Supplementary Material: UniPROT: UNIform PRtotype selection via Partial Optimal Transport with Submodular Guarantees

CONTENTS

Supplementary Material: UniPROT: UNIform PRtotype selection via Partial Optimal Transport with Submodular Guarantees

A Code

We release our code at the following link

A.1 Submodularity Ratio

The notion of submodularity ratio is given by approximate submodularity in (Das & Kempe, 2018b). For a monotone function f the submodularity ratio w.r.t a set S and a parameter $k \geq 0$ as

$$\alpha_{L,K}(f) = \min_{\substack{\mathcal{S}\subseteq L,\, \mathcal{A}\subseteq L \\ |\mathcal{A}|\leq K,\, \mathcal{A}\cap\mathcal{S}=\emptyset}} \frac{\sum_{u\in A} f(\mathcal{S}\cup\{u\}) - f(\mathcal{S})}{f(\mathcal{S}\cup A) - f(\mathcal{S})}, \quad \text{with } \frac{0}{0} := 1.$$

f is submodular if and only if $\alpha_{L,K}(f) \geq 1$. If the ratio

$$\alpha := \frac{\sum_{u\in A} f(\mathcal{S}\cup\{u\}) - f(\mathcal{S})}{f(\mathcal{S}\cup\mathcal{A}) - f(\mathcal{S})}$$

is strictly positive but not necessarily greater than 1, then f is said to be α-weakly submodular.

B Theoretical Results

Lemma 1. *The set function $h(\mathcal{P}) : 2^{|\mathcal{S}|} \to \mathbb{R}_+$, defined in (6), satisfies the following properties:*

1. *Non-negativity: $h(\mathcal{P}) \geq 0 \ \forall \mathcal{P} \subseteq \mathcal{S}$.*
2. *Monotonicity: $h(\mathcal{P}_2) \geq h(\mathcal{P}_1) \ \forall \mathcal{P}_1 \subseteq \mathcal{P}_2 \subseteq \mathcal{S}$.*
3. *Super-additivity over disjoint sets: $h(\mathcal{P}_1 \cup \mathcal{P}_2) \geq h(\mathcal{P}_1) + h(\mathcal{P}_2) \ \forall \mathcal{P}_1 \cap \mathcal{P}_2 = \phi$.*

Proof. We prove each property in turn (refer to the definition of $h(\mathcal{P})$ in (6)).

1. Non-negativity. The non-negativity follows from the definition of $h(\cdot)$ in (6) namely, $h(\mathcal{P}) := \max_{\boldsymbol{\gamma}\in\Gamma(\mathbf{1}_{\mathcal{P}}, |\mathcal{P}|\mathbf{1}/n)} \langle \mathbf{S}, \boldsymbol{\gamma}\rangle$, where the similarity matrix $\mathbf{S}$ is a non-negative matrix and the transport plan is enforced to non-negative.

2. Monotonicity. Consider a subset $\mathcal{P}_1$ and define a set $\mathcal{P}_2 = \mathcal{P}_1 \cup \{\mathbf{x}_i\}$ for any $\mathbf{x}_i \notin \mathcal{P}_1$. To prove monotonicity of $h(.)$, it is sufficient to show that $h(\mathcal{P}_2) \geq h(\mathcal{P}_1)$. To this end, let $\boldsymbol{\gamma}_{\mathcal{P}_1}$ be the argmax for $h(\mathcal{P}_1)$ and consider the sub-matrix $\boldsymbol{\gamma}_{\mathcal{P}_1}(\mathcal{I}_{\mathcal{P}_1}, :)$ which is the restriction of the optimal solution to the points in $\mathcal{P}_1$. We note that $\boldsymbol{\gamma}_{\mathcal{P}_1}(i, :) = \mathbf{0}; \mathbf{x}_i \notin \mathcal{P}_1$. We construct a feasible transport plan $\hat{\boldsymbol{\gamma}}$ for the set $\mathcal{P}_2$ as:

$$\hat{\boldsymbol{\gamma}}(\mathcal{I}_{\mathcal{P}_2}, :) = \left[\boldsymbol{\gamma}_{\mathcal{P}_1}(\mathcal{I}_{\mathcal{P}_1}, :)^\top, \frac{\mathbf{1}}{n}\right]^\top,$$

and $\hat{\boldsymbol{\gamma}}(j, :) = \mathbf{0}$ for $\mathbf{x}_j \notin \mathcal{P}_2$. Let $\hat{h}(\mathcal{P}_2; \hat{\boldsymbol{\gamma}})$ indicate the function value evaluated at the feasible transport plan $\hat{\boldsymbol{\gamma}}$ for the set $\mathcal{P}_2$. We then have

$$\begin{aligned} h(\mathcal{P}_2) \geq \hat{h}(\mathcal{P}_2; \hat{\boldsymbol{\gamma}}) &= \langle \mathbf{S}(\mathcal{I}_{\mathcal{P}_2}, :), \hat{\boldsymbol{\gamma}}(\mathcal{I}_{\mathcal{P}_2}, :)\rangle \\ &= \langle \mathbf{S}(\mathcal{I}_{\mathcal{P}_1}, :), \boldsymbol{\gamma}_{\mathcal{P}_1}(\mathcal{I}_{\mathcal{P}_1}, :)\rangle + \left\langle \mathbf{S}(i, :), \frac{\mathbf{1}}{n}\right\rangle \\ &= h(\mathcal{P}_1) + \left\langle \mathbf{S}(i, :), \frac{\mathbf{1}}{n}\right\rangle \\ &\geq h(\mathcal{P}_1) (\text{Since } \mathbf{S}(i, :) \geq \mathbf{0}.) \end{aligned} \tag{9}$$

3. Super-additivity over disjoint sets. Consider two disjoints sets $\mathcal{P}_1$ and $\mathcal{P}_2$. Let $\boldsymbol{\gamma}_{\mathcal{P}_1}(\mathcal{I}_{\mathcal{P}_1}, :)$ and $\boldsymbol{\gamma}_{\mathcal{P}_2}(\mathcal{I}_{\mathcal{P}_2}, :)$ represent the sub-matrices of the respective optimal solutions to the points in $\mathcal{P}_1$ and $\mathcal{P}_2$. For the disjoint union set $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$, we construct a feasible transport plan $\hat{\boldsymbol{\gamma}}$ as:

$$\hat{\boldsymbol{\gamma}}(\mathcal{I}_{\mathcal{P}}, :) = \left[\boldsymbol{\gamma}_{\mathcal{P}_1}(\mathcal{I}_{\mathcal{P}_1}, :)^\top, \boldsymbol{\gamma}_{\mathcal{P}_2}(\mathcal{I}_{\mathcal{P}_2}, :)^\top\right]^\top,$$

and $\hat{\gamma}(j,:) = \mathbf{0}$ for $\mathbf{x}_j \notin \mathcal{P}$. Evaluating the function $\hat{h}(\mathcal{P};\hat{\gamma})$ at the feasible solution, we get

$$\begin{aligned} h(\mathcal{P}) \geq \hat{h}(\mathcal{P};\hat{\gamma}) &= \langle \mathbf{S}(\mathcal{I}_{\mathcal{P}},:), \hat{\gamma}(\mathcal{I}_{\mathcal{P}},:)\rangle \\ &= \langle \mathbf{S}(\mathcal{I}_{\mathcal{P}_1},:), \gamma_{\mathcal{P}_1}(\mathcal{I}_{\mathcal{P}_1},:)\rangle + \langle \mathbf{S}(\mathcal{I}_{\mathcal{P}_2},:), \gamma_{\mathcal{P}_2}(\mathcal{I}_{\mathcal{P}_2},:)\rangle \\ &= h(\mathcal{P}_1) + h(\mathcal{P}_2) \end{aligned} \tag{10}$$

□

Lemma 2. *The optimization problem defined in (7) is a non-negative, monotone, submodular maximization problem subject to cardinality constraint k.*

Proof. We derive all the three properties below.

1. Non-negativity: $f(\mathcal{P}) \geq 0 \ \forall \mathcal{P} \subseteq \mathcal{S}$. The proof follows along similar lines of the non-negativity proof in Lemma 1.

2. Monotonicity: $f(\mathcal{P}_2) \geq f(\mathcal{P}_1) \ \forall \mathcal{P}_1 \subseteq \mathcal{P}_2 \subseteq \mathcal{S}$. Akin to the monotonicity proof in Lemma 1, for a super-set $\mathcal{P}_2 = \mathcal{P}_1 \cup \{\mathbf{x}_i\}; \mathbf{x}_i \notin \mathcal{P}_1$, we can construct a feasible transport $\hat{\gamma}$ using the optimal solution $\gamma_{\mathcal{P}_1}$ as:

$$\hat{\gamma}(\mathcal{I}_{\mathcal{P}_2},:) = \left[\gamma_{\mathcal{P}_1}(\mathcal{I}_{\mathcal{P}_1},:)^\top, \frac{\mathbf{v}}{\|\mathbf{v}\|_1}\right]^\top,$$

where $\mathbf{v} = k\mathbf{1}/n - \gamma_{\mathcal{P}_1}^\top \mathbf{1} \geq \mathbf{0}$. Following similar lines to the argument in Lemma 1, we obtain the monotonicity result.

3. Submodularity: To prove submodularity of function $f(\mathcal{P})$ in (7), we first note the following result (Kawano et al., 2022, Lemma 2).

Lemma 6. *[(Kawano et al., 2022, Lemma 2)] Let l, m, n be positive integers. Given a positive valued $m \times n$ matrix $\mathbf{S} > 0$, the following set function $\psi : 2^m \to \mathbb{R}_+$ is a submodular function:*

$$\psi(\mathcal{P}) = \max_{\gamma \in \Gamma_{\leq}(\mathbf{1}_{\mathcal{P}}/l, \mathbf{1}_n/n)} \langle \mathbf{S}, \gamma \rangle \tag{11}$$

where, as defined earlier, $\Gamma_{\leq}(\boldsymbol{\mu}, \boldsymbol{\nu}) = \{\gamma \in \mathbb{R}^{m \times n} \mid \gamma \geq 0, \gamma\mathbf{1} = \boldsymbol{\mu}, \gamma^\top \mathbf{1} \leq \boldsymbol{\nu}\}$ and $\mathbf{1}_n$ is a $n \times 1$ vector of 1.

We observe that for $l = k$, (11) is equivalent to the proposed function $f(\cdot)$ defined in (7) as follows:

- For a given $\mathcal{P}$, let γ_1 be an optimal solution for (11). Then, $\gamma_2 = k\gamma_1$ is an optimal coupling for computing $f(\mathcal{P})$ in (7). Similarly, if γ_2 be an optimal solution for computing $f(\mathcal{P})$ in (7), then $\gamma_1 = \gamma_2/k$ is an optimal solution for computing $f(\mathcal{P})$ in (11).
- Hence, for a given $\mathcal{P}$, $f(\mathcal{P}) = k\psi(\mathcal{P})$

Due to the above, $\forall A, B \subseteq \mathcal{S}$

$$\psi(A \cup B) + \psi(A \cap B) \leq \psi(A) + \psi(B) \Rightarrow f(A \cup B) + f(A \cap B) \leq f(A) + f(B),$$

which proves that f is a submodular function.

□

Lemma 3. *Let $\mathcal{P}^*$ of cardinality k be an optimal solution of (6). Then $\mathcal{P}^*$ is also an optimal solution of (7), and vice-versa.*

Proof. Recall that for any set $\mathcal{P}$ of cardinality k, $f(\mathcal{P}) = h(\mathcal{P})$. Due the monotonicity properties in Lemmas 1 and 2, we can restrict the feasible region in problems (6) and (7) only across sets of cardinality k where they are equivalent, and have the same optimal solution. □

Lemma 4. *Let $\hat{\mathcal{P}}$ be the classical greedy solution of (7) with $|\hat{\mathcal{P}}| = k$. Let $\mathrm{OPT} = h(\mathcal{P}^*)$, where $\mathcal{P}^*$ is an optimal solution of (6). Then, $h(\hat{\mathcal{P}}) \geq (1 - 1/e)\mathrm{OPT}$.*

Proof. Recall that for any set $\mathcal{P}$ with $|\mathcal{P}| \leq k$, $f(\mathcal{P}) \geq h(\mathcal{P})$. As $|\hat{\mathcal{P}}| = k$, we have the equality $f\left(\hat{\mathcal{P}}\right) = h\left(\hat{\mathcal{P}}\right)$. Applying the classical greedy approximation theorem in (Nemhauser et al., 1978), we get $h\left(\hat{\mathcal{P}}\right) = f\left(\hat{\mathcal{P}}\right) \geq (1 - 1/e)\, f\left(\mathcal{P}^*\right) \geq (1 - 1/e)\, OPT$. □

Lemma 5. *Let $\alpha_{j,\min}$ denote $\frac{1}{\lfloor n/k \rfloor}$ times the sum of the $\lfloor n/k \rfloor$ smallest entries of the vector $\mathbf{S}(j,:)$, and let $\alpha_{j,\max}$ denote $\frac{1}{\lfloor n/k \rfloor}$ times the sum of the $\lfloor n/k \rfloor$ largest entries of $\mathbf{S}(j,:)$. Define $\alpha = \min_{j\in[m]} \frac{\alpha_{j,\min}}{\alpha_{j,\max}}$. Let $\hat{\mathcal{P}}$ be the solution returned by the greedy algorithm for (7), where the proposed approximate marginal gain function (8) is used in each iteration and $|\hat{\mathcal{P}}| = k$. Then, $f(\hat{\mathcal{P}}) = h(\hat{\mathcal{P}}) \geq (1 - e^{-\alpha})\,\mathrm{OPT}$, where $\mathrm{OPT} = f(\mathcal{P}^*) = h(\mathcal{P}^*)$ and $\mathcal{P}^*$ is an optimal solution to (7) with $|\mathcal{P}^*| = k$.*

Proof. At the iteration $i+1$, let $\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{S}\setminus\mathcal{P}_i}{\arg\max}\, \hat{f}(\mathbf{x}|\mathcal{P}_i)$ be the point that maximizes the approximate marginal gain function (8), which is used to update the solution to $\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\mathbf{x}^*\}$. Then,

$$f\left(\mathbf{x}^*|\mathcal{P}_i\right) \geq \hat{f}\left(\mathbf{x}^*|\mathcal{P}_i\right) \geq \frac{1}{k} \sum_{\mathbf{x}_l\in\mathcal{P}^*\setminus\mathcal{P}_i} \left[\hat{f}\left(\mathbf{x}_l|\mathcal{P}_i\right)\right] \tag{12}$$

Further, for any $\mathbf{x}_l \notin \mathcal{P}_i$ let $\mathcal{P} = \mathcal{P}_i \cup \{\mathbf{x}_l\}$. We derive the inequality

$$\begin{aligned} f\left(\mathbf{x}_l|\mathcal{P}_i\right) = f\left(\mathcal{P}\right) - f\left(\mathcal{P}_i\right) &= \left\langle \mathbf{S}\left(\mathcal{I}_{\mathcal{P}},:\right), \boldsymbol{\gamma}_{\mathcal{P}}\left(\mathcal{I}_{\mathcal{P}},:\right)\right\rangle - \left\langle \mathbf{S}\left(\mathcal{I}_{\mathcal{P}_i},:\right), \boldsymbol{\gamma}_{\mathcal{P}_i}\left(\mathcal{I}_{\mathcal{P}_i},:\right)\right\rangle \\ &\leq \left\langle \mathbf{S}\left(\mathcal{I}_{\mathcal{P}},:\right), \boldsymbol{\gamma}_{\mathcal{P}}\left(\mathcal{I}_{\mathcal{P}},:\right)\right\rangle - \left\langle \mathbf{S}\left(\mathcal{I}_{\mathcal{P}_i},:\right), \boldsymbol{\gamma}_{\mathcal{P}}\left(\mathcal{I}_{\mathcal{P}_i},:\right)\right\rangle \\ &= \left\langle \mathbf{S}(l,:), \boldsymbol{\gamma}_{\mathcal{P}}(l,:)\right\rangle \\ &\leq \alpha_{l,\max}. \end{aligned}$$

The inequality in the second line follows from the fact that $\boldsymbol{\gamma}_{\mathcal{P}_i}$ is the $\arg\max$ for the set $\mathcal{P}_i$ in (7) and $\boldsymbol{\gamma}_{\mathcal{P}}\left(\mathcal{I}_{\mathcal{P}_i},:\right)$ —appended with $\mathbf{0}$ for other rows— is one of the feasible solution. Likewise, $\hat{f}\left(\mathbf{x}_l|\mathcal{P}_i\right) \geq \alpha_{l,\min}$. Hence,

$$\hat{f}\left(\mathbf{x}_l|\mathcal{P}_i\right) \geq \frac{\alpha_{l,\min}}{\alpha_{l,\max}} f\left(\mathbf{x}_l|\mathcal{P}_i\right). \tag{13}$$

Pugging the inequality (13) in (12) we have

$$f\left(\mathbf{x}^*|\mathcal{P}_i\right) \geq \frac{1}{k} \sum_{\mathbf{x}_l\in\mathcal{P}^*\setminus\mathcal{P}_i} \left[\frac{\alpha_{l,\min}}{\alpha_{l,\max}} f\left(\mathbf{x}_l|\mathcal{P}_i\right)\right] \geq \frac{\alpha}{k} \sum_{\mathbf{x}_l\in\mathcal{P}^*\setminus\mathcal{P}_i} \left[f\left(\mathbf{x}_l|\mathcal{P}_i\right)\right]. \tag{14}$$

Leveraging the submodular and monotonic properties of the function $f(\cdot)$ from Lemma 2, we obtain the inequalities

$$\sum_{\mathbf{x}_l\in\mathcal{P}^*\setminus\mathcal{P}_i} f\left(\mathbf{x}_l|\mathcal{P}_i\right) \geq f\left(\mathcal{P}_i \cup \left(\mathcal{P}^* \setminus \mathcal{P}_i\right)\right) - f\left(\mathcal{P}_i\right) \geq f\left(\mathcal{P}^*\right) - f\left(\mathcal{P}_i\right). \tag{15}$$

Noting that $f\left(\mathbf{x}^*|\mathcal{P}_i\right) = \left[f\left(\mathcal{P}^*\right) - f\left(\mathcal{P}_i\right)\right] - \left[f\left(\mathcal{P}^*\right) - f\left(\mathcal{P}_{i+1}\right)\right]$, and using (15) in (14) gives the recurrence relation

$$f\left(\mathcal{P}^*\right) - f\left(\mathcal{P}_{i+1}\right) \leq \left(1 - \frac{\alpha}{k}\right)\left[f\left(\mathcal{P}^*\right) - f\left(\mathcal{P}_i\right)\right],$$

frow which it follows that

$$f\left(\mathcal{P}^*\right) - f\left(\hat{\mathcal{P}}\right) \leq \left(1 - \frac{\alpha}{k}\right)^k \left[f\left(\mathcal{P}^*\right) - f\left(\varnothing\right)\right].$$

As $f\left(\varnothing\right) = 0$, we get the desired result namely,

$$f\left(\hat{\mathcal{P}}\right) \geq \left(1 - \left(1 - \frac{\alpha}{k}\right)^k\right) f\left(\mathcal{P}^*\right) \geq (1 - e^{-\alpha})\,\mathrm{OPT}.$$

□

C ALGORITHM DETAILS

Algorithm 1: `UniPROT`

Input: Similarity matrix $\mathcal{S}$ between $\mathcal{S}$ and $\mathcal{T}$, number of prototypes required k, entropic regularization parameter λ

Output: Uniformly weighted prototypical set $\mathcal{P}_k \subseteq \mathcal{S}$ of $\mathcal{T}$

1 $\mathcal{P}_0 \leftarrow \varnothing$;

2 **for** $i = 1$ **to** k **do**

3 $\quad \boldsymbol{\gamma}^*_{\mathcal{P}_i} \leftarrow \arg\max_{\boldsymbol{\gamma} \in \Gamma_{\leq}(\mathbf{1}_{\mathcal{P}_i}, k\mathbf{1}_n/n)} \langle \mathbf{S}, \boldsymbol{\gamma} \rangle - \lambda \langle \boldsymbol{\gamma}, \ln \boldsymbol{\gamma} \rangle$

$\quad \mathbf{x}^* \leftarrow \arg\max_{\mathbf{x} \in \mathcal{S} \setminus \mathcal{P}_i} \hat{f}(\mathbf{x}|\mathcal{P}_i)$ From (8)

$\quad \mathcal{P}_{i+1} \leftarrow \mathcal{P}_i \cup \{\mathbf{x}^*\}$

4 **return** $\mathcal{P}_k$

To solve the partial optimal transport problem in Step 3 we use Bregman-Dykstra iterations (Benamou et al., 2015).

D IMPLEMENTATION DETAILS

D.1 HARDWARE AND LICENSE

All models are implemented in `Python 3.10` using `PyTorch 2.3.0`. Image and language training are performed on servers with Intel(R) Xeon(R) Gold 6226R CPUs (2.90GHz) and three NVIDIA RTX A6000 GPUs. For language model pretraining, we use `JAX` (Bradbury et al., 2018) (v0.7.2) on the same GPU infrastructure.

D.2 ALGORITHM IMPLEMENTATION

Implementation of POT Objective: To solve the entropic-regularized partial optimal transport (OT) problem, we rely on the Python Optimal Transport (POT) library[1] . Specifically, we use the function `ot.partial.entropic_wasserstein`[2], which implements the entropic-regularized variant of partial OT. This formulation allows for transporting only a fraction of the total mass between the source and target distributions along with the enforcement of inequality on the marginals.

In our implementation, the cost matrix C is constructed using pairwise distances between features of the source and target prototypes, which could be Euclidean or cosine distances depending on the application. The fraction of transported mass τ and the entropic regularization λ are treated as hyperparameters. The function `ot.partial.entropic_wasserstein` efficiently returns both the optimal transport plan and the associated partial OT cost, which we use as the objective function $f(\cdot)$ in downstream optimization or prototype selection procedures.

A typical usage in Python is as follows:

```
import ot

# mu: source weights
# nu: target weights
# C: cost matrix
# tau: fraction of transported mass
# lambda: entropy regularization
T, cost = ot.partial.entropic_wasserstein(
    mu, nu, C,
    tau=tau,
    reg=lambda
)
```

The default maximum iterations parameter for the above function is set adaptively along with a stopping threshold of $1e-6$.

[1] https://pythonot.github.io/

[2] https://pythonot.github.io/gen_modules/ot.partial.html#ot.partial.entropic_partial_wasserstein

Table 3: Source Size vs Maximum Iterations

Source Set Size	Max Iterations
64-200	100
500-1000	1000
1000-4000	2000
5000	4000

D.3 FINETUNING EXPERIMENTS

We adapt the codebase of Nguyen et al. (2025) for all our finetuning experiments. We use adam Adam et al. (2014) with learning rate of 1e-5, gradient accumulation steps of 64 with batch size 1. We directly use raw lora gradients for constructing similarity matrices for CoLM, GREATS, `UniPROT`. For GREATS Wang et al. (2024) we randomly sample 2-random points from train-set as anchors at every train step. For SBERT Reimers & Gurevych (2019), we use BERT-BASE-UNCASED as the embedding model, and construct similarity matrix from the embeddings instead of gradients.

D.4 DETAILS OF BASELINES

GREATS Wang et al. (2024). GREATS formulates online batch selection as optimizing a set utility that measures the single-step reduction in validation loss under a gradient-descent update. Let w_t be the current parameters, B_t a candidate batch, and $S \subseteq B_t$ a subset of size k. The ideal utility at iteration t is

$$U^{(t)}(S; \mathbf{z}^{(\mathrm{val})}) \;:=\; \ell(\boldsymbol{\theta}_t, \mathbf{z}^{(\mathrm{val})}) \;-\; \ell\Big(\boldsymbol{\theta}_t - \eta_t \sum_{\mathbf{z}\in S} \nabla \ell(\boldsymbol{\theta}_t, \mathbf{z}),\, \mathbf{z}^{(\mathrm{val})}\Big),$$

and selection solves $\arg\max_{S\subseteq B_t, |S|=k} U^{(t)}(S; \mathbf{z}^{(\mathrm{val})})$. Since exact evaluation is intractable, GREATS applies a lower-order Taylor approximation of the validation loss around $\boldsymbol{\theta}_t$ to obtain a closed-form surrogate for the marginal gain of adding a training point $\mathbf{z}$:

$$U^{(t)}(\mathbf{z} \mid S) \;\approx\; \eta_t\, \mathbf{g}(\mathbf{z})^\top \mathbf{g}\big(\mathbf{z}^{(\mathrm{val})}\big) \;-\; \eta_t^2\, \mathbf{g}(\mathbf{z})^\top H\big(\mathbf{z}^{(\mathrm{val})}\big)\, \mathbf{g}(\mathbf{z}^*),$$

where $\mathbf{g}(\cdot) = \nabla \ell(\boldsymbol{\theta}_t, \cdot)$, $\boldsymbol{\mathcal{H}}(\cdot)$ is the Hessian of the validation loss, and z^* denotes the current aggregate. In practice, $\boldsymbol{\mathcal{H}}$ is approximated (e.g., $\boldsymbol{\mathcal{H}} \approx I$), yielding a gradient inner-product scoring with a correction term. A greedy procedure iteratively adds the point with largest approximate marginal gain until k points are selected. To avoid materializing per-example model-sized gradients, GREATS computes all required gradient inner-products in a single backpropagation via a "ghost inner-product" reparameterization that expresses layerwise gradient inner-products using already-available activations and output gradients, and merges selection with the update without extra passes.

CoLM (Nguyen et al., 2025). CoLM casts mini-batch construction as coreset selection in gradient space for memory-efficient fine-tuning. Let a large random batch be partitioned by sources V_q. CoLM first addresses imbalance by including *all* examples from "small" sources (those with insufficient sample count in the large batch), while selecting representatives (medoids) from each "big" source. To align selection with Adam, per-example gradients are normalized by the optimizer's exponential-moving-average statistics, yielding normalized directions proportional to $m_t/(\epsilon+\sqrt{v_t})$. To reduce dimensionality and denoise, CoLM estimates the gradient of the *last V-projection* parameters (e.g., LoRA V) using a zeroth-order SPSA estimator with two perturbed forward passes and precached penultimate activations, then sparsifies by keeping the coordinates with largest normalized magnitudes. Within each big source, a greedy medoid selection is performed in the projected, sparsified, Adam-normalized gradient space so that the aggregated coreset gradient approximates that of the full large batch; the final mini-batch is the union of all small-source examples and the selected big-source medoids.

SBERT Reimers & Gurevych (2019). SBERT modifies BERT into siamese/triplet architectures with shared weights that encode each sentence independently. A fixed-size embedding $u \in \mathbb{R}^d$ is obtained via a pooling operation over token representations (commonly mean pooling). Training uses sentence-pair supervision: (i) a classification objective on NLI pairs, where a classifier consumes a concatenation of functions of the two embeddings (e.g., $[u; v; |u - v|]$) to predict the label; (ii) a regression objective for semantic textual similarity, where the cosine of (u, v) is regressed to a gold score via MSE; and (iii) optionally, triplet loss $\max\{0, \cos(u_a, u_n) - \cos(u_a, u_p) + \gamma\}$ for

anchor–positive–negative tuples. At inference, sentence embeddings are compared with cosine or dot-product for retrieval and clustering.

Maxloss (Shalev-Shwartz & Wexler, 2016)

D.5 Calculation of gradient features

Let $\mathbf{z}_i$ denote an example, $\ell(\boldsymbol{\theta}; \mathbf{z}_i)$ the training loss, and let $\mathbf{W}^{(L)}_{V,\text{LoRA}}$ be the parameter tensor of the *last* transformer block's value projection adapted by LoRA.[3] We flatten $\mathbf{W}^{(L)}_{V,\text{LoRA}}$ to a vector $v \in \mathbb{R}^{d_{\text{vp}}}$. At iteration t and current parameters $\boldsymbol{\theta}_t$, we compute per-example gradients

$$\mathbf{g}^{\text{vp}}_{i,t} \;:=\; \nabla_v \, \ell(\boldsymbol{\theta}_t; \mathbf{z}_i) \;\in\; \mathbb{R}^{d_{\text{vp}}},$$

restricted to the LoRA-adapted last V-projection. Unlike the zeroth-order MeZO estimator used in Nguyen et al. (2025), these gradients are obtained directly by backpropagation.

Adam-aligned normalization. To align with the update rule of Adam, we normalize each per-example gradient using the optimizer's moment statistics. Let $m_t, v_t \in \mathbb{R}^{d_{\text{vp}}}$ denote the first and second moment accumulators,

$$m_t \;=\; \beta_1 m_{t-1} + (1-\beta_1)\,\bar{\mathbf{g}}_t, \qquad v_t \;=\; \beta_2 v_{t-1} + (1-\beta_2)\,\bar{\mathbf{g}}_t^{\odot 2}, \tag{16}$$

with $\beta_1, \beta_2 \in (0,1)$, $\epsilon > 0$, and $\bar{\mathbf{g}}_t$ the average gradient over the current pool. The normalized gradient feature for an example $\mathbf{z}_i$ is then

$$\phi_{i,t} \;:=\; \frac{g^{\text{vp}}_{i,t}}{\epsilon + \sqrt{v_t}} \;\in\; \mathbb{R}^{d_{\text{vp}}}, \tag{17}$$

where the division is elementwise. These features are stored and subsequently used in similarity computations.

D.6 Pretraining Experiments

We implement the pretraining setup using `JAX` (Bradbury et al., 2018), primarily due to its just-in-time (JIT) compilation framework and empirical 2–3× training speedups over `PyTorch`. For the base architecture, we pretrain a LLaMA-3 model consisting of approximately 500M parameters on the OpenWebText corpus[4] (Radford et al., 2019). The training is carried out for 20k training steps with an effective batch size of 64 sequences, each of length 512. Optimization is performed using the Adam algorithm with a fixed learning rate of 1×10^{-4}, without auxiliary learning rate schedules.

For all methods involving prototype-based subset selection, we begin with a candidate batch of 128 sequences and select 50% prototypes, resulting in an effective batch size of 64 sequences used for parameter updates. Subset selection is performed on a per-batch basis, without leveraging history across iterations.

In the case of `UniPROT`, the underlying optimal transport (OT) problem is solved using 20 Sinkhorn iterations with entropic regularization strength set to 1×10^{-2}. The similarity (cost) matrices are constructed directly from the last-layer gradients of the model, and no additional low-pass filtering, smoothing, or adaptive reweighting is applied. Throughout, cosine similarity is used as the base kernel to define pairwise affinities.

The dataset is partitioned into a 95% training split and a 5% validation split, where the validation portion is reserved for monitoring generalization and reporting final performance metrics.

E Experimental Setup Details

E.1 Model Details

Phi-2 (2.7B). Phi-2 is a 2.7B parameter model trained with an emphasis on mathematical and logical reasoning, derived from curated synthetic corpora and filtered web data. It supports a context length of 2,048 tokens. In our fine-tuning setup, we apply LoRA adapters (rank 128, $\alpha = 512$, dropout 0.05) to all attention projection matrices (QKV) and the two feed-forward layers.

[3] "Last" refers to the topmost transformer block in the forward stack.

[4] https://huggingface.co/datasets/Skylion007/openwebtext

PHI-3 family. We experiment primarily with the 3.8B variant (PHI-3 MINI), though the broader family also includes 7B and 14B models. The PHI-3 series continues the focus on compact models optimized for reasoning tasks, with available context lengths of 4K and 128K tokens depending on variant. Similar to PHI-2, we apply LoRA adapters to QKV projections and feed-forward layers during fine-tuning.

STABLELM ZEPHYR 3B. STABLELM ZEPHYR 3B is a 3B parameter instruction-tuned model designed as a general-purpose assistant, without a specific emphasis on mathematical reasoning. It supports input sequences up to 4K tokens. For LoRA fine-tuning, we insert adapters into all attention projection matrices (QKVO).

E.2 DATASETS

For image settings we do on MNIST, CIFAR10, CIFAR100 / INaturalist as well as synthetic distributions.

For the mathematical reasoning experiments, we fine-tune on the MATHINSTRUCT dataset (**?**), which contains roughly 260K instruction–response pairs. The data is aggregated from 14 open-source mathematics corpora, covering diverse subfields and spanning a broad range of difficulty levels. The composition of MathInstruct is highly imbalanced—the largest constituent source is nearly 300 times larger than the smallest—and the detailed distribution across sources is provided in Figure 4a of the Appendix. Prior work has shown that fine-tuning on MathInstruct leads to state-of-the-art results on multiple standardized mathematical reasoning benchmarks.

For classification experiments, we additionally use three datasets from the SUPERGLUE benchmark (Wang et al., 2019): SST-2, CB, and MultiRC. For CB, we retain the complete training set of 250 labeled examples. For SST-2 and MultiRC, we randomly subsample 3K examples each for training.

E.3 TRAINING DETAILS

Following the configuration in **?**, we employ a learning rate of 2×10^{-5} with a cosine decay scheduler. The learning rate is linearly warmed up from 0 to 2×10^{-5} during the first 3% of training steps and subsequently decays to 0 following a cosine schedule. We fix the maximum sequence length to 512 tokens. Unless otherwise stated, all experiments on MATHINSTRUCT are trained for the equivalent of 1K gradient update steps. To enable larger effective batch sizes, we use gradient accumulation with an accumulation factor of 8.

For parameter-efficient fine-tuning, we adopt LoRA (**?**) with rank 128, scaling parameter $\alpha = 512$, and a dropout rate of 0.05. On PHI models, LoRA adapters are applied to all attention projection matrices (QKV) as well as the two feed-forward layers. On ZEPHYR, we apply LoRA to all attention projections (QKVO). All experiments are conducted on 4 NVIDIA A40 GPUs, and each configuration is repeated three times to account for variance in training.

E.4 EVALUATION DATASETS AND METRICS

Following **?**, we evaluate our models on a diverse suite of mathematical reasoning benchmarks spanning both in-domain and out-of-domain distributions.

In-domain benchmarks. The in-domain evaluation covers three widely used datasets: GSM8K (**?**), MATH (**?**), and NUMGLUE (Mishra et al., 2022). GSM8K focuses on grade-school arithmetic word problems, MATH contains high-school competition-style problems across 29 mathematical domains, and NUMGLUE extends natural language understanding tasks with quantitative reasoning components.

Out-of-domain benchmarks. To test generalization beyond the training distribution, we additionally include SVAMP (**?**), the MATHEMATICS dataset (**?**), and SIMULEQ (**?**). These datasets emphasize robustness across algebraic manipulations, probability and statistics, number theory, and systems of equations, while also incorporating instances requiring multi-step logical reasoning and commonsense knowledge.

E.5 EVALUATION SETUP

1NN Classifier Setup Let $\mathcal{X}$ and $\mathcal{Y}$ denote the source and target datasets, respectively, with potentially differing class distributions, and let $\mathcal{P} \subseteq \mathcal{X}$ be a candidate representative set for the target dataset $\mathcal{Y}$. The quality of $\mathcal{P}$ is assessed using a 1-nearest neighbour (1-NN) classifier parameterized

by the elements of $\mathcal{P}$. Each instance $y \in Y$ is assigned the label of its nearest prototype in $\mathcal{P}$, where the ground-truth class labels of the elements in $\mathcal{P}$ are assumed to be available during this evaluation. The resulting classification accuracy serves as the evaluation metric for comparing prototype selection algorithms.

LLM-Finetuning: All questions are posed in an open-ended format. We adopt the standard *exact match* metric, where a prediction is considered correct only if it exactly matches the gold reference solution. Evaluation is conducted under the 0-shot setting with a maximum decoding context length of 2048 tokens. We use the Program-of-Thought (PoT) prompting strategy as the default, and fall back to Chain-of-Thought (CoT) prompting when PoT is not applicable, following **?**.

F Additional Experimental Results

F.1 Additional Results on UniPROT-batch

Experiments on bs-256 prototype selection

We experiment is UniPROT-PB batch size of 256 for selection instead of source wise prototype selection. We finetune Phi-3 for 2048 steps, selection batch size of 256, with prototype percentage as 0.25, resulting in a effective batch of 32. Table 5 shows that UniPROTcontinues to be effective even in full batch setting. Moreover Figure 7 indicates that CoLM's validation perplexity suffers as batch size increases

Here, we report additional results on UniPROT-batch on MathInstruct Dataset.

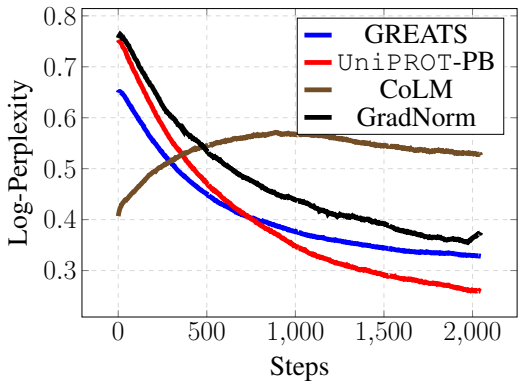

Figure 7: Validation perplexity when bs=256 and prototype ratio is 25%.

Table 4: Comparison of UniPROT-PB performance for bs256, following Yue et al. (2023) on MathInstruct Dataset.

Method	Avg	In-domain			Out-of-domain		
		GSM8K	MATH	NumGLUE	SVAMP	Mathematics	SimulEq
COLM (Nguyen et al., 2025)		74.13	36.7	63.14	86.5	36.40	62.3
GREATS (Wang et al., 2024)		78.62	37.9	63.9	85.5	36.9	61.9
UniPROT-PB (Ours)		78.2	37.6	66.03	84.9	37.7	63.6

Experiments on full-batch prototype selection

For this variant, we experiment is full-batch selection instead of source wise prototype selection. We finetune Phi-3 for 2048 steps, selection batch size of 128, with prototype percentage as 0.5, resulting in a effective batch of 64. Table 5 shows that UniPROTcontinues to be effective even in full batch setting.

F.2 Additional Results on Phi-2

Here, we report additional results on Phi-2 on MathInstruct Dataset.

F.3 Additional Results on Zephyr-3B

Here, we report additional results on Zephyr-3B on MathInstruct Dataset.

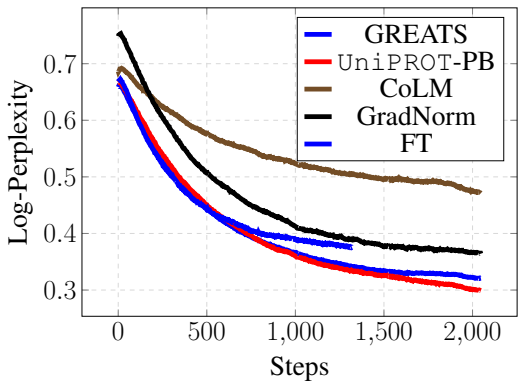

Figure 8: Validation perplexity when bs=128 and subset ratio 50% and prototype selection is batch wise.

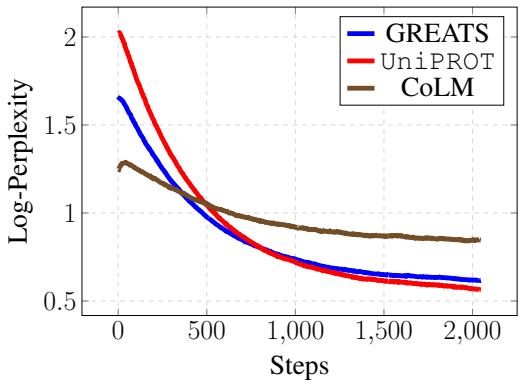

Figure 9: Validation perplexity when bs=128 and prototype ratio is 25% on ZEPHYR-3B.

Table 5: Comparison of UniPROT-batch performance across in-domain and out-of-domain datasets, following Yue et al. (2023) on MATHINSTRUCT Dataset. **Green Bold** = best, *Yellow Italic* = second best.

Method	Avg	In-domain			Out-of-domain		
		GSM8K	MATH	NumGLUE	SVAMP	Mathematics	SimulEq
COLM (Nguyen et al., 2025)		75.36	34.05	64.1	85.3	37.40	63.6
GREATS (Wang et al., 2024)		79.07	33.58	64.4	85	38	62.06
UniPROT-PB (Ours)		77.8	33.95	65.9	85.7	34.1	68.28

Table 6: Comparison of PHI-2 performance across in-domain and out-of-domain datasets, following Yue et al. (2023) on MATHINSTRUCT Dataset. **Green Bold** = best, *Yellow Italic* = second best.

Method	Avg	In-domain			Out-of-domain		
		GSM8K	MATH	NumGLUE	SVAMP	Mathematics	SimulEq
COLM (Nguyen et al., 2025)		61.03	26.67	52.65	60.45	21.04	37
GREATS (Wang et al., 2024)		61.92	27.21	52.4	62	20.8	38.03
UniPROT-PS (Ours)		62.4	27.63	53.97	64.72	20.3	38.91

Table 7: Comparison of ZEPHYR-3B performance across in-domain and out-of-domain datasets, following Yue et al. (2023) on MATHINSTRUCT Dataset. **Green Bold** = best, *Yellow Italic* = second best

Method	Avg	In-domain			Out-of-domain		
		GSM8K	MATH	NumGLUE	SVAMP	Mathematics	SimulEq
COLM (Nguyen et al., 2025)		50.6	21.42	40.15	55.8	16.7	22.17
GREATS (Wang et al., 2024)		52.8	19.01	40.8	54.6	17.1	21.98
UniPROT-PS (Ours)		54.89	19.6	41.3	54.1	15.5	24.7

G ABLATION STUDY

Effect of number of selected prototypes. We finetune ZEPYR-3B on MATHINSTRUCT for 2048 steps while varying the selection ratio $r \in 50\%, 25\%, 12.5\%$. Figure 6 reports the validation perplexity. We observe that UniPROT remains consistently stable across all ratios, showing only a minor increase at 12.5%. In contrast, COLM degrades noticeably as r decreases, while GREATS shows a smaller but still measurable rise. Overall, UniPROT exhibits the lowest sensitivity to prototype budget, indicating stronger robustness.

Number of optimal transport iterations. We vary the number of iterations in the partial optimal transport solver while finetuning PHI-3 on MATHINSTRUCT for 512 steps, and report downstream accuracy on GSM8K and NumGLUE (Table 8). With only 5 iterations, accuracy drops noticeably on both tasks. At 20 iterations, performance matches that of 100 iterations, indicating that the solver converges quickly and that a small iteration budget is sufficient to reach the best downstream accuracy.

Effect of regularization on downstream performance. We ablate the entropic regularization coefficient in the partial optimal transport objective by finetuning PHI-3 on MATHINSTRUCT and evaluating downstream on GSM8K, NumGLUE and Svamp. We finetune for 128 steps and 20 OT iterations.(Table 9). With $\lambda = 0.1$, performance is consistently lower, while reducing to $\lambda = 0.01$ yields clear gains on both tasks. This trend aligns with prior observations (Cuturi, 2013), where smaller regularization improves transport fidelity and leads to better downstream accuracy.

H BROADER IMPACT

This work develops a principled framework for prototype selection that aims to improve fairness and robustness in settings with distributional imbalance. By explicitly enforcing uniform weighting, UniPROTcan reduce systematic under-representation of minority classes, which has positive impli-

Table 8: Variation with number of iterations.

Method	GSM-8k	NumGlue
20 iters	78.2	64.8
100 iters	78.3	64.8

Table 9: Variation with number of regularisation strength.

Method	GSM-8k	NumGlue	Svamp
0.1	47.76	37.5	52.9
0.05	48.36	36.1	54
0.01	49.4	36.7	54.5

cations for equitable model performance across demographic or domain groups. At the same time, more efficient subset selection methods could also be leveraged to accelerate training of harmful or biased systems if applied without safeguards. We believe that open discussion of both the benefits and limitations of prototype selection methods is important to ensure they are deployed responsibly, and that continued transparency in this line of work will help maximize positive societal impact.