VarShare Update

Niklas Nertinger

January 16, 2026

Contents

1 Motivation: The Stability-Plasticity Dilemma

1.1 Problem Statement

The primary challenge in Multi-Task Reinforcement Learning (MTRL) is the *Stability-Plasticity Dilemma*: agents must be plastic enough to learn task-specific nuances but stable enough to maintain shared dynamics across environments. Traditional approaches typically fall into two extremes:

- **Hard Parameter Sharing:** A shared backbone with task-specific heads. This often suffers from *Negative Transfer* (Gradient Interference) where conflicting gradients from one task distort the shared representation of another.
- **Independent Learning:** Separate networks for each task. This avoids interference but fails to leverage shared structure, leading to poor sample efficiency and redundant parameterization.

Recent high-capacity architectures (like BRC) attempt to mitigate this by scaling model size, but typically rely on unconstrained task embeddings. We argue that this lack of constraint leads to inefficient transfer and poor zero-shot initialization, as the network treats even identical tasks as distinct problems.

1.2 The VarShare Solution

To address this, we introduce **VarShare**, a framework that formalizes the trade-off between sharing and specializing using **Variational Inference (VI)**. We model task-specific parameters not as deterministic values, but as variational residual adapters in weight space.

1.2.1 Probabilistic Formulation

Let θ be the shared base parameters and ϕ_m be the task-specific residual for task m. The effective weight matrix is defined as $W_m = \theta + \phi_m$. In VarShare, ϕ_m is governed by a hierarchical structure:

1. **The Prior (Hypothesis of Similarity):** A fixed centered Gaussian prior $p(\phi_m) = \mathcal{N}(0, \sigma^2_{prior} I)$ that assumes tasks are identical to the base by default.
2. **The Posterior (Learned Adaptation):** An approximated variational distribution $q_m(\phi_m) = \mathcal{N}(\mu_m, \sigma^2_m)$, where μ_m and σ_m are learnable parameters.

We maximize the Evidence Lower Bound (ELBO):

$$\mathcal{L} = \underbrace{E_{\phi_m \sim q_m}[\log p(\mathcal{D}_m | W_m)]}_{\text{RL Performance}} - \beta \underbrace{D_{KL}(q_m || p)}_{\text{Complexity Cost}} \tag{1}$$

1.2.2 Mechanism: Robustness and Exploration

The learnable variance σ_m introduces two distinct advantages over deterministic sharing:

- **Robustness (Flat Minima):** By optimizing expected loss over the distribution q_m, the system penalizes brittle solutions. High σ_m forces the backbone to find broad, stable basins of attraction ("flat minima") rather than sharp peaks. This ensures the shared parameters θ remain valid even when perturbed by task-specific noise.
- **Exploration:** The stochastic sampling ($W = \theta + \mu + \sigma\epsilon$) acts as parameter-space noise. This drives exploration automatically when the agent is uncertain, preventing the optimizer from getting stuck in poor local optima (similar to Thompson Sampling).

1.2.3 Cost Asymmetry & Signal Amplification

A critical property of VarShare is its preference for sharing. The shared base θ receives gradients summed across all M tasks ($\sum \nabla L_m$) and incurs zero KL cost. In contrast, task-specific parameters μ_m receive only local gradients ∇L_m and face a direct KL penalty. As a result, the optimizer naturally moves θ to solve the common structure first ("The Crowd Effect"), using μ_m only for irreducible task-specific deviations.

2 Initial Experimental Results

Demonstrating VarShare's baseline competency against standard approaches.

2.1 Algorithm Baselines

2.1.1 Base Algorithm: PPO

For all experiments, we utilize **Proximal Policy Optimization (PPO)** (Schulman et al., "Proximal Policy Optimization Algorithms", 2017) as the underlying Reinforcement Learning algorithm. PPO is a policy gradient method that optimizes a surrogate objective function using stochastic gradient ascent. We employ the standard "Clipping" mechanism to ensure stability, preventing the new policy from diverging too far from the old one. We chose PPO for its robustness to hyperparameter settings and its standard usage in continuous control benchmarks.

2.1.2 Baselines Compared

We evaluate VarShare against a comprehensive suite of 6 Multi-Task RL baselines, covering architectural, regularization, and gradient-projection approaches:

1. **Independent (Plasticity Bound):** We train M separate PPO agents, one for each task. This approach eliminates gradient interference (negative transfer) entirely but fails to leverage any shared structure. It serves as the upper bound for parameter count and plasticity.

2. **Hard Sharing (Stability Bound):** A single network where all hidden layers (backbone) are shared across tasks, with only the final output layer (Head) being task-specific. This is the standard "Multi-Head" architecture. It minimizes parameters but is maximally exposed to conflicting gradients.

3. **Contextual Policy:** A single fully shared network (including the head) that receives a learned *Task Embedding* vector z_m concatenated to the state input. Unlike Hard Sharing, the head is shared, forcing the network to condition its entire policy on the embedding. (Yu et al., "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning", 2019).

4. **L2-Soft Sharing (L2Soft):** Similar to VarShare, this method defines task weights as $W_m = \theta + \phi_m$. However, instead of a variational objective, it applies a fixed L2 penalty $\lambda||\phi_m||^2$ to the task-specific parameters. **Difference from VarShare:**

   - **Point Estimate vs. Distribution:** L2Soft finds a single deterministic point μ_m. VarShare optimizes a full distribution $\mathcal{N}(\mu_m, \sigma_m^2)$. This allows VarShare to maximize entropy (via the $-\log\sigma$ term in KL) where the task is insensitive to precise weight values, effectively finding "flat minima" which generalize better.
   - **Loss Smoothing (Exploration):** L2Soft optimizes on the sharp, potentially non-convex loss landscape. VarShare optimizes the *expected* loss over the noise distribution $E_\epsilon[\mathcal{L}(\theta + \mu + \sigma\epsilon)]$. This effectively smooths the loss landscape (Gaussian Convolution), allowing the optimizer to escape sharp local optima that trap L2Soft.

5. **PCGrad (Projected Conflict Gradients)** (Yu et al., "Gradient Surgery for Multi-Task Learning", 2020): A gradient projection method applied to the Hard Sharing architecture. If the gradients of two tasks conflict (negative cosine similarity), PCGrad projects one gradient onto the normal plane of the other, effectively removing the conflicting component.

6. **CAGrad (Conflict-Averse Gradient Descent)** (Liu et al., "Conflict-Averse Gradient Descent for Multi-Task Learning", 2021): An advanced optimization method that seeks an update direction maximizing the average return while ensuring that performance on no individual task degrades beyond a specific constraint. It optimizes for the best "common direction" within a localized region.

2.2 Experimental Setup: Constructing Multi-Task Environments

To evaluate the agents' ability to adapt to varying dynamics, we modified standard OpenAI Gym environments by randomizing their physical parameters for each task. This forces the agent to learn a shared representation of the task structure while adapting to specific physical constants.

- **High-Variance CartPole (10 Tasks):** We varied the **pole length**, **cart mass**, **pole mass**, and **gravity**. Variations included heavy carts with short poles (fast, aggressive dynamics) versus light carts with long poles in low gravity (slow, unstable but floaty dynamics).
- **High-Variance LunarLander (20 Tasks):** We introduced significant **Wind** and **Turbulence** as the primary variations, alongside **Gravity** changes. Tasks ranged from calm, low-gravity environments (Moon-like) to high-gravity storms with strong lateral winds (Jupiter-like), creating conflicting control requirements.

2.3 Performance Summary

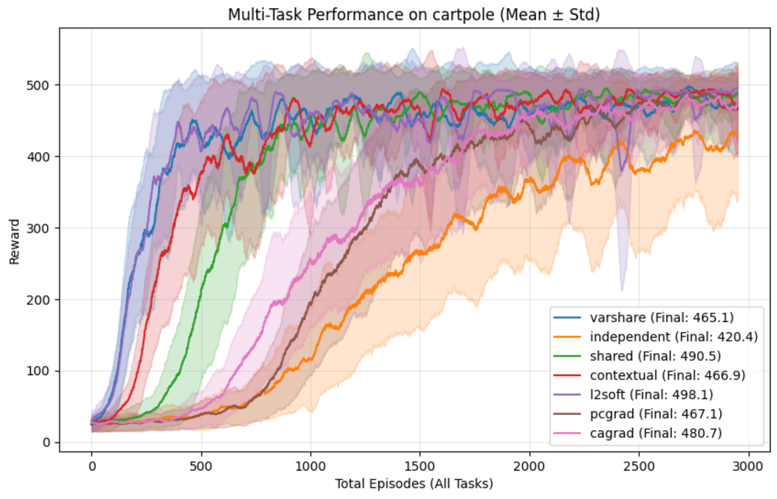

Figure 1: Multi-Task Performance on CartPole (Mean ± Std).

Key Findings:

- **Superiority over Baselines:** As illustrated in Figures 1 and 2, both VarShare and L2Soft significantly outperform the baselines.

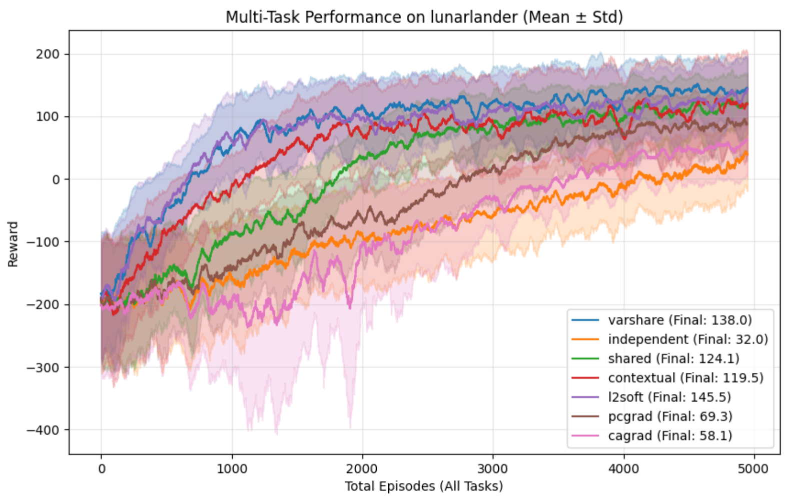

Figure 2: Multi-Task Performance on LunarLander (Mean ± Std).

- **VarShare ≈ L2Soft:** Notably, VarShare and L2Soft perform very similarly across both domains in this simplified setting. A detailed ablation study to disentangle the benefits of the adaptive variance σ_m versus the fixed penalty λ is planned for future work.

2.4 Ablation Study: Sampling Strategy

To validate the Bayesian premise of our architecture, we compared stochastic vs. deterministic inference.

- **Hypothesis:** Neural networks are non-linear functions (due to ReLUs, etc.). Therefore, $E[f(W)] \neq f(E[W])$. Using the mean weights μ at test time might be sub-optimal compared to sampling from the trained distribution, as the network was optimized to be robust to the noise σ.

- **Empirical Findings:** However, as shown in Figure 3, our studies found **no significant difference** between Mean and Sample evaluation strategies across varying task parameters. The performance curves overlap almost entirely, suggesting that the learned posterior variance σ_m is either tight enough that the non-linearity effect is negligible, or that the solution lies in a locally linear region.

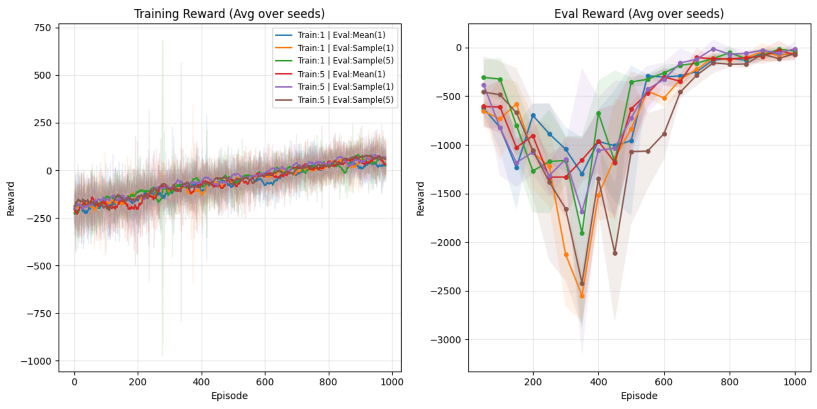

Figure 3: Training and Evaluation Reward comparison for Mean vs. Sample strategies (1 or 5 samples). No significant divergence is observed.

3 Improvement Portfolio Selection

We evaluated 10 algorithm variants to identify improvements that offer distinct functional advantages.

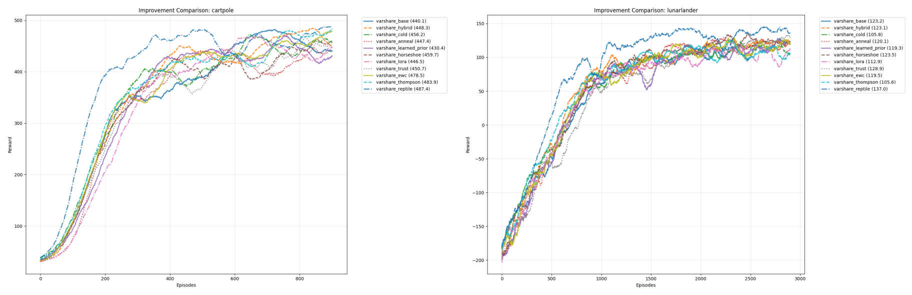

Figure 4: **Comprehensive Screening:** Performance curves for all 10 evaluated variants on CartPole (Left) and LunarLander (Right).

From this extensive screening, we have selected two core improvements that address the primary limitations of the base algorithm:

3.1 Performance: VarShare-Reptile

- **The Initialization Problem:** In standard multi-task optimization, the shared parameter θ is optimized to minimize the *sum* of task losses $\sum \mathcal{L}_m$. When task optima are located in distant, non-convex regions of the landscape, this compromise solution can land θ in a high-loss valley (a ”saddle point” between tasks) that is a poor starting point for any individual task.

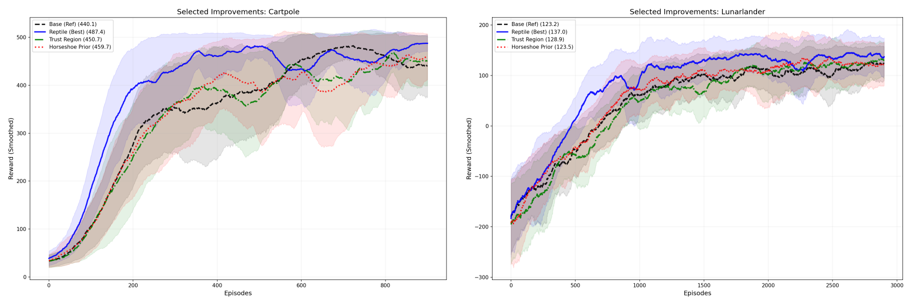

Figure 5: **Top Candidates:** A focused comparison of the some promising baselines (Reptile, Trust Region, Horseshoe) against the base model.

- **The Reptile Solution (Re-Centering):** VarShare-Reptile modifies the update rule to treat the adapted weights $(\theta + \mu_m)$ as a target for θ. It essentially "pulls" the shared backbone towards the task-specific solution after every update cycle, ensuring θ remains effectively centrally located in the parameter manifold.

- **Algorithm Implementation:**

```
for each iteration:
    for task m in batch:
        # 1. Standard PPO Update (Gradient Step)
        # Updates both theta and mu_m using data D_m
        # to maximize ELBO via Adam
        J_ppo = PPO_Surrogate(theta + mu_m, D_m) - beta * KL(mu_m)
        theta', mu_m' = Optimizer.step(J_ppo)

        # 2. Reptile Meta-Update (Re-Centering)
        # Shift theta towards the new effective weight (theta' + mu_m')
        # and shrink mu_m' correspondingly to preserve the total W
        shift = alpha * mu_m'
        theta_new = theta' + shift
        mu_new = mu_m' - shift
```

- **Key Details:**
  - **Gradient Step:** Refers to a full PPO update phase (multiple epochs over the collected minibatch).
  - **Data Usage:** The inner loop uses standard on-policy rollouts. The meta-update is "free" as it operates purely on the weight algebraic decomposition, requiring no additional data.

  - **Variational Consistency:** We sample $W \sim \mathcal{N}(\theta + \mu, \sigma)$ during the PPO update. The Reptile step is a deterministic re-parameterization that shifts the mean structure without altering the realized function or variance.

- **Result:** As evident in Figure 5, this approach yields the best performance.

3.2 Scalability: VarShare-LoRA

- **The Parameter Constraint:** Standard VarShare learns a full-rank residual matrix $\phi_m \in R^{d_{out} \times d_{in}}$. This scales quadratically ($O(d^2)$). For large backbones ($d = 4096$), distinct matrices for hundreds of tasks are prohibitive.
- **The LoRA Solution:** We decompose the variational residual into two low-rank matrices: $B_m \in R^{d_{out} \times r}$ and $A_m \in R^{r \times d_{in}}$, where $r \ll d$. The effective residual is $\phi_m = B_m A_m$.
- **Variational Formulation:** We define independent variational posteriors for both adapter matrices:

  $$q(B_m) = \mathcal{N}(\mu_B, \sigma_B^2), \quad q(A_m) = \mathcal{N}(\mu_A, \sigma_A^2)$$

  During the forward pass, we sample $A \sim q(A)$ and $B \sim q(B)$ separately to construct the noisy weight $W = \theta + BA$. The KL divergence is computed as the sum of the KLs for all elements in A and B.
- **Efficiency Gain:** reduces task-specific parameters significantly with negligible performance loss (see Figure 4, pink curve).

4 Base Algorithm Analysis

Investigating the independence of the VarShare mechanism.

4.1 PPO vs. SAC

- **PPO (Robust Baseline):** Our primary experiments utilize PPO due to its stability. It performs reliably across all tasks but requires more samples.
- **SAC (Sample Efficiency):** Soft Actor-Critic offers significantly higher sample efficiency. However, long compute times have limited the scale of research on this so far. Some experiments are running at this time.

HPO Methodology: To rigorously compare these baselines, we conducted Hyperparameter Optimization (HPO) campaigns. We employed Bayesian Optimization (BO) for approximately 50 trials per algorithm. To manage the computational cost, we utilized an early-pruning strategy and short overall trial length (episodes per trial). Especially in RL such shortened trials in HPO might not be ideal.

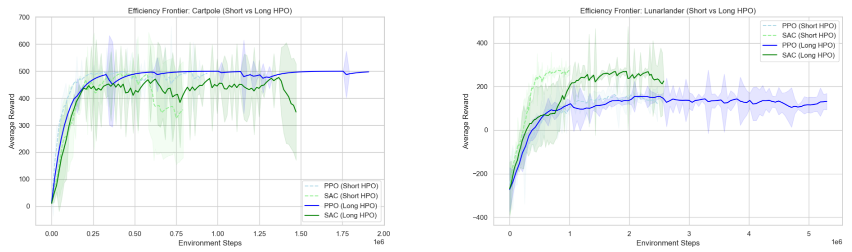

Figure 6: Comparison of PPO and SAC performance on CartPole (Left) and LunarLander (Right). The "Long HPO" curves represent the best configurations found after 50 trials of Bayesian Optimization.

4.2 Future Directions: Inference-Based RL (MPO)

- **Theoretical Alignment:** VarShare is built on Variational Inference. Current RL algorithms (PPO/SAC) treat the policy update as a separate optimization step. **MPO (Maximum a Posteriori Policy Optimization)** and **V-MPO** formulate RL *itself* as an inference problem (EM algorithm).
- **Prediction:** Combining VarShare with MPO would create a unified probabilistic framework where both the *policy update* and the *multi-task sharing* are solved via a single coherent ELBO maximization, potentially unlocking superior theoretical properties and performance.

5 Todos / Future Work

5.1 Leave-One-Out (LOO) Generalization

The goal of Multi-Task RL is not just performance on trained tasks, but also the ability to adapt to unseen scenarios. We plan to execute a **Leave-One-Out (LOO)** cross-validation study.

For a domain with M tasks, we train the agent on $M-1$ tasks and hold out task k. We then measure the sample efficiency required to solve task k under three regimes:

1. **Scratch (Baseline):** Initializing a fresh agent with random weights. This establishes the "naive" learning curve.
2. **Full Fine-Tuning:** initializing with the pre-trained shared backbone θ and allowing all parameters to update.
3. **Residual Adaptation (The "Acid Test"):** freezing the shared backbone θ and training *only* the task-specific variational adapter (μ_k, σ_k).

If VarShare has successfully disentangled the ”Universal Dynamics” (e.g., gravity, wind physics) into θ from the ”Task Specifics” (e.g., target location), then **Residual Adaptation** should converge significantly faster than Scratch. This would prove that θ has become a re-usable ”Neural Physics Engine”.

5.2 Other TODOs

- **Adaptive σ_{prior}:** Experiment with learning or dynamically annealing the prior variance, potentially per weight (similar to VCL).
- **In-Depth Ablation (L2Soft vs. VarShare):** Conduct granular analysis to isolate the benefits of variational noise vs. static L2 regularization.
- **Meta-World Scaling:** Scale all benchmarks and competitor comparisons to the full MT10 suite.
- **SAC Portfolio:** Validate the key improvements (Reptile, LoRA) within the Soft Actor-Critic framework.
- **SAC Benchmarks:** Implement competitor algorithms with SAC and run benchmarks.
- **Base Algorithm:** Explore other base algorithms like MPO.