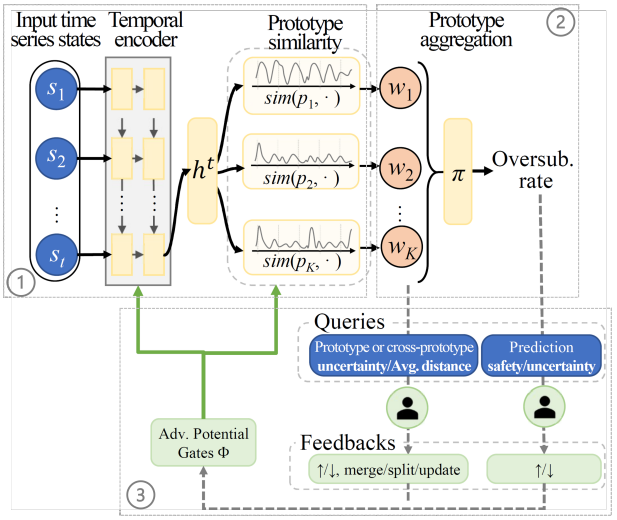


Figure 2. PROTORAIL overview

2.3 Active Knowledge-guided learning

Why? Sample sparsity or label noise (our case) result in highly sub-optimal models and hence *arbitrary risk*. Knowledge-guided training leverage domain knowledge beyond data for better risk cognizance. The most generic formulation of knowledge is an inductive bias (π_{know}) that pushes sub-optimal models (distributions) learned from noisy observations towards, possibly unknown, true distribution; $\pi^*(\cdot) = \alpha\pi_{data}(\cdot) \oplus (1-\alpha)\pi_{know}(\cdot)$, where π^* is the optimal distribution and α is a trade-off.

Such knowledge may have 2 modes — (1) passive priors or constraints, (2) active feedback during the training which we adopt, as it is difficult to design/compute priors/constraints in prototypical IL. While conceptually motivated from active learning (Cohn et al., 1994), we (and other related works (Neider et al., 2021; Brown et al., 2018; Deng et al., 2020)) show that the knowledge we acquire is **richer** than mere data labels. Knowledge can be represented in many ways, such as explanations (Minton et al., 1989), preference rules (Odom & Natarajan, 2015; Das et al., 2021), pseudo-labels (Jiang et al., 2018; Goldberger & Ben-Reuven, 2017; Patrini et al., 2017; Miyato et al., 2018; Lee, 2013). We represent feedback with simple voting on predictions and prototypes making our approach efficient and interpretable. We infuse such knowledge into training via loss scaling inspired from KCLN (Das et al., 2021). Knowledge can be elicited from either domain experts (humans) as seen in above mentioned literature or, more recently, from generative models such as LLMs (Du et al., 2023; Ma et al., 2024; Cao et al., 2024).

3 METHOD

3.1 Problem Formulation

We formulate the oversubscription problem in Cloud as a prototypical imitation learning problem. Since there there are no true labels on oversubscription rate we assume the CPU usage as proxy and use usage trajectories $\tau_{e \in E}$ for training. Given input trajectory till timestamp t, $\tau_t^e = (s_1, a_1, ..., s_{t-1}, a_{t-1}, s_t)$, The base IL problem is $\mathcal{L}_{IL} = D(\pi(\tau_t^e)||\pi(\theta))$ where $\pi(\theta)$ is the learned policy. and $\pi(\tau^e)$ is the expert policy induced by training data. The expert policy distribution can be different at different granularities. *For example, CPU usage patterns of similar distribution across different VMs/users/groups etc. or co-located in the same region or serve same customer base*. Thus trajectories representing equivalence classes of CPU usage patterns with approximate symmetry can be considered as different prototypes. Prototypical imitation learning aims to identify these usage prototypes, then predict ideal oversubscription via referring the nearest prototypes based on the current context.

Definition 1 (Prototypical Imitation Learning)**.** Formally prototypical imitation learning as a kind of IL that learns an oversubscription policy π_θ by aligning with a reference prototype trajectories $\arg\min_{p_i \in \mathcal{P}} \pi_\theta \underset{Distance}{\rightsquigarrow} \pi_{p_i}$. Each prototype $p_i \in \mathcal{P}$ intuitively represents equivalence classes of patterns and prototypical IL learns a metric space in which decision-making is conditioned on the distance to the policies induced by prototypes. New prediction is computed and explained by closest prototype trajectories.

3.2 Overview of Prototypical Imitation Learning

The architecture of Prototypical Risk-cognizant Active Imitation Learning (PROTORAIL) is shown in Fig 2. There are three main components: (1) discover prototypes, where we classify the trajectories into K groups and learn prototypical representations p_k; (2) learn $\pi_t heta$ by aligning with its similar prototypes at different states; (3) obtaining active feedback from the knowledge-in-the-loop for risk-cognizant policy reducing $\downarrow \chi(\zeta)$ while enhancing benefit $\uparrow \beta(\zeta)$.

For input trajectory τ_t, the trajectory encoder f maps τ_t into an embedding vector $h = f(\tau_t)$, $h \in \mathbb{R}^m$. The encoder can be any sequence encoder, *e.g.*, LSTMs or Transformers. The prototype layer p contains K prototype embeddings $\mathcal{P} = \{p_1, ..., p_K\}$, where each $p_k \in \mathbb{R}^m$ have the same length as h. This layer scores the similarity between h and each prototype p_k. We consider L_2 distance metric as the similarity function for simplicity. With the computed similarity vector $[sim(f(\tau_t), p_1), ..., sim(f(\tau_t), p_K)]$, the policy layer π computes the action with a linear layer with sigmoid activation. In the end, an KITL module is leveraged to refine PROTORAIL for better interpretability and performance by validating and refining the prototypes.

3.3 Prototype Discovery

The *prototype* is defined as the representative instance $\tau^{E,k}$ selected from a class of expert trajectories. Specifically, the trajectory encoder is shared for encoding τ_t and $\tau^{E,k}$ to the same embedding space, *i.e.*, $p_k = f(\tau^{E,k})$ and $h = f(\tau_t)$. Note that $\tau^{E,k}$ is the full expert trajectory and we omit the time subscript. We consider three aspects as part of the objective for learning prototypes (1), *i.e.*, representative, diversity, and interpretability.

For **representative** aspect, we aim to learn prototypes that can well represent a subset of trajectories with the regularization term $\mathcal{L}_{rep}$. It encourages a clustering structure in the embedding space by minimizing the L_2 distance between an encoded trajectory and its nearest prototype embedding:

$$\mathcal{L}_{rep} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|D_k|} \sum_{\tau \in D_k} ||p_k - f(\tau)||_2^2, \quad (2)$$

where D_k indicates a subset of trajectories that could be represented by p_k.

Now, to improve the **diversity** and reduce redundancy among prototypes, the term $\mathcal{L}_{div}$ penalizes the prototypes that are close to each other:

$$\mathcal{L}_{div} = -\frac{1}{{}^K C_2} \sum_{i=1}^{K} \sum_{j=i+1}^{K} ||p_i - p_j||_2^2, \quad (3)$$

where ${}^K C_2 = \frac{K!}{2!(K-2)!}$.

To give **interpretability**, we assign one expert trajectory instance $\tau^{E,k}$ as one prototype via $\mathcal{L}_{int}$. Then, each prototype embedding can be explained by a real-world instance.

$$\mathcal{L}_{int} = \frac{1}{K} \sum_{k=1}^{K} ||p_k - f(\tau^{E,k})||_2^2, \quad (4)$$

where $\tau^{E,k}$ is the nearest expert trajectory instance to p_k, *i.e.*, $\tau^{E,k} = \text{argmin}_{\tau^E \in \mathcal{T}} ||p_k - f(\tau^E)||_2^2$ and $\mathcal{T}$ is the set of expert trajectories.

3.4 Imitation Learning over Prototypes

$\pi(a|\tau_t, \mathcal{P})$ is the policy layer that learns to take an action aligning with the prototypes (2) from the experts' policy:

$$\pi(a|\tau_t, \mathcal{P}) = \phi([sim(f(\tau_t), p_1), .., sim(f(\tau_t), p_K)]) \quad (5)$$

where $\mathcal{P} = \{p_1, ..., p_K\}$ is the set of prototypes embeddings, ϕ is a linear layer with sigmoid activation, and $sim(f(\tau_t), p_k) = -\|f(\tau_t) - p_k\|_2^2$ is the negative Euclidean distance measuring the similarity between the embedded vector and the prototype embedding p_k and $a = \zeta$

We consider prototypical imitation learning with two base imitation learning models to learn the policy, *i.e.*, behavior cloning (BC) and adversarial imitation learning (AIL).

The goal of BC is to mimic the action of the expert at each time step via supervised learning.

$$\mathcal{L}_{IM_{BC}} = \sum_{\tau^E \in \mathcal{T}} \sum_{(s^E, a^E) \sim \tau^E} [\pi^E(a^E|\tau_t^E) \log \pi(a|\tau_t, \mathcal{P})], \tag{6}$$

where τ^E is the expert's trajectory, π^E is the expert policy.

On the other hand, the goal of AIL is to minimize the JS divergence between the expert trajectory distribution and trajectory distribution generated by our policy.

$$\mathcal{L}_{IM_{AIL}} = D_{KL}\left(\rho_\pi \| \frac{\rho_\pi + \rho_{\pi^E}}{2}\right) + D_{KL}\left(\rho_{\pi^E} \| \frac{\rho_\pi + \rho_{\pi^E}}{2}\right), \tag{7}$$

where ρ_π and ρ_{π^E} are discounted occupancy measures of our policy π and the expert policy π^E.

In summary, the **full loss function** we are minimizing is:

$$\mathcal{L}_{Full} = w_1 \cdot \mathcal{L}_{rep} + w_2 \cdot \mathcal{L}_{div} + w_3 \cdot \mathcal{L}_{int} + w_4 \cdot \mathcal{L}_{IM_{loss}} \tag{8}$$

where $\mathcal{L}_{IM_{loss}}$ is the imitation learning loss with either BC or AIL, and $w_1, w_2, w_3, w_4, \in [0, 1]$ are hyper-parameters to balance the weights of the three kinds of loss. We conduct grid-search to determine the value of these hyper-parameters (shown in Appendix).

3.4.1 *Reinterpretation as a Quadratic Model*

The policy π is equivalent to a quadratic model with particular parameterization. By plugging in the similarity function, Equation 5 can be re-written as,

$$\pi = -b_1\|f(\tau_t) - p_1\|_2^2 - ... - b_K\|f(\tau_t) - p_K\|_2^2 \tag{9}$$

where $b_k, k = 1, ..., K$ are the values of the linear neurons in the fully connected layer. By checking each term in π, we can see its linear form:

$$-b_k\|f(\tau_t) - p_k\|_2^2 = -b_k f(\tau_t)^T f(\tau_t) + 2b_k p_k^T f(\tau_t) - b_k p_k^T p_k \tag{10}$$

where the first term is a quadratic term with regard to $f(\tau_t)$, the second term can be treated as $w_k^T f(\tau_t)$ where $w_k = 2b_k p_k$ and the final term can be treated as a constant term with regard to $f(\tau_t)$.

Starting from the above observation, we can treat the action as a summation of K quadratic functions with the same sign in quadratic coefficients with regard to $f(\tau_t)$, which means the relationship between the action and $f(\tau_t)$ can be decomposed to at most two pieces and within each piece the relationship is monotonic. This observation makes our learned policy easier to interpret.

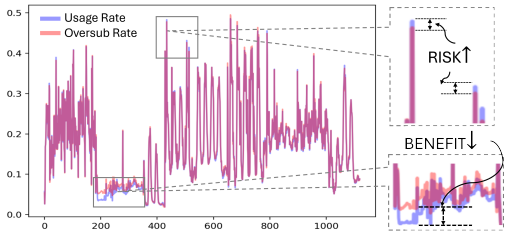


Figure 3. Example plot: Risk increases due to noisy prediction of oversub rate $\leq$ usage or COGS benefit decreases

3.5 Risk-cognizance via active knowledge-in-the-loop

Sample and label sparsity and noise leads to sub-optimal prototype embedding, prototype selection, and policy model resulting in either *high risk* when predicted oversubscription rate goes too close or lower than actual usage or loss of COGS benefit (Fig. 3). Our efficient KITL module (3) leverages additional knowledge via active feedback to refine the learned noisy "policy over prototypes". One essential aspect is that the — **(i)** The query framework must elicit relevant knowledge with minimum query budget $\sum_{t=1}^{T} |\mathcal{Q}_t|$, since – a domain expert's (cloud operator) time and effort is expensive and state-of-the-art LLMs also have token/access budgets or efficiency concerns **(ii)** Appropriate infusion of obtained knowledge into learning to shape policies.

We cannot directly minimize query budget while expecting effective knowledge infusion. We optimize the *query schedule* instead. We implicitly get most relevant knowledge from a knowledge source $\boxed{\kappa}$ with minimum feedback calls. $\boxed{\kappa}$ can be a domain expert with rich understanding of cloud resources and/or LLMs which are effective in reasoning about generic temporal patterns. Active feedback can be of two forms: (1) feedback about the quality of prototypes - embedding, alignment and diversity, (2) feedback about the risk of predicted actions.

3.5.1 *Query Framework (active knowledge elicitation):*

Posing focused queries at relevant points to $\boxed{\kappa}$ is essential for getting useful and relevant knowledge. In a traditional active learning setting, just prediction/label uncertainty is a sufficient heuristic to identify relevant query points. However, in our context, we need richer feedback on both relevant prototypes and output actions. Thus queries at step t is a set of tuples $\mathcal{Q}_t = \{\langle p_{q_t}, a_{q_t}\rangle | p_{q_t} \in \mathcal{P}_{q_t}, a_{q_t} \in \mathcal{A}_{q_t}\}$ that comprises a set of prototypes embeddings $\mathcal{P}_{q_t} \subseteq \mathcal{P}$ and predicted action(s) $\mathcal{A}_{q_{(t)}}$ that need feedback. $\mathcal{Q}_{(t)} = \emptyset$ signifies there is no query at that step. Now $\mathcal{P}_{q_t} = \mathcal{P}_\mu \cap \mathcal{P}_d$ where $\mathcal{P}_\mu$ is defined as the set of uncertain prototypes,

$$\mathcal{P}_\mu = \{p_k \in \mathcal{P} | \mu(p_k) \geq \mathbb{U}_p\},$$
$$\mu(p_k) = \mathbb{H}(P(||f(\tau) - p_k||_2^2)), \tau \in D_k$$

where $\mathbb{U}_p$ is the uncertainty threshold and uncertainty is the prototype cluster entropy $\mathbb{H}$ via the distribution $P(\cdot)$ induced over L2 distances. $\mathcal{P}_d$ comprises prototype embeddings with 'top-N' average distance of the form,

$$\mathcal{P}_d = \arg\max_{\mathbf{p}\subset\mathcal{P}} \left(\frac{1}{|D_k|} \sum\nolimits_{\tau\in D_k} \overline{||f(\tau) - p_k||_2^2} \right), |\mathbf{p}| = N$$

where $\mathbf{p} \models p_k$ is the top-N set. Now, a_{q_t} is identified via both prediction uncertainty $\mu(a_{q_t}) = \mathbb{H}(\pi(a|\tau_t, \mathcal{P})) \geq \mathbb{U}_a$ ($\mathbb{U}_a$ is the action uncertainty threshold), and oversubscription risk which indicates if output action is less than the expert action $a < a^E$. Note that lower oversubscription action/rate means higher risk (cf. Section *Domains(Datasets)*).

> This query formulation is aligned with the interpretable quadratic form of policy (sec. 3.4.1). The square term in trajectory embeddings (Eqn. 10) may lead to higher risk from label noise requiring action feedback. Whereas, feedback on prototypes p_k essentially controls the coefficient of the second term $2b_k p_k^T f(\tau_t)$.

3.5.2 *Feedback representation:*

Given $\mathcal{Q}_t$, $\boxed{\kappa}$ can provide feedback on prototype quality and membership/diversity or on action predictions. Feedback on prototype quality at step t for a prototype embedding p_j is an up/down vote ($\mathfrak{F}(p_k|t) = \uparrow / \downarrow = +1/-1$). Then we use cumulative feedback (over previous iterations) $\mathfrak{F}(p_k) = \sum_{t'=1}^{t} \mathfrak{F}(p_k|t')$. We obtain action feedback $\mathfrak{F}(a)$ in a similar fashion. Feedback on prototype diversity is complicated. For a given pair p_i, p_j, expert may choose to up/down-vote the pair or merge p_i, p_j or split them further. **(1)** If the prototypes are merged, it gets the mean of the embedding vectors of $\vec{p_k} \leftarrow \frac{1}{2}(\vec{p_i} \bigoplus \vec{p_j})$, **(2)** if a prototype p_k is split, the new prototype is initialized with embedding $f(\tau) : \max_{\tau\in D_k} ||p_k - f(\tau)||_2^2$ and retrained.

3.5.3 *Prototype/policy refinement:*

In our context it is difficult to design either inductive bias distribution or proper constraints from feedback. So we allow the feedback to control and scale the loss via exponential 'advice potential gates' (cf. K-CLN (Das et al., 2021)). Advice potentials selectively alter the loss such that training moves in the direction that is expected to produce better model parameters.

Definition 2 (Advice potential gate Φ)**.** An advice potential gate is a product term of an exponential form $\Phi(x) = e^{-\mathcal{G}(\mathfrak{F}(x))}$ where $-\infty \leq \mathfrak{F} \leq +\infty$ is the cumulative feedback. $\mathcal{G}$ scales the unbounded (cumulative) $\mathfrak{F}$ to $[-1, +1]$

With the available feedback over a prototype ($\mathfrak{F}(p_k)$), action ($\mathfrak{F}(a)$) or between prototypes ($\mathfrak{F}(p_i, p_j)$) we modify the loss function of PROTORAIL in Equation 8 as,

$$\begin{aligned}\mathcal{L}'_{Full} =& w_1 \cdot \mathcal{L}_{rep} \times \Phi(p_k) + w_2 \cdot \mathcal{L}_{div} \times \Phi(p_i, p_j) \\ &+ w_3 \cdot \mathcal{L}_{int} \times \Phi(p_k) + w_4 \cdot \mathcal{L}_{IM_{loss}} \times \Phi(a), \quad (11)\end{aligned}$$

where operator $\times$ takes effect on related prototypes/actions when computing loss. Advice potentials *upscale or downscale the relevant loss components* as per cumulative feedback and adaptively control their amplitude, appropriately navigating the loss landscape towards **de-risked** models.

Impact of active knowledge-in-the-loop. Active KITL

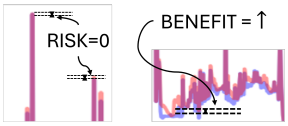


Figure 4. Zoomed portions in Fig 3 after active KITL refinement. We get de-risked (≈ 0) policy [left] w/ increased benefit [right]

softly guides the selection and learning of prototypes that represent diverse oversubscription policies (Figure 2) leveraging $\boxed{\kappa}$ (humans or generative AI). Intuitively, prototypes project decision space (actual ζ) into a lower dimensional manifold. So, feedback over prototype quality and diversity as well as actions gives a much richer knowledge to strongly **de-risk**, $\chi(\{\zeta_e\}) \downarrow$, policies (Fig 4). Intelligent query schedule elicits active feedback on prototypes or predicted actions that need more information/help (knowing-what-it-knows), thus minimizing budget $\sum_{t=1}^{T} |\mathcal{Q}_t|$ implicitly.

4 EXPERIMENTS

We empirically evaluate our method on virtual CPU oversubscription scenario in Cloud Sytstem cloud platform. Additionally, to highlight the generality of PROTORAIL we also evaluate on a novel airline ticket overbooking domain described later. We collect real data from these domains and propose respective simulators for evaluation that are described next.

4.1 Domain (Data)

4.1.1 *Primary: vCPU Oversub. in Internal Cloud.*

We evaluate with real data (de-identified) from cloud platform for internal users, *i.e.*, owners of Microsoft's services and applications. We collect two-week data of VM features, including the usage of vCPU, memory, and network, that belong to 30 randomly sampled services. As the vCPU usage has a lot of fine-grained variances, we take the peak usage in the one-hour bucket as the representative data point. Note that in Figure 1, VMs are allocated onto nodes, and VMs from different services can be collocated in the same node. Then, we propose a simplified allocation simulator that allocates VMs via Best-Fit allocation policy (Hadary et al.,

2020). VMs are allocated after vCPU oversubscription. For more details refer to Appendix B.2.

4.1.2 Other: Airline Overbooking

Even though vCPU oversubscription in cloud is the main focus, our method generalizes to any oversubscription problem in other domains. We demonstrate that with airline overbooking. We collect airline passengers' data from the overbooking reports of the U.S. Department of Transportation (DOT). The dataset covers overbooking information of 32 airline companies in the U.S. from 1998 to 2021 [2], reported quarterly and train GBDT emulator (further details in Appendix B.2).

4.2 Experiment Setting

State, Action, Reward In our vCPU oversubscription problem, the state s_t consists of the feature of the historical CPU usage rate, users' memory, CPU, and network requests q_t for each VM, Nodes' capacity and etc. At time step t, the action $a_t \in [0, 1]$ indicates the oversubscriptio rate which indicates we will allocate a VM with $a_t * q_t$ resources. The reward $r_t = -h_t + m_t$, where h_t indicates the number of hot nodes and m_t indicates the number of saved vCPUs or remain Core. In the flight tickets oversubscription problem, the state s_t indicates the historical sold tickets, the number of onboarding customers, the number of seats e_t in the airplane. The action $a_t \in [0, 1]$ indicates the oversubscription rate. The reward function $r_t = -c_t + o_t$, where c_t indicates the compensation cost and o_t indicates the profits. In summary, the state space is factored with a hybrid feature vector, including temporal features. The action/decision space is continuous, *i.e.*, the oversubscription level. This makes our problem complex to be solved in a straightforward behavior cloning way. Instead, PROTORAIL embeds into a latent space of equivalence prototypes and exploiting approximate symmetries in the trajectory patterns.

Configuration settings of KITL: Some of the settings of the KITL module as follows, `use_kitl` toggles the human-in-the-loop feedback system, which is set to `True` in experiments. `FREQUENCY` parameter in the feedback submodule of PROTORAIL allows for additional control on the frequency of queries to the human if needed, which defaults to 10 iterations. It does not necessarily mean that a query will always be generated at every 10 iterations. Queries are subject to the query generation framework (sec. 3.5.1). However, if this parameter is set to 10 and if ≥ 2 queries are generated between consecutive 10 iterations then extras will be skipped allowing stricter budgeting. `Uncertainty Threshold` Tr, (defaulted to 0.8) in experiments based on empirical observation.

[2] https://www.bts.gov/denied-confirmed-space

Other hyperparameters are inherited from the base prototypical IL module. KITL experiments on vCPU domain has been performed with a `batch_size = 128`, which also the default in base protoypical IL. In case of airplane ticket overbooking, we performed KITL experiments with `batch_size` = {32, 64, 128}, all of them giving us the same results that have been reported in the paper. Base IL uses are batch size of 1 due to the way the training loop has been designed for this domain, however in KITL batch size of 1 does not allow proper prototype uncertainty computations so we experimented with batch sizes > 1 and we made sure the performance is not affected. Another important aspect is the number of epochs/iterations are same as default values in based code IL (*e.g.* 300 in vCPU experiments). However, if a `split` or `merge` feedback is given and the prototype structure is altered, we introduce 30 extra training iterations after such feedback.

Choice of knowledge source κ. We primarily leverage human domain experts (cloud operators, developers, and provisioning specialists) as κ for active feedback in de-risking policy learning in vCPU oversubscription. The choice of human experts over LLMs is motivated by – (1) LLMs are computationally heavy to be used for active feedback (2) As shown later with additional studies that zero-shot (non fine-tuned) LLMs cannot provide better feedback than human experts. (3) Compliant LLMs that are needed to experiment on deployed internal cloud come with token budgets. Thus designing proper prompts that represent the current prototypes, predictions, queries to get policy or prototype level feedback is difficult since off-the-shelf LLMs are not good at symbolic/logical reasoning yet. We do show some studies with LLMs as the knowledge/feedback source.

4.3 Experimental Settings

We evaluate our approach on the vCPU oversubscription as well as the ticket overbooking domains, focusing on the interpretability and effectiveness from three aspects: (1) learned prototypes visualization: We visualize prototypes to analyze the policy learned by our method. (2) Task inference. We show how PROTORAIL learns to compose the prototypical options to solve both continuous and discrete control tasks. (3) The performance of PROTORAIL in these tasks. In particular, we outline two evaluation metrics for vCPU oversubscription task based on pracitcal behavior, *i.e.*, *hot node* and *remain core*. Specifically, a node hosts several VMs (Appendix A) and reserves 15% resources. If the node CPU utilization is higher than 85%, it is a *hot node* (Qiao et al., 2021) which diminishes the performance of hosted services (risk). *Remain core* as the other metric measures the benefits, defined as the remaining cores that could be potentially used by additional VMs after oversubscription (benefit). Adapting to the flight ticket overbooking, *compensation cost* refers to the monetary compensations for

Table 1. Results of different learning strategies on *vCPU oversubscription* and generalization to *flight tickets overbooking*.

Approach	*vCPU Oversubscription*		*Flight Tickets*	
	Hot Node/Risk↓	Core (Benefit)↑	Cost/Risk↓	Profit↑
Grid-search	0%	7450	0M	0M
Moving Average	1.39%	7628	0.96M	6.79M
DDPG	1.47%	5030	12.37M	2.35M
Behavior Cloning	1.19%	7870	1.47M	7.21M
GAIL	1.2%	6980	2.74M	4.56M
Dagger (*20 time steps*)	0.96%	7938	0.47M	6.95M
LSTM	1.27%	7749	1.82M	4.98M
Coop. Multi-Agent RL	0.89%	7897	0.59M	8.17M
PROTORAIL (*w/o KITL*)	**0%**	**8153**	0.31M	8.79M
PROTORAIL	**0%**	**8161**	0.14M	13.65M

offloaded passengers (risk), and *profit* refers to the extra revenues gained by overbooking (benefit).

Fewer hot nodes or lower compensation costs (χ) with more remain cores/ more profits (β) on vCPU and airline oversubscription resp., (shown in Table 1) highlight how PROTORAIL outperforms baselines. Our detailed ablation studies in section 4.5 on the learned prototypes and policies as well as the KITL module highlight how, (1) learned prototypes are interpretable representatives of an equivalence classes of trajectories and (2) efficient KITL refinement leads to *de-risked* policies.

Additional studies on (1) hyper-parameter tuning, such as evaluations against a discretized sweep over the *number of prototypes/options*, (2) pressure tests on reduced "hot-node ratio level/threshold as well as (3) more details on the datasets are presented in Appendix[3] A.1, A.2 & B.2.

4.4 Comparisons against SOTA Baselines

Baselines: We compared three baselines, (1) **Heuristic policy**: Grid-search, which searches for the overall oversubscription rate to improve the benefits by reducing the costs/risk. Moving average, which averages the historical usage rate as the current time step's oversubscription action. (2) **Imitation learning**: BC (Pomerleau, 1991), GAIL (Ho & Ermon, 2016) and human-guided IL such as Dagger (with 20 steps of human guidance). (3) **Reinforcement learning**: DDPG (Lillicrap et al., 2016) for continuous action space. We also compare against two recent, closely related frameworks for VM provisioning, namely, (4) **LSTMs:** ScroogeVM (Jacquet et al., 2024) which uses LSTM-driven supervised learning of usage patterns, and (5) **Coop. Multi-agent RL:** Cooperative MARL framework (Sheng et al., 2023) for optimizing oversub. strategies, treating each VM as an agent with cooperative updates.

Results: We run five seeds to conduct Wilcoxon signed-rank tests to check the statistical significance of the results. It shows that all the p-values of Wilcoxon signed-rank tests at 95% confidence level are smaller than 0.05. We further observe that: (1) DDPG (RL) fails in both scenarios as it is hard to optimize two competing objectives simultaneously in standard RL. (2) The heuristic policies, *i.e.*, Grid-search and Moving Average, show robust performance on vCPU oversubscription. But Grid-search fails on airline overbooking due to the high risk of that problem, *i.e.*, the ticket sales are highly dynamic, and a static oversubscription rate fails to adapt to different situations. (3) BC beats GAIL, possibly because BC has better sample efficiency and, in oversubscription scenarios, training samples are limited. Furthermore, the human-guided approach (Dagger) shows the second-best performance, demonstrating the benefits of human guidance in this domain. We set 20 steps of human guidance in Dagger, which is fair enough because it accounts for almost 1/6 of the trajectory length on average. Also, in our model, the number of queries to humans in KITL is much smaller than 20 (*e.g.*, 6 queries on average in the vCPU experiments and 8 queries on average in the Air Ticketing experiments). (4) Sequential methods such as LSTM models fail to do anything remarkable, and, in fact, the risks are quite high with no improvement in benefits. Multi-agent RL does try to minimize the risk as far as possible, but with no improvement in benefits. However, that is not even at par with PROTORAIL. (5) PROTORAIL outperforms all, both with and without KITL, as it captures different classes of patterns via learning a set of prototypes, and KITL refinement of such classes further improves benefit (Core/Profit) and mitigates risk (Hot node/Cost). Although grid-search, the most conservative policy, is at par in terms of risk mitigation, PROTORAIL achieves both the least risk and highest benefit in both domains.

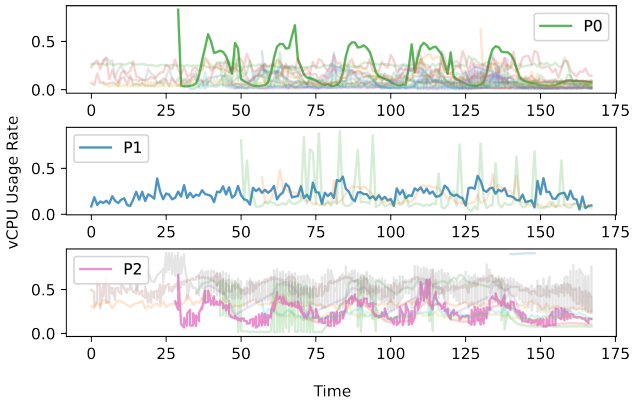
4.5 Additional Studies

4.5.1 Analysis of the Learned Prototypes:

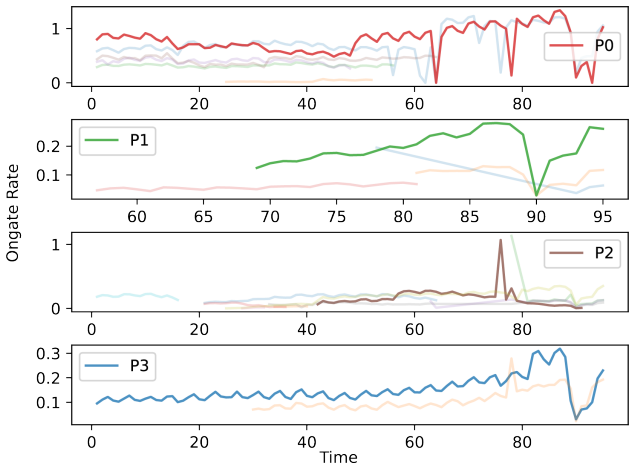
We learn 3 and 4 prototypes in vCPU oversubscription and airline overbooking, respectively. PROTORAIL learns these prototypes and represents them via finding the nearest trajectories. In Figure 5(a), different sub-figures show the trajectories in different prototype clusters. The prototype embedding is highlighted, and the trajectories are in transparent color. We analyze the interpretability of the policy via the learned prototypes: prototypes with hourly patterns are email/work-related services ($P0$), non-user-facing services that do not show temporal patterns ($P1$), and social media/gaming services ($P2$). This demonstrates the capability of PROTORAIL in learning interpretable prototypes.

As for airplane tickets (Figure 5(b)), PROTORAIL learns 4 prototypes with different trends. $P0$ indicates policies within a stable constant range. $P1$ indicates continuously increasing levels with occasional drops. $P2$ indicates policies that are sensitive to the environment and quickly increase

[3]All Appendices at https://aka.ms/protorail



(a) Prototypes in vCPUs oversubscription



(b) Prototypes in airplane tickets overbooking

Figure 5. Learned prototypes vizualized on top of equivalence classes of usage patterns they represent

action value. $P3$ indicates a smooth increase over time.

Limitation: Airline ticket data does not show a clear pattern due to demand drop in COVID-19 (Amankwah-Amoah, 2021) and change in the distribution over booking patterns.

4.5.2 *Analysis of active feedback:*

As Table 1 illustrates, KITL module further de-risks (and enhances benefits) policies with minimal queries to the human by, (1) Querying only at relevant stages and (2) Effective refinement. In vCPU domain, *we initialize with 6 prototypes, as per prior insights, which may do well during early stages of training.* However, once the learning plateaus we observe, (1) for prototype $P1$ cluster, the predicted oversubscription rate remains significantly higher than usage (2) in $P5$, predicted oversubscription is less than usage, *an overloading risk* (3) $P4$ has almost no services in the cluster, hence redundant. PROTORAIL automatically detects such nuances and poses targeted queries, at only certain stages, to actively get feedback, as shown below. Note that this is one particular example during one training run. A query can vary in "what" is being asked about and "when" during the training process is the query fired, based on how earlier queries were answered.

```
STEP 110; Query:  P=[5, 1]=[0.9904377
1.0770873] seem unstable.  Please
suggest:  (1=good, 0=none, -1=bad OR
"split ⟨prototype#⟩").
$ input:  1 = -1
STEP 240; Following prototype pairs seem
redundant [(4,1), (2,0), (2,5)].  Wish to
merge any (merge ⟨prototype#⟩)?
$ input:  merge 4 1
```

It automatically identifies $P5/P1$ as problematic and P4 as redundant (symmetric with P1); probes about possible merging. Feedback on predicted actions with overloading risk queried and obtained similarly. In our experiments, the total number of queries required to reach stable performance was ≤ 10 showing active feedback efficiency. The feedback is then used to jointly refine both the prototype memberships and the embedding function. Figure 5(a)/5(b) shows how # prototypes have reduced to 3 and 4 post-refinement. Additionally, the queries may be sensitive to hyperparameters such as thresholds($\mathbb{U}_p$ & $\mathbb{U}_a$). So we performed a grid search and observed that any threshold in a range of $(0.5, 0.75)$ led to a stable performance on average. Reported results are with 0.55 as thresholds.

4.5.3 *LLMs as knowledge source* κ

"Cost of domain experts" is common concern for human-knowledge guided learning and so, LLMs could prove to be good alternative knowledge source κ. However the cost is a holistic measure of the difference between β and monetary value of expert's time, which is impossible to track. A practical measure is query budget. Our analysis shows a minimal number of queries (6-8), significantly lower than other methods, suggesting that the expert's cost is offset by substantial benefits and risk reductions.

But we have also analyzed if LLMs are reasonable κ for policy and prototype refinement. Preliminary studies (table 2) indicate LLMs offer promising results in terms of clustering accuracy and response speed, yet require further fine-tuning and prompt engineering to fully leverage their potential. Note that clustering accuracy as a metric has a certain amount of ambiguity. There exists cluster quality measures Davies-Bouldin index that quantify intra and inter cluster densities together. However, such measures did not prove useful in our case because the notion of usage prototype quality needs to be domain-aware. Thus, we use expert evaluations to measure ac-

Table 2. Perf. Comparison between Human Experts and LLMs

Metric	Human	LLM
Accuracy in Clustering ↑	98%	87%
Answers per Second ↑	0.35	0.57
Prototype Restructuring ↑	30%	21%
Risk Mitigation ↑	100%	46%
Overhead Cost ↓	C_H	C_{LLM}

curacy here. We use an open source LLM, Llama-3.1-70B (Meta, 2024), in zero-shot fashion without any task-specific fine-tuning. The prompts were designed but combining usage telemetry clusters, prototypes or predictions and queries. where C_H (Human Cost) would be calculated as [Query budget $\times$ Expert time$_{/query}$ $\times$ \$$_{/unit\ time}$] and C_{LLM} (LLM Cost) as [Hosting \$ + ((Compute \$$_{/token}$ + API \$$_{/token}$) $\times$ tokens$_{/query}$) $\times$ Query budget].

These findings underscore the continued value of human input in risk-cognizant training, while also highlighting the potential of integrating LLMs as low-latency low-cost knowledge sources complementing domain experts. Also finetuned LLM on oversubscription domain could do a better job than querying zero-shot ones. Further experimentation with more advanced API based models like GPT4-o series or task-specific tuned versions of the models is an immediate future work.

5 Application in Practice

The vCPU oversubscription policy learned by PROTORAIL was successfully deployed in several scenarios (services/applications/features) within Microsoft's internal cloud services, but with limited blast radius. Our policy improves vCPU utilization by 9.4% and maintains a 0% hot node rate. The following case study demonstrate the effectiveness on real workloads.

5.1 Case Study

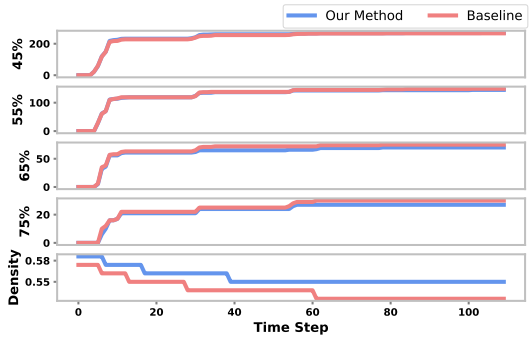


Figure 6. A/B test Results

We illustrate comparison between our policy and strongest baseline, behavior cloning on Microsoft's internal cloud. The observation was made from one of services over a two-week period. This service has been deployed in approximately 300 clusters. Since these are real workloads, although the usage rate telemetry exhibit patterns it has a lot of sample noise as well. Appendix B.1 discusses more details about usage patterns.

The results are shown in Figure 6. The y-axis shows the cumulative number of hot nodes during the test time at different hot thresholds (45%, 55%, 65%, and 75%). Density is the average node density, calculated as $\frac{usedCPU}{totalCPU}$, which corresponds to the remaining core capacity. *Lesser hot nodes and larger densities indicate a more effective oversubscription policy.* We observe that (1) our method results in fewer hot nodes and a larger node density. The risk-congizant policy helps to quickly adjust the policy when hot nodes are detected. (2) we see that a larger node density does not necessarily mean more hot nodes. With the right oversubscription rate and risk-cognizant mechanism, we can effectively coordinate VMs over the long term. This indicates significant potential for optimizing the oversubscription policy. In fact, there are no hot nodes when the threshold is set to 85%, and less than 10% of clusters have a usage rate above 75%.

Table 3 also abstracts our VM-level benefits and risk observations on our internal deployment, where "VMs w/ β" are VMs which show ore than 1K vCore savings, and "VMs at Risk" are VMs running on hot nodes, BRR is the Benefit to Risk Ratio. This result here is computed from two-week long observations on the clusters of only 2 regions and not all 300 clusters. In the approaches "Manual" indicates manually decided oversubscription, "Heuristics" indicate statistical non-adaptive prediction model based on collected heuristics. We see that PROTORAIL has the best BRR.

Table 3. Observations from deployment on internal cloud

Method	MSE ↓	VMs w/ β ↑	VMs at Risk ↓	BRR ↑
Manual	–NA–	109	14	7.79
Heuristics	0.065	3502	185	18.95
PROTORAIL	**0.042**	**3542**	113	**31.34**

6 Related Work

Guided imitation learning. Learning decision-making policy can be challenging in the presence of (1) "systematic noise" which include sample/feedback sparsity/noise, delayed signals (Rauber et al., 2021), cognitive bias, sub-optimal trajectories, etc.),

Pure data-driven learning can be risky. Even in offline training, such as IL, inverse RL, or offline RL, trajectories could be noisy and catastrophic failure can occur in out-of-distribution cases. Knwoledge-guided IL is an active area of research. Some approaches exploit prior knowl-

edge (Syed & Schapire, 2007; Kunapuli et al., 2013), some design constraints based on domain knowledge (Eppner et al., 2009; Perico et al., 2019; Kim & Park, 2018) or use reward-shaping functions (Judah et al., 2014) and some use statistical models as priors (Englert et al., 2013). Brantley *et al.* (Brantley et al., 2020) propose an *active learning form of IL* with imperfect human guidance and is the closest in spirit to our formulation. Guided Behavior Cloning, DAGGER, and HgDAGGER (Sasaki & Yamashina, 2021; Ross & Bagnell, 2010; Kelly et al., 2019) are some popular interactive human-in-the-loop IL frameworks; however, their feedback elicitation is naive, inefficient and not suitable for rich multi-level feedback.

Interpretable Modeling. Interpretable modeling mainly falls into two ways (Molnar, 2020): (1) **intrinsic explanation** which transparentizes the model by restricting the complexity, *e.g.*, decision tree or case-based model (Li et al., 2018; Ming et al., 2019; Kim et al., 2016), (2) **post-hoc explanation** which is achieved by analyzing the model after training

(Mott et al., 2019), distilling a black-box policy into a simple structure (Verma et al., 2018). A set of post-hoc imitation learning approaches were proposed for generating meaningful policy. However, the intrinsic explanation model is sometimes desirable since post-hoc explanations usually do not fit the original model precisely (Rudin, 2019). Prototype learning (Newell et al., 1972; Cohen et al., 1996; Kolodner, 1992) which draws conclusions for new inputs by comparing them with a few exemplar prototype belongs to the intrinsic explanation method.

7 CONCLUSION

We presented our novel prototypical imitation learning framework with active knowledge-guided refinement (leveraging human domain experts) that addresses a critical decision-making problem of *adaptive* oversubscription with competing objectives of efficient capacity utilization with minimal risk. We show via extensive and ablation studies on real Microsoft's internal could service how learned prototypes discover implicitly interpretable classes approximately symmetric demand patterns. This results in effective oversubscription ratio at *any granularity*, i.e. VMs, services or any other grouping. Efficient active KITL refinement of prototypes and actions *substantially de-risks policies with enhanced COGS benefit*. Through extensive empirical evaluation and case studies we have exhibited how our framework is extremely effective on cloud platform and can be generalized seamlessly to other oversubscription domains as well. Our studies also show how LLMs are promising in complement to domain experts but needs more investigation.

In future, we plan to extend PROTORAIL to other provisioning problems with similar properties. We also plan to infuse knowledge about characteristics of workloads (application level metrics) running or hosted on these VMs for even better decision making. Finally a holistic KITL framework that can leverage both LLMs and domain experts together organically is of extreme interest to us.

REFERENCES

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. PMLR, 2017.

Amankwah-Amoah, J. Covid-19 pandemic and innovation activities in the global airline industry: A review. *Environment International*, 156:106719, 2021.

Baset, S. A., Wang, L., and Tang, C. Towards an understanding of oversubscription in cloud. In *2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE 12)*, 2012.

Brantley, K., Sharaf, A., and Daumé III, H. Active imitation learning with noisy guidance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2093–2105, 2020.

Breitgand, D. and Epstein, A. Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds. In *2012 Proceedings IEEE INFOCOM*, pp. 2861–2865. IEEE, 2012.

Brown, D. S., Cui, Y., and Niekum, S. Risk-aware active inverse reinforcement learning. In *Proceedings of The 2nd Conference on Robot Learning*, 2018.

Cao, M., Shu, L., Yu, L., Zhu, Y., Wichers, N., Liu, Y., and Meng, L. Drlc: Reinforcement learning with dense rewards from llm critic. *arXiv preprint arXiv:2401.07382*, 2024.

Cohen, M. S., Freeman, J. T., and Wolf, S. Metarecognition in time-stressed decision making: Recognizing, critiquing, and correcting. *Human Factors*, 38(2):206–219, 1996.

Cohn, D., Atlas, L., and Ladner, R. Improving generalization with active learning. *Machine learning*, 15:201–221, 1994.

Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., and Bianchini, R. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *SOSP*. Association for Computing Machinery, 2017. ISBN 9781450350853. doi: 10.1145/3132747.3132772.

Das, M., Dhami, D. S., Yu, Y., Kunapuli, G., and Natarajan, S. Human-guided learning of column networks: Knowledge injection for relational deep learning. In *COMAD/CODS*, 2021.

De Haan, P., Jayaraman, D., and Levine, S. Causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Deng, C., Ji, X., Rainey, C., Zhang, J., and Lu, W. Integrating machine learning with human knowledge. *iScience*, 23(11):101656, 2020.

Du, Y., Watkins, O., Wang, Z., Colas, C., Darrell, T., Abbeel, P., Gupta, A., and Andreas, J. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pp. 8657–8677. PMLR, 2023.

Englert, P., Paraschos, A., Deisenroth, M. P., and Peters, J. Probabilistic model-based imitation learning. *Adaptive Behavior*, 2013.

Eppner, C., Sturm, J., Bennewitz, M., Stachniss, C., and Burgard, W. Imitation learning with generalized task descriptions. In *2009 IEEE International Conference on Robotics and Automation*, pp. 3968–3974. IEEE, 2009.

Franklin, M., Graybeal, P., and Cooper, D. *Principles of accounting. Volume 1: Financial accounting*. OpenStax, 2019.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Goldberger, J. and Ben-Reuven, E. Training deep neural-networks using a noise adaptation layer. In *ICLR*, 2017.

Hadary, O., Marshall, L., Menache, I., Pan, A., Greeff, E. E., Dion, D., Dorminey, S., Joshi, S., Chen, Y., Russinovich, M., et al. Protean:VM allocation service at scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pp. 845–861, 2020.

Harel, S. and Radinsky, K. Accelerating prototype-based drug discovery using conditional diversity networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 331–339, 2018.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *NeurIPS*, volume 29, 2016.

Householder, R., Arnold, S., and Green, R. On cloud-based oversubscription. *arXiv preprint arXiv:1402.4758*, 2014.

Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Jacquet, P., Ledoux, T., and Rouvoy, R. Scroogevm: Boosting cloud resource utilization with dynamic oversubscription. *IEEE Transactions on Sustainable Computing*, 2024.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018.

Judah, K., Fern, A., Tadepalli, P., and Goetschalckx, R. Imitation learning with demonstrations and shaping rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

Kelly, M., Sidrane, C., Driggs-Campbell, K., and Kochenderfer, M. J. Hg-dagger: Interactive imitation learning with human experts. In *ICRA*, 2019.

Kim, B., Khanna, R., and Koyejo, O. O. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in neural information processing systems*, pp. 2280–2288, 2016.

Kim, K.-E. and Park, H. S. Imitation learning via kernel mean embedding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Kolodner, J. L. An introduction to case-based reasoning. *Artificial intelligence review*, 6(1):3–34, 1992.

Kunapuli, G., Odom, P., Shavlik, J. W., and Natarajan, S. Guiding autonomous agents to better behaviors through human advice. In *ICDM*, 2013.

Lee, D.-H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, 2013.

Li, O., Liu, H., Chen, C., and Rudin, C. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Li, Z. An adaptive overload threshold selection process using markov decision processes of virtual machine in cloud data center. *Cluster Computing*, 22(2):3821–3833, 2019.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

Ma, R., Luijkx, J., Ajanovic, Z., and Kober, J. Explorllm: Guiding exploration in reinforcement learning with large language models. *arXiv preprint arXiv:2403.09583*, 2024.

Mahapatra, N. R. and Venkatrao, B. The processor-memory bottleneck: problems and solutions. *Crossroads*, 5(3es): 2, 1999.

Mazyavkina, N., Sviridov, S., Ivanov, S., and Burnaev, E. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134: 105400, 2021.

Meta. `https://huggingface.co/meta-llama/Llama-3.1-70B`, 2024.

Ming, Y., Xu, P., Qu, H., and Ren, L. Interpretable and steerable sequence learning via prototypes. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 903–913, 2019.

Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., and Gil, Y. Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40 (1-3):63–118, 1989.

Miyato, T., Maeda, S.-i., Ishii, S., and Koyama, M. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *TPAMI*, 2018.

Molnar, C. *Interpretable machine learning*. Lulu. com, 2020.

Mott, A., Zoran, D., Chrzanowski, M., Wierstra, D., and Rezende, D. J. Towards interpretable reinforcement learning using attention augmented agents. In *Advances in Neural Information Processing Systems*, pp. 12329–12338, 2019.

Neider, D., Gaglione, J.-R., Gavran, I., Topcu, U., Wu, B., and Xu, Z. Advice-guided reinforcement learning in a non-markovian environment. In *AAAI*, 2021.

Newell, A., Simon, H. A., et al. *Human problem solving*, volume 104. Prentice-Hall Englewood Cliffs, NJ, 1972.

Odom, P. and Natarajan, S. Active advice seeking for inverse reinforcement learning. In *AAAI*, 2015.

Paternain, S., Chamon, L., Calvo-Fullana, M., and Ribeiro, A. Constrained reinforcement learning has zero duality gap. *Advances in Neural Information Processing Systems*, 32, 2019.

Patrini, G., Rozza, A., Menon, A. K., Nock, R., and Qu, L. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, 2017.

Perico, C. A. V., De Schutter, J., and Aertbeliën, E. Combining imitation learning with constraint-based task specification and control. *IEEE Robotics and Automation Letters*, 4(2):1892–1899, 2019.

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 1991.

Qiao, B., Yang, F., Luo, C., Wang, Y., Li, J., Lin, Q., Zhang, H., Datta, M., Zhou, A., Moscibroda, T., et al. Intelligent container reallocation at microsoft 365. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1438–1443, 2021.

Rauber, P., Ummadisingu, A., Mutz, F., and Schmidhuber, J. Reinforcement learning in sparse-reward environments with hindsight policy gradients. *Neural Computation*, 33: 1498–1553, 2021.

Rosch, E. H. Natural categories. *Cognitive psychology*, 4 (3):328–350, 1973.

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In *AISTATS*, 2010.

Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215, 05 2019. doi: 10.1038/s42256-019-0048-x.

Sasaki, F. and Yamashina, R. Behavioral cloning from noisy demonstrations. In *International Conference on Learning Representations*, 2021.

Sheng, J., Wang, L., Yang, F., Qiao, B., Dong, H., Wang, X., Jin, B., Wang, J., Qin, S., Rajmohan, S., et al. Learning cooperative oversubscription for cloud by chance-constrained multi-agent reinforcement learning. In *Proceedings of the ACM Web Conference 2023*, pp. 2927–2936, 2023.

Syed, U. and Schapire, R. E. Imitation learning with a value-based prior. In *UAI*, 2007.

Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri, S. Programmatically interpretable reinforcement learning. *arXiv preprint arXiv:1804.02477*, 2018.

Wang, H. and Tianfield, H. Energy-aware dynamic virtual machine consolidation for cloud datacenters. *IEEE Access*, 6:15259–15273, 2018.

Wang, L., Das, M., Yang, F., Du, C., Qiao, B., Dong, H., Bansal, C., Qin, S., Rajmohan, S., Lin, Q., Zhang, D., and Zhang, Q. Coin: Chance-constrained imitation learning for safe and adaptive resource oversubscription under uncertainty. In *Proceedings of the 33rd ACM International*

Conference on Information and Knowledge Management, CIKM '24, 2024.

Williams, D., Jamjoom, H., Liu, Y.-H., and Weatherspoon, H. Overdriver: Handling memory overload in an oversubscribed cloud. *ACM SIGPLAN Notices*, 46(7):205–216, 2011.

Wittman, M. D. Are low-cost carrier passengers less likely to complain about service quality? *Journal of Air Transport Management*, 35:64–71, 2014.