ReBaPL: Repulsive Bayesian Prompt Learning

Supplementary Material

A. Additional Details and Experiments

Implementation Details. We utilize the ViT-B/16 variant of CLIP [34] as our vision-language backbone. All experiments are executed on NVIDIA L4 and L40S GPUs. We report average accuracy on base classes, novel classes, and their harmonic mean, computed over three independent runs with different random seeds, to ensure statistical reliability, following the experimental protocol for prompt learning methods [21, 46]. Since the MMRL method uses the AdamW optimizer, for our MMRL + ReBaPL method we perform a AdamW burn-in during the initial iterations of each cycle to reach the neighborhood of the local mode, and then switch to SGHMC in the final iterations of the cycle.

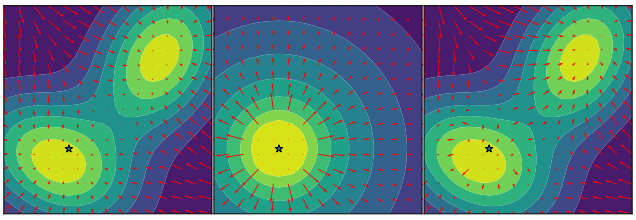
A.1. Toy Example

We start with a toy example to illustrate our method. We consider a simple Gaussian mixture potential $U(\theta) = \frac{1}{2}(\mathcal{N}(\cdot|\mu_1, \Sigma_1) + \mathcal{N}(\cdot|\mu_2, \Sigma_2))$ and $V(\theta, \theta')$ as in Equation 17, with $d(\theta, \theta') = \|\theta - \theta'\|_2$ for simplicity. The underlying potential U has two modes (μ_1 and μ_2), and we want to explore both of them through MCMC (SGLD in this case, for simplicity).

We start by running standard SGLD to obtain a sample in the posterior, denoted by the blue star in Figures 2 and 3. As shown in Figure 2 (a), this point is located in a high density region of $U(\theta)$. For the next cycle, we add repulsion for exploring the second mode in the landscape of $U(\theta)$. The repulsion potential, V, is shown in Figure 2 (b). Note how the vector field, i.e., $F(\theta, \theta_{k,T}^{(c-1)})$, point outwards from $\theta_{k,T}^{(c-1)}$. We show the posterior alongside the final vector field $\nabla_\theta \left(U(\theta) + \xi V(\theta, \theta_{k,T}^{(c-1)}) \right)$ in Figure 2 (c). Overall, repulsion modifies the vector field, driving samples away from $\theta_{k,T}^{(c-1)}$, thus enhancing mode exploration.

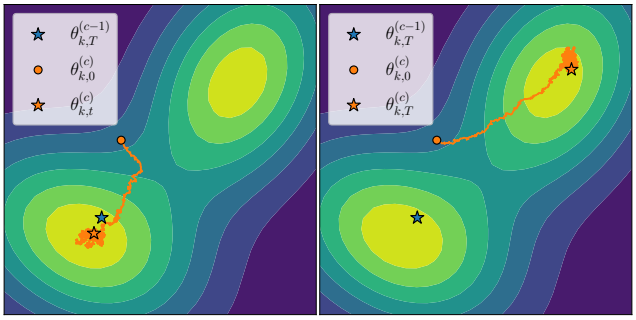
We then run SGLD and SGLD with repulsion on this toy example. The results are shown in Figure 3 (a) and (b) from an initialization $\theta_{k,0}^{(c)}$. Due the initialization, SGLD converges to the same mode as $\theta_{k,T}^{(c-1)}$. In comparison, by adding the repulsion term, $\theta_{k,0}^{(c)}$ converges to the second mode in the posterior.

Remark. *(Intuition on repulsion strength ξ) The repulsion strength ξ strikes a balance between the force driving samples to the modes of the log-likelihood, i.e., $\nabla_\theta U(\theta)$, and the repulsive force driving samples away from previous cycles' sample, i.e., $\nabla_\theta V(\theta, \theta_{k,T}^{(c-1)})$. Intuitively, if $\xi \to +\infty$, the repulsive strength overpowers the force towards the modes of the posterior. Therefore, tuning ξ is important to strike a balance between standard SGMCMC and mode exploration. We show in Figure 4 an illustration of this phenomenon.*



(a) Potential $U(\theta)$ (b) Potential $V(\theta, \theta')$ (c) Field w/ repulsion

Figure 2. Conceptual illustration of the benefits of repulsive MCMC on a mixture of Gaussian distributions. In (a), we show the potential $U(\theta)$ alongside the vector field $\nabla U(\theta)$. The blue point, $\theta_{k,T}^{(c-1)}$ represents the minimum of $\theta \mapsto U(\theta)$ obtained through SGLD. In (b), we show the repulsion potential $V(\theta, \theta_{k,T}^{(c-1)})$ alongside the vector field $F(\theta, \theta_{k,T}^{(c-1)}) = \nabla_\theta V(\theta, \theta_{k,T}^{(c-1)})$. As we show in (c) the repulsion vector field changes the initial vector field, pushing samples away from the sample of the previous cycle.



(a) SGLD (b) SGLD + Repulsion

Figure 3. As we show in (a) and (b), we encourage mode exploration by driving samples from the current cycle, $\theta_{k,t}^{(c)}$, away from those of the previous cycle, $\theta_{k,T}^{(c-1)}$.

A.2. Prompt Diversity

An important aspect of MCMC is yielding diverse samples from the posterior distribution. This can be analyzed through the lens of mode collapse, i.e., whether the samples $\{\theta_{k,T}^{(c)}\}_{k=1}^{K}$ are close with respect to some defined metric.

To assess whether our method successfully mitigates mode collapse, we analyzed the similarity in feature distributions between the sampled models. We computed the Wasserstein distance between the representations for three independent checkpoints. This metric quantifies how dis-

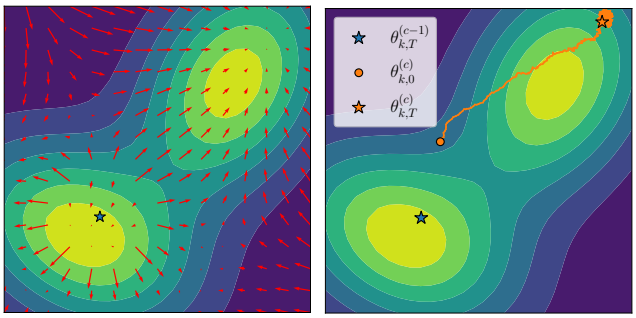
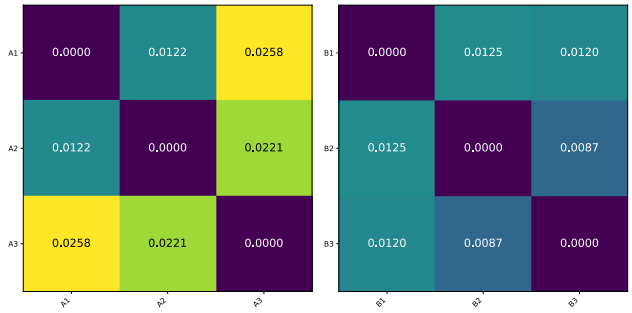


Figure 4. SGMCMC with very high repulsion strength ξ. In this case the repulsive force dominates the sampling trajectory.



(a) Training A with repulsion (b) Training B without repulsion

Figure 5. Wasserstein distance matrix after training on Eurosat, with and without repulsion. Average Wasserstein distance with repulsion is greater.

tinct the internal feature distributions are between models, with a higher distance indicating greater diversity between the feature distributions.

To quantify the diversity of the learned features, we compute the pairwise Wasserstein distances between the latent representations of models from different training cycles. Using a test batch of 100 images, we construct a matrix D with entries $D_{i,j} = W_2(U_{\theta_i}, U_{\theta_j})$. Here, U_{θ_i} is defined as the empirical distribution of the latent features associated with cycle i.

As illustrated in Figure 5, the resulting distance matrices reveal the extent of functional separation between samples. We observe that for Training A, which employs repulsion, the Wasserstein distances between the representation distributions across different cycles are, on average, greater than those observed in Training B.

A.3. Ablations

In this section we ablate the hyperparameters of our method, including the batch size n used to calculate the repulsion force, the repulsion strength ξ, the number of cycles C, and the number of posterior samples K.

Batch size n and repulsion strength ξ. We use $n \in \{16, 32, 64\}$, $\xi \in \{10^{-3}, 10^{-2}, 10^{-1}\}$, and distance $\in$ $\{\text{MMD}, \text{Wasserstein}\}$, with a total of 18 combinations. We show our results in Figure 6 for the UCF dataset. Overall, this figure shows the inherent trade-off associated with repulsion strength. On the one hand, a small repulsion strength is not enough to encourage exploration, resulting in a smaller improvement in performance. On the other hand, large repulsion leads to too much exploration, as we discussed in our toy example (Appendix A.1).

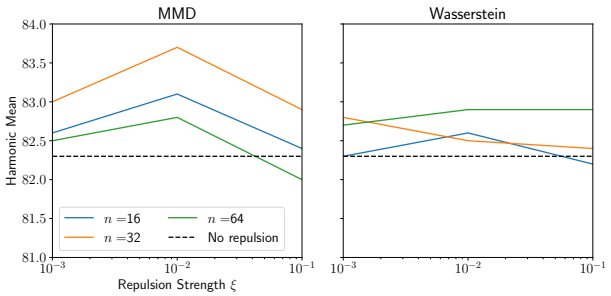


Figure 6. Harmonic mean of base and novel accuracies on the UCF dataset, as a function of repulsion strength ξ for various batch sizes.

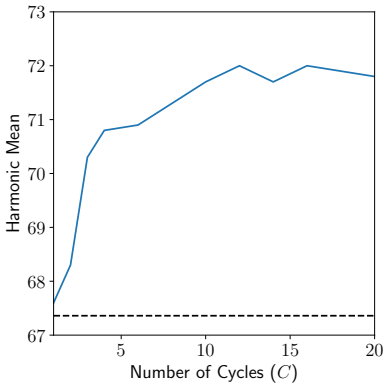


Figure 7. Ablation on the number of cycles in the rcSGHMC algorithm for the DTD dataset.

Number of cycles C. For $C \in \{1, 2, 3, 4, 6, \cdots, 20\}$ in our algorithm. We show our results in Figure 7. First, repulsive cSGMCMC improves over the MaPLe baseline for all number of cycles. Second, running Algorithm 1 for more cycles generally leads to better performance. This is somewhat intuitive, since we are able to acquire more samples from additional modes in the posterior distribution. These findings show that, through repulsion, our method is able to better explore the posterior landscape.

Number of posterior samples. First, we ensemble over $\{\{\theta_{k,T}^{(c)}\}_{c=1}^{C}\}_{k=1}^{K}$, i.e., K samples from C cycles. Our results in Table 1 use $K = 1, C = 3$ (3 total samples). Figure 7 shows the baseline $C = K = 1$ (single sample), where performance is $\approx 67.5\%$ HM, marginally above the no repulsion baseline (dashed line, $\approx 67.4\%$). Increasing C yields clear gains: ≈ 70.5 at $C = 3$ up to $\approx 72\%$ at $C = 10$, before saturating. This ablation demonstrates that multiple posterior samples ($K = 1$ from each cycle) are necessary to achieve the reported performance. For completeness, Table 5 explores performance with varying K on Flower102, showing that performance is stable and gains come primarily from C. We use $K = 1$ in the main paper for efficiency.

# Postrior Samples K	Base	Novel	HM
1	97.43	73.93	84.07
2	97.30	74.10	84.12
3	97.30	74.13	84.14
4	97.37	74.17	84.19
5	97.33	74.23	84.22

Table 5. MaPLe + ReBaPL classification accuracy (%) vs. K, avg. over 3 seeds on Flowers102.

A.4. Overhead of computing probability metrics

The use of probability metrics such as the MMD or the Wasserstein distance does not introduce much overhead, since we compute them on mini-batches of cached representations from the MCMC samples. For reference, computing MMD has computational complexity $\mathcal{O}(n^2)$, where n is the number of samples in the support of the distribution. The Wasserstein distance has $\mathcal{O}(n^3)$ complexity for exact solvers based on linear programming. We refer readers to [29] for more details on these complexities. Table 6 provides a comparison of the MMD and Wasserstein distance running time vs. # of representation samples. In comparison, the entire computational overhead involved in repulsion is ≈ 700 ms, related to the forward and backward passes. Hence, the overhead is negligible.

Batch size	MMD	Wasserstein
2	0.381	0.496
4	0.382	0.523
8	0.396	0.558
16	0.538	0.525
32	0.890	0.657

Table 6. Running time (in miliseconds) of probability metrics for different batch sizes.

A.5. Efficiency Analysis and Inference Parallelism

ReBaPL trains for $E \times C$ total epochs ($E = 5, C = 3$). Training MaPLe for 15 epochs yields similar times (10m40s vs. 10m54s), as repulsion uses cached features ($\approx$ 700ms/iter). Critically, MaPLe at 15 epochs **overfits**: Base/Novel/HM of 97.23/70.90/82.00 vs. 96.33/73.33/83.27 at 5 epochs. Thus, the gain stems from ReBaPL itself, not more compute. Furthermore, ReBaPL inference parallelizes with minimal overhead; see Table 7 for varying C when running on 4 L4 GPUs.

Cycles (C)	MaPLe	MaPLe + ReBaPL
1	242.33	242.39
2	–	259.35
3	–	276.50
4	–	298.91

Table 7. Avg. inference latency (in miliseconds) over 20 runs.

B. Additional Implementation Details

We describe in Table 8 the set of hyperparameters used to run our experiments in Table 1.

Table 8. Hyperparameters used in our ReBaPL experiments for both MMRL and MaPLe.

Hyperparameter	MaPLe	MMRL
Learning rate α	0.002	0.001
Batch size b	1	4
Epochs per cycle T	5	5
# Cycles C	3	3
Samples per cycle K	1	1
Repulsion strength ξ	0.001	10^{-4}
Repulsion batch size n	32	64
Distance	MMD	Wasserstein
Algorithm	rcSGHMC	
Weight decay	5e-4	
Random seed	[1, 2, 3]	

C. Additional Background

C.1. Probability Metrics

We give further details on the probability metrics introduced in Section 2.3, especially Equations 13 and 14. Here, we cover the essential theory behind the MMD and the Wasserstein distance. For further details, we refer readers to [15] and [29].

Maximum Mean Discrepancy. Let $\mathcal{F}$ be a family of test functions over $\mathcal{U}$. The MMD is defined by,

$$\text{MMD}(p, q) = \sup_{f \in \mathcal{F}} \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)].$$

When $\mathcal{F}$ is a reproducing kernel Hilbert space (see [15, Section 2.2]) with kernel $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, the squared MMD is expressed in terms of the mean embeddings of p and q,

$$\text{MMD}(p, q)^2 = \|\mu_p - \mu_q\|_2^2.$$

Here, the mean embedding is the element $\mu_p \in \mathcal{F}$ such that $\langle f, \mu_p \rangle_{\mathcal{F}} = \mathbb{E}_{x\sim p}[f]$, $\forall f \in \mathcal{F}$. By [15, Lemma 6], the squared MMD admits the form,

$$\text{MMD}(p, q)^2 = \mathop{\mathbb{E}}_{x\sim p, x'\sim p}[\kappa(x, x')] + \mathop{\mathbb{E}}_{y\sim q, y'\sim q}[\kappa(y, y')] - 2 \mathop{\mathbb{E}}_{x\sim p, y\sim q}[\kappa(x, y)], \tag{19}$$

which then admits the *unbiased* estimator in Equation 13. Due to the double summations in Equation 13, and with the

assumption that $n \approx m$, the computational complexity of computing the empirical MMD is $\mathcal{O}(n^2)$.

Wasserstein distance. This distance is rooted in the theory of OT [29, 39], and gives the *least amount of effort or energy* required to move the probability mass from p to q. In other words, consider the set $\Gamma(p,q)$ so that $\gamma \in \Gamma(p,q)$ satisfies,

$$\int_{\mathcal{U}} \gamma(x,y)dy = p(x) \qquad \int_{\mathcal{U}} \gamma(x,y)dx = q(y).$$

The elements of γ are called *transport plans*, and can be conceptualized as joint distributions with marginals p and q. Note that $\Gamma(p,q)$ also impose mass preservation constraints on the transportation plans. With these concepts,

$$\gamma^\star = \underset{\gamma\in\Gamma(p,q)}{\text{arginf}} \int_{\mathcal{U}}\int_{\mathcal{U}} c(x,y)\gamma(x,y)dxdy, \tag{20}$$

where $c : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ is called the *ground-cost*, i.e., a function that measures the effort of moving x to y.

The problem in Equation 20 can induce a metric on $\mathcal{P}(\mathcal{U})$ under certain conditions. Especially, let $(\mathcal{U}, d_{\mathcal{U}})$ be a metric space, and $c(x,y) = d(x,y)^\alpha$, $\alpha \in [1, +\infty)$. The OT cost under these conditions defines the $\alpha-$Wasserstein distance,

$$W_\alpha(p,q)^\alpha = \inf_{\gamma\in\Gamma} \int_{\mathcal{U}}\int_{\mathcal{U}} d(x,y)^\alpha\gamma(x,y)dxdy. \tag{21}$$

This is an infinite dimensional program on the variable γ. Given samples $x_i^{(p)} \sim p$ and $y_j^{(q)} \sim q$, the Wasserstein distance admits an empirical estimator given by Equation 14. In this case, it is calculated through a finite linear program with $n \times m$ variables. In that sense, its computational complexity is $\mathcal{O}(n^3)$.

Remark. *(Entropic Regularization) An alternative to Equation 14 is to consider the Sinkhorn divergence [7, 14], which is a regularized version of the OT problem, which has $\mathcal{O}(n^2)$ complexity* per iteration*. However, in small sample scenarios (e.g., at the level of a mini-batch), Sinkhorn divergence usually needs many iterations to accurately estimate the transportation plan, which leads to a running time that is comparable or worse than exact OT. For that reason, in our experiments, we use the standard Wasserstein distance.*

Remark. *(Mini-batch estimation) In the main paper (c.f. Equations 13 and 14), we presented the empirical estimators of the continuous MMD and Wasserstein distance (c.f. Equations 19 and 21). These estimators assume n and m i.i.d. samples from distributions p and q, respectively. Now, note that our main interest is computing $d_{\mathcal{P}(\mathcal{U})}(U_\theta, U_{\theta'})$, where $U_\theta = \{u_{\theta,i}\}_{i=1}^n$ and $U_{\theta'} = \{u_{\theta',i}\}_{i=1}^n$. In other words, we understand U_θ and $U_{\theta'}$ as the i.i.d. samples from the distribution of representations produced by networks θ and θ', respectively.*

For the sake of completeness, the MMD reads as,

$$MMD(U_\theta, U_{\theta'})^2 = \frac{1}{n^2}\Bigg(\sum_{i,j}\kappa(u_{\theta,i}, u_{\theta,j}) + \sum_{i,j}\kappa(u_{\theta',i}, u_{\theta',j}) - 2\sum_{i,j}\kappa(u_{\theta,i}, u_{\theta',j})\Bigg),$$

and the Wasserstein distance,

$$W_2(U_\theta, U_{\theta'})^2 = \sum_{i=1}^n\sum_{j=1}^n \gamma_{ij}^\star \|u_{\theta,i} - u_{\theta',j}\|_2^2,$$

where $\gamma^\star$ is obtained through linear programming.