
Supplementary Material: Bandit Guided Submodular Curriculum for Adaptive Subset Selection

Contents

Appendices	21
A Organization of the Appendix	22
B Notation Summary	22
C Implementation Details	22
C.1 Details about model architectures used	22
C.2 Details on submodular functions implementation	23
C.3 Gradient Computation	23
C.4 Evaluation metrics	24
D Experimental Setup Details	25
D.1 Software and Hardware	25
D.2 Language Model Experiments	25
D.3 Vision Model Experiments	25
D.4 Baseline Training Details	26
D.5 Comparison between DINO and Gradient-Based Features for Submodular Selection	27
D.6 Fisher Information Matrix	27
E Additional Experiments	28
E.1 Sensitivity of Validation Dataset Configuration	28
E.2 Effect of Submodular Functions Individually and RANDOM Selection over Arms	28
E.3 Additional Experiments on Vision datasets	30
E.4 Additional Experiments on Large Language Models	31
F Details of Submodular Function used in all our training settings	32
F.1 Diversity based Submodular Function	32
F.2 Representative based Submodular Function	32
G Additional Related Work	33
G.1 Online Submodular Maximization	33
G.2 Pruning Mechanisms	34
H Main Theoretical Results	35
H.1 Proof for permutation invariance of Expected Marginal Gain	35
H.2 Theorem: Capacity-Controlled Risk Convergence Theorem	35
H.3 No-Regret Bounds under Constant $\lambda(\cdot)$	35
H.4 Regret bounds in the case of growing with time exploration dampening function	39
I Broader Impact	42

Supplementary Material: Bandit Guided Submodular Curriculum for Adaptive Subset Selection

A Organization of the Appendix

The appendix is organized as follows. Section I provides a summary of the impact of our work. Section B provides a summary of the notation used throughout the paper. Section H presents our main theoretical results. Section D outlines the experimental setup and implementation details for both vision and language model tasks. Section G discusses additional related work. Section F describes the various submodular functions employed in our experiments.

B Notation Summary

Table 4: Table of Notations

Topic	Notation	Explanation
Data (sub)Sets Indices	$\mathcal{D}_{\text{train}}$	Entire Training Set consisting of n instances
	\mathcal{D}_{val}	Entire Validation Set consisting of m instances
	\mathbf{z}_i	i -th training instance in a batch
	\mathcal{B}_t	Denotes the full sized t -th step train minibatch : $\{\mathbf{x}_p\}_{p=1}^{ \mathcal{B}_t }$
	$\mathcal{B}_t^{\text{val}}$	Denotes the full sized t -th step validation minibatch
	$\mathcal{B}_t^{(<i)}$	Denotes the t -th step minibatch being constructed upto \mathbf{x}_{i-1} i.e. $\{\mathbf{x}_p\}_{p=1}^{i-1}$
Parameters	$\mathcal{S}_{(a_t)}^{\text{opt}}$	Optimal subset obtained when submodular function $f^{(a_t)}$ is applied
	θ^*	Optimal model parameter (vector)
	θ_t	Model parameter at t^{th} step
	θ_{t+1}	Model parameter at $(t + 1)^{\text{th}}$ step
Loss Function	ℓ	Strongly convex instance-wise loss function
	\mathcal{L}_t	Total loss over mini-batch
	$\mathcal{U}_t(\mathcal{B}_t; z_t^{\text{val}})$	Utility metric capturing validation loss drop for a particular validation data point
	$\mathcal{U}_t(\mathcal{B}_t; \mathcal{B}_t^{\text{val}})$	Aggregated utility metric over validation set
Hyperparams	$\lambda(t)$	(<i>Exploration Dampening</i>) modulates the inertia of exploration
	ϑ	Reward function
	Ξ_t	Exploration-exploitation threshold
	$\mathcal{F}_{\text{sub}}^{\text{div}}, \mathcal{F}_{\text{sub}}^{\text{repr}}$	Diversity/representative function subsets
	$\zeta \sim \text{Uniform}(0, 1)$	Random sample for trade-off
	$\pi(t)$	(<i>Exploration Sharpness</i>) controls the curvature of the annealing schedule rule

C Implementation Details

C.1 Details about model architectures used

Vision Model Architecture Details:

The ResNet18 model [15] architecture begins with a *basic block*, which is composed of two main sections. The first section consists of a convolution layer followed by a batch normalization layer, and then a ReLU activation function. The second section similarly comprises a convolution layer followed by batch normalization. This entire *basic block* is repeated twice for each of the four layers in the network. These layers progress with input dimensions of [64, 128, 256, 512] to form the complete ResNet18 architecture.

Language Model Architecture Details

The LLaMA-2-7B model is a decoder-only transformer comprising approximately 7 billion parameters. It includes 32 transformer layers, each built with pre-normalization using RMSNorm and employing the SwiGLU activation function. The self-attention mechanism uses multi-head causal attention with 32 heads and a hidden dimensionality of 4096. Rotary positional embeddings (RoPE) are applied to the query and key vectors within each attention head. The model begins with a learned token embedding layer and concludes with a tied output projection layer to predict the next token. Mistral-7B-v0.3 is architecturally similar to LLaMA-2-7B, also featuring 32 transformer layers and a 4096-dimensional hidden state, but introduces several efficiency-focused modifications. It uses grouped-query attention (GQA) with 8 query groups across 32 heads, improving inference throughput. Furthermore, Mistral replaces full causal attention with sliding-window attention to handle long contexts more efficiently. As with LLaMA, it utilizes RoPE for positional encoding and SwiGLU activations. These optimizations maintain strong modeling performance while enabling greater scalability in both training and inference settings.

C.2 Details on submodular functions implementation

We provide detailed formulations of the specific submodular functions employed as arms in our experiments in Section F. From an implementation standpoint, each submodular arm operates on a similarity kernel computed over the set of instances within a given batch. This kernel, typically represented as a symmetric positive semi-definite matrix, encodes pairwise affinities between samples based on their embedding representations (e.g., cosine similarity or RBF kernel). Once the similarity structure is established, any submodular function can be instantiated over this ground set—such as facility location, log-determinant, or graph-cut functions—depending on the desired coverage, diversity, or representativeness property being optimized.

To operationalize this, we leverage the `Submodlib` library⁴, an open-source framework maintained by the Decile organization⁵, which provides efficient and modular implementations of a wide family of submodular functions. The library supports both dense and sparse similarity representations and includes greedy as well as lazy-greedy optimization routines, enabling scalable computation even for large batch sizes.

C.3 Gradient Computation

Computing full-model gradients in modern deep networks is computationally prohibitive due to the extremely high dimensionality of parameter spaces—often exceeding billions of parameters for large vision or language models. Moreover, for the purpose of subset selection, what is typically required is not the full parameter gradient but an informative proxy that captures the *relative contribution* of individual samples to the model’s training dynamics.

Following this motivation, we adopt *partial-gradient approximations* that preserve discriminative signal while substantially reducing computational cost. Specifically, for vision models, we compute gradients only with respect to the *last linear classification layer*, as in [3]. This choice leverages the empirical observation that gradients in earlier layers are highly correlated and redundant, and that last-layer gradients retain sufficient information to distinguish hard, redundant, or noisy samples based on their contribution to the decision boundary.

For large language models (LLMs), computing full backpropagation across all transformer layers is infeasible. We therefore restrict gradient computation to *Low-Rank Adaptation (LoRA)* adapter parameters (rank 128), following the setup of [52]. This approach not only reduces memory and compute overhead by several orders of magnitude but also captures localized curvature information relevant to the fine-tuning or instruction-following objective. Since LoRA adapters are trained in the low-dimensional subspace most sensitive to task adaptation, their gradients provide a faithful and low-noise estimate of per-sample learning signals.

Importantly, both approximations maintain *gradient informativeness* under the assumption that the selected subspace (last layer or adapter) spans the most discriminative directions of parameter updates. Prior empirical evidence (see [3, 52]) shows that subset selection, influence estimation, and sample

⁴<https://submodlib.readthedocs.io/en/latest/>

⁵<https://decile.org/>

reweighting methods computed in these reduced spaces closely match those computed with full gradients. In our experiments, we verify that this approximation incurs negligible performance degradation while providing up to $30\times$ faster per-batch computation. Thus, the proposed gradient computation scheme achieves a favorable balance between computational efficiency and fidelity of learning signal for submodular subset selection.

Warm-starting Data Selection. A common challenge in data subset selection methods lies in the instability of early-stage gradients. During the initial epochs of training, model parameters are far from any local minimum, and per-sample gradients tend to be highly noisy and uninformative. Consequently, performing subset selection too early can result in biased or suboptimal subsets that fail to represent the underlying data distribution or learning dynamics. To mitigate this issue, for the image experiments, we conduct a *warm-start* strategy, wherein the model is first trained for a small number of epochs on the full dataset before invoking any subset selection procedure.

Concretely, for each algorithm considered in this paper (i.e., `ONLINESUBMOD`, `GRADMATCH`, `GRADMATCHPB`, `CRAIG`, `CRAIGPB`, and `GLISTER`), we include a warm-start variant. Let T denote the total number of training epochs and k the subset size. We define two quantities: T_f (the number of full-training epochs prior to subset selection) and T_s (the number of epochs during which subset selection is active). We set these in proportion as $T_s = \kappa T$, $T_f = \frac{T_s k}{n}$, where n is the total number of training samples and $\kappa \in (0, 1]$ is the fraction of total training epochs used for subset selection. This parametrization ensures that the effective compute budget remains comparable across methods, while allowing early-stage training to stabilize the model representation before adaptive data selection begins.

Empirically, we observe that performing a few epochs of full-data warm-up (T_f) consistently improves convergence stability and downstream accuracy across all subset selection algorithms. The warm-start phase enables the gradient space to form a meaningful geometry, allowing the submodular or gradient-based selection objectives to more accurately identify informative and diverse samples. In contrast, starting selection from random initialization often leads to premature overfitting or unstable subset composition due to noisy or poorly conditioned gradients.

Setting T_f too large, however, diminishes the benefit of subset selection, as the model effectively performs full training with minimal adaptive sampling. In this limit, the behavior approaches that of the full-batch baseline with early stopping, which we include as a control setting in our experiments. Thus, the warm-start scheme provides a principled balance between computational efficiency and representational stability—retaining the benefits of subset-based training while ensuring robust and smooth convergence.

C.4 Evaluation metrics

Image Classification: For image classification experiments, we report the standard *test accuracy* as the primary performance metric, measured as the proportion of correctly classified samples on the held-out validation or test split. This metric provides a direct and interpretable indicator of the model’s generalization performance under different subset selection strategies.

In addition to accuracy, we evaluate the *computational efficiency* of our method by comparing the total training time required to reach convergence across different selection policies. To ensure a fair comparison, all other hyperparameters—including optimizer configuration, learning rate schedule, batch size, and data augmentations—are held fixed across runs. The only varying factor is the subset selection mechanism applied at each training step.

We define the *speedup* metric with respect to the baseline model trained using full-batch selection (i.e., without any submodular or adaptive sampling). Formally, if T_{full} denotes the wall-clock training time for the full-batch model and T_{sub} denotes the time under our submodular selection strategy, the speedup is given by $\text{Speedup} = \frac{T_{\text{full}}}{T_{\text{sub}}}$. A higher speedup thus indicates a more efficient training regime, achieved without sacrificing downstream accuracy. In practice, we observe consistent gains in training efficiency—typically in the range of $3\times$ – $8\times$ —depending on the dataset and the choice of submodular objective, confirming that adaptive selection substantially reduces redundant gradient computations while maintaining comparable predictive performance.

D Experimental Setup Details

D.1 Software and Hardware

Vision Experiments All experiments were conducted using Python 3.10.13 and PyTorch 2.1.2. Our proposed methods, `ONLINESUBMOD` and `ONLINESUBMOD-Batch`, along with their corresponding ablations, were trained on NVIDIA RTX A6000 GPUs (48 GB). Baseline methods, including RHO-LOSS and BOSS, were also trained using the same GPU configuration to ensure comparability.

For reference, a typical training run of our ResNet18-based model on an RTX A6000 consists of 300 epochs, with each epoch averaging approximately one minute (excluding certain baselines). Model checkpointing is employed to retain only the best-performing model based on validation accuracy, as well as the final model. Running multiple training jobs concurrently on the same GPU incurs only a slight overhead in training time due to resource contention.

D.2 Language Model Experiments

We experiment for the LLM finetuning setup using a `RANDOM` subset of 9 datasets from MMLU, and on TydiQA. We choose Sociology, Policy, History, Anatomy, ML, Ethics, Genetics, High School Biology, High School Chemistry. All language model experiments, including both our proposed methods and the baselines, were conducted using 8 NVIDIA H100 GPUs. Additionally, Weights & Biases (WandB) ⁶wandb was used to manage and monitor all experiments. For all experiments we take batch size of 16, initial learning rate of $2e-5$ using adam optimizer with default state, finetuned on 10% of LESS[55] version of OpenWebText.

Additional Experiment Results: For MMLU we also showcase additional experiments on LLaMa2-7b and Mistral-7b for TydiQA, later in the appendix.

Mathematical definitions of the submodular objectives used as arms are provided in Appendix F. For this experiment, each arm is a mutual information variant of a classical submodular function, designed to maximize $I_f(X; Q) = f(X) + f(Q) - f(X \cup Q)$, where X is the candidate training set, Q is the validation set, and f is a base submodular function (Facility-Location, Graph-Cut, or Log-Determinant).

We use mutual information forms to ensure the selected subset is explicitly conditioned on the current validation set, making the acquisition process adaptive to the downstream task. Features for X and Q are derived either from Sentence-BERT embeddings or from gradient vectors, with the latter shown to yield better alignment with task-specific error signals and improved selection performance.

D.3 Vision Model Experiments

The experimental setup was configured to evaluate the proposed method on several datasets, including CIFAR-10, CIFAR-100, Tiny-ImageNet-200, and SVHN. The data module used a batch size of 128, with four workers for data loading. The model architecture employed was ResNet18 [15], and the training followed a curriculum-based mode, progressively utilizing 10%, 30%, and 50% of the training data. The optimizer used was SGD with a learning rate of 0.05, momentum of 0.9, weight decay of 0.0005, and Nesterov momentum enabled.

For all our settings (across different baselines and dataset), we consider ResNet18 [15] as our primary model with the following architecture and training details:

In our training setup, we employed batch-wise Nesterov accelerated gradient descent with a batch size of 128. The optimization configuration included a learning rate of 0.05 and a momentum of 0.9, alongside a cosine-annealing scheduler.

Across all dataset comparisons, we set the submodular function budget β to 10%, 30%, and 50% of the entire batch size.

Dataset Specifics. We conduct experiments across a range of standard vision benchmarks. For the **MNIST** dataset, we use 60,000 training instances, 10,000 test instances, and 10,000 validation instances, with training proceeding until full convergence, typically around **200 epochs**. On CIFAR-

⁶<https://wandb.ai/site/>

10, we use 50,000 training instances, 10,000 test instances, and 10,000 validation instances, with models trained for up to **300 epochs**. For **CIFAR-100**, we similarly use 50,000 training examples spread across 100 classes (500 per class), and a validation set of 10,000 examples (100 per class). The **SVHN** dataset comprises 73,257 training images across 10 classes with variable class frequencies, and a validation set of 26,032 images distributed proportionally. Finally, for **TINYIMAGENET**, we use 100,000 training images across 200 classes (500 per class), and a validation set of 10,000 images (50 per class), covering the same label space as the training data.

D.4 Baseline Training Details

We compare our method **ONLINESUBMOD** with several state of the art baselines for our experiments:

MAX-LOSS [27]: Within each training batch, the loss is computed for every example. A fixed fraction (e.g., top-K%) of samples with the highest per-example loss is selected for gradient computation and model update. This assumes that high-loss samples are currently mis-predicted and could contribute the most to updating the decision boundary.

GRADNORM [17]: The l_2 norm of each example’s per-sample gradient i.e. $\|\nabla_{\theta}\mathcal{L}(z; \theta)\|_2$ is computed, and a subset with the highest norms is selected for each batch. This prioritizes examples inducing the largest parameter updates under the current model, helping direct learning toward sensitive or uncertain regions.

RHO-LOSS [30]: Each example’s *reducible loss* is estimated as the difference between the current model’s loss and its *irreducible loss*, the latter approximated by a small auxiliary model trained on held-out clean data. Examples with high reducible loss are selected, as they are considered learnable but not yet learned, making them useful for continued training. For LLM experiments, we use `LLaMa3-7b-instruct` as our auxiliary model. For image experiments, we begin by training an irreducible model on the specific task for 100 epochs. Subsequently, we precompute the irreducible losses for the training set, which are required for the target model training. During the target model training phase, we train the model for 300 epochs across the CIFAR-10, CIFAR-100, SVHN, and TINYIMAGENET datasets, using subset ratios of 0.1, 0.3, and 0.5. We employ the ResNet-18 architecture for both the irreducible model and the target model. For training, we use the SGD optimizer with Nesterov accelerated gradient descent, a batch size of 128, and the following configuration: a learning rate of 0.05, momentum of 0.9, and a cosine-annealing scheduler. One observation we made is that RHO Loss converges to reasonably good accuracy within a few epochs, but further training does not significantly improve performance, and it fails to reach the accuracy levels of other state-of-the-art (SOTA) baselines.

SBERT [40]: In this case, each training and validation example is encoded into a sentence embedding using a pre-trained SBERT model. Cosine similarity is computed between training examples and the validation set, and those with the highest average similarity are selected. This favors examples that align semantically with the validation distribution.

For **GREATS** [52], each example’s impact on the loss is approximated using a first-order Taylor expansion of the objective. For model parameters θ and a batch $\{x_1, \dots, x_n\}$, gradients $\nabla\mathcal{L}(x_i; \theta)$ are used to estimate loss reduction. A greedy selection strategy then chooses the subset expected to most decrease validation loss under this approximation.

For fair comparison against our model we considered the configuration where subset selection happens at every epoch for all the 3 baselines with a *lazy* optimizer. Due to our multi-class image classification setup we utilise *CrossEntropy* loss for our model training.

BOSS [1]: For BOSS, to select the subset, we first initialized a model by training it using the full dataset. With the help of the training dynamics obtained from the initialized model, we calculated the difficulty score for each sample that is used to select the subset. We evaluated the selected subset keeping the subset fixed and using it to train a new RANDOM initialized model. For the difficulty score, we experimented using the EL2N score because it can be efficiently computed early on during training. We trained the model for 300 epochs across the CIFAR-10, CIFAR-100, SVHN and TINYIMAGENET datasets, using subset ratios of 0.1, 0.3, and 0.5. We employed ResNet18 model using SGD with a learning rate of 0.1, and momentum of 0.9 with a batch size of 128.

D.5 Comparison between DINO and Gradient-Based Features for Submodular Selection

To evaluate how closely our feature representations must align with the downstream objective, we compared two ways of representing each training item when optimising submodular acquisition functions (and their mutual-information variants):

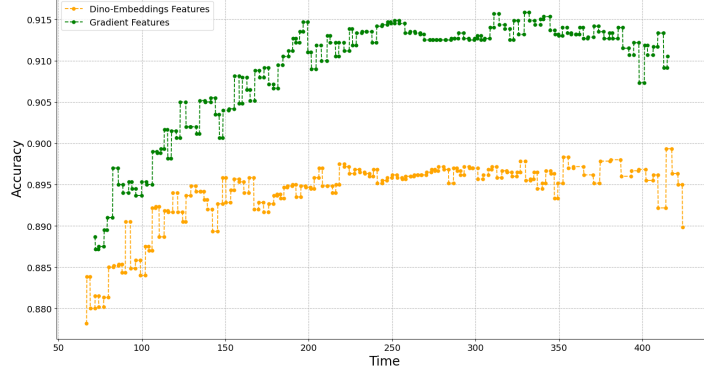


Figure 7: Comparison of Fashion-MNIST with DINO-embeddings, and with Gradient Features for submodular optimization

1. **DINO embeddings.** We obtain a fixed d -dimensional feature vector for every image by running it through a frozen DINO vision transformer, exactly as one would use a CLIP encoder. These representations are task-agnostic and remain static throughout training.
2. **Gradient-based features.** At every training step we compute the gradient of the scalar loss with respect to the parameters of the final layer. We average these per-example gradients within the mini-batch to form a single vector⁷. For mutual-information objectives, validation gradients serve as the query features.

Figure 7 shows that gradient features yield substantially higher test accuracy on FASHION-MNIST across all subset sizes: they encode task-specific error signals that guide the submodular optimiser toward examples most useful for loss reduction, whereas DINO embeddings capture only generic visual similarity. Hence, directly leveraging gradients as features is the more effective choice for data subset selection in this setting.

D.6 Fisher Information Matrix

Fisher Information Matrix Approximation An alternative and potentially more informative approach to approximating the Hessian is through the use of the Fisher Information Matrix (FIM) [10]. The FIM provides insights into the curvature of the loss landscape and can serve as a useful surrogate for the Hessian. While the exact computation of the FIM requires calculating an expectation, which can be computationally intensive, it can be efficiently approximated using an exponential moving average of the outer product of the gradients from the validation data points.

Let $\Omega_i := g(\mathbf{z}_i, \theta_t)g(\mathbf{z}_i, \theta_t)^\top$ denote the outer product of the gradient for the i th data point in the current batch \mathcal{B}_t . The approximate FIM $\hat{\mathcal{H}}_{\mathcal{B}_t}^{(t)}$ at time step t for the current mini-batch \mathcal{B}_t can be computed recursively as

$$\hat{\mathcal{H}}_{\mathcal{B}_t}^{(t)} = \begin{cases} \frac{1}{|\mathcal{B}_0|} \sum_{\mathbf{z}_i \in \mathcal{B}_0} \Omega_i, & \text{if } t = 0 \\ (1 - \alpha) \hat{\mathcal{H}}_{\mathcal{B}_{t-1}}^{(t-1)} + \alpha \frac{1}{|\mathcal{B}_t|} \sum_{\mathbf{z}_i \in \mathcal{B}_t} \Omega_i, & \text{else} \end{cases} \quad (9)$$

where $\hat{\mathcal{H}}_{\mathcal{B}_{t-1}}^{(t-1)}$ is the approximate FIM at the previous time step $t - 1$ for the mini-batch \mathcal{B}_{t-1} , and $\alpha \in (0, 1]$ is the smoothing parameter for the exponential moving average. This recursive formulation provides a computationally efficient approach to approximating the Hessian, particularly in high-dimensional settings where direct Hessian computation is prohibitively expensive.

⁷Using the batch-wise average was consistently superior to concatenating per-example gradients, and last-layer gradients are sufficient while keeping the computation inexpensive.

E Additional Experiments

E.1 Sensitivity of Validation Dataset Configuration

To better understand the influence of validation data composition on model performance, we investigate how sensitive the algorithm is to different validation set configurations. In particular, we examine what occurs during the early stages of training when the validation dataset includes harder samples that the model has not yet adequately learnt. This analysis provides a deeper perspective on how the distribution and difficulty of validation examples can affect optimization dynamics and generalization, thereby strengthening the empirical validity of our findings.

Specifically, we aim to understand the following questions:

- **Q1:** To what extent is the algorithm’s performance sensitive to the configuration and composition of the validation dataset?
- **Q2:** How does the presence of harder, yet-unlearned samples in the validation set during early training stages affect convergence and generalization?

To better understand this issue, we conducted a controlled experiment on CIFAR-100 (300 epochs) where we varied the hardness of the validation dataset. Hardness was measured via gradient norm where the gradient is calculated w.r.t model parameter at that time step. In accordance with other literature, a crude way to approximate difficulty of a sample is to check if the gradient norm is high. (higher gradient norm \sim harder example). We compared four validation subset configurations:

Validation Subset	10%	20%	30%
Easiest	72.3	74.4	76.03
EasyHard	73.1	74.5	76.4
HardEasy	72.29	74.6	76.2
Hardest	71.31	74.3	75.9

Table 5: Final test accuracies under different validation subset configurations.

- **Easiest:** Lowest gradient norms
- **EasyHard:** Easy samples early, hard samples later
- **HardEasy:** Hard samples early, easy samples later
- **Hardest:** Highest gradient norms

Each configuration was evaluated at validation subset sizes of 10%, 20%, and 30%.

Observations:

Validation sets composed of the most difficult examples tend to yield lower performance, particularly when smaller subsets are used. This decline likely stems from noisy or overly pessimistic reward signals during the early stages of training. In contrast, mixed validation configurations such as **EasyHard** and **HardEasy** generally perform best, indicating that a balanced distribution of sample difficulty across training can enhance robustness. These findings suggest that further exploring how validation sample difficulty and ordering interact especially through the lens of curriculum learning could be a promising direction for future work. Importantly, even validation sets containing difficult samples early in training do not lead to instability or model collapse.

E.2 Effect of Submodular Functions Individually and RANDOM Selection over Arms

To assess the contribution of the multi-armed bandit formulation in our framework, we perform ablation experiments on CIFAR-100 (10% subset, 300 epochs) under two simplified settings: (a) using a single, fixed submodular arm throughout training (i.e., no bandit-driven adaptation), and (b)RANDOMly selecting an arm at each round (i.e., no explore–exploit balancing). These ablations isolate the effect of static versus dynamic subset selection policies on training efficiency and generalization.

We compare various submodular selection strategies that define the reward structure of the curriculum. *Representative* functions (e.g., **GraphCut**, **FacilityLocation**) promote coverage and ensure the selected subset reflects the global data distribution, while *Diversity*-oriented functions (e.g., **DisparitySum**, **LogDeterminant**) encourage maximal dissimilarity among chosen samples. These functions capture different inductive biases: representation versus decorrelation.

Selection Strategy	Accuracy (%)
DisparitySum (Div., Static)	68.6
FacilityLocation (Rep., Static)	72.0
LogDeterminant (Div., Static)	71.1
GraphCut (Rep., Static)	72.6
Random arm per round	72.0
ONLINESUBMOD (ours)	73.6

Table 6: Performance comparison of individual and RANDOM arm selection strategies on CIFAR-100. ONLINESUBMOD adaptively balances diversity and representativeness over training epochs.

Our proposed **ONLINESUBMOD** method dynamically alternates between these functions through an adaptive explore–exploit policy governed by the bandit controller. This dynamic weighting enables the model—whether a **ResNet-18** backbone or a small **LLM fine-tuning setup**—to exploit high-yield submodular arms while continually exploring others that may improve validation loss or perplexity.

Observations:

Different submodular functions show complementary but limited strengths. Coverage-based methods such as *GraphCut* and *FacilityLocation* converge quickly early in training, while diversity-based ones like *DisparitySum* and *LogDet* encourage better generalization but can become unstable when applied uniformly. Random arm selection gives reasonable results, suggesting that diversity matters, but it lacks feedback to adapt to validation performance. In contrast, the proposed **ONLINESUBMOD** approach adjusts arm selection based on past rewards, maintaining a stable balance between exploration and exploitation. This supports our main hypothesis that adaptive, reward-driven selection leads to more robust and generalizable outcomes than static or random strategies.

E.3 Additional Experiments on Vision datasets

Table 7 summarizes batchwise data selection results across multiple vision datasets. Across all budgets and datasets, **ONLINESUBMOD** consistently achieves the highest test accuracy while maintaining competitive or lower training time compared to prior methods. Notably, it surpasses strong baselines such as GRADMATCH, MILO, and RHO-LOSS, particularly at low data budgets (10%–30%), indicating superior sample efficiency and adaptivity under constrained training regimes. The improvement is most pronounced on CIFAR100 and TINYIMAGENET, where the model benefits from dynamic online selection over diverse feature manifolds. In contrast, static coreset-based methods (e.g., CRAIG, GLISTER) exhibit slower convergence and lower performance as the budget increases.

Here, **red** highlights the best result and **blue** denotes the second-best result for each setting. Overall, these results confirm that **ONLINESUBMOD** provides a strong trade-off between accuracy and computational efficiency across datasets of varying complexity.

Table 7: (Batchwise) Data Selection Results on Vision Datasets

Dataset	Selection Strategy	Test accuracy (%)			Training time (hrs)			
		Budget(%)	10%	30%	50%	10%	30%	50%
CIFAR10	FULL (skyline for test accuracy)		95.09	95.09	95.09	1.73	1.73	1.73
	RANDOM (skyline for training time)		77.49	89.62	91.85	0.29	0.75	0.85
	CRAIG		90.07	92.4	93.12	0.26	0.62	1.54
	GLISTER		91.15	92.18	92.65	0.38	1.05	1.34
	GRADMATCH		92.27	93.28	93.15	0.42	0.95	1.21
	MILO		92.25	93.21	94.16	0.34	0.85	0.89
	RHO-LOSS		90.16	91.54	94.03	0.76	1.13	1.54
	BOSS		91.64	93.04	93.8	0.36	0.94	1.18
	ONLINESUBMOD (ours)		92.44	93.75	94.18	0.32	0.87	0.83
CIFAR100	FULL (skyline for test accuracy)		76.8	76.8	76.8	1.52	1.52	1.52
	RANDOM (skyline for training time)		35.03	61.93	64.67	0.15	0.42	0.78
	CRAIG		67.25	72.38	73.12	0.31	0.62	1.12
	GLISTER		64.27	72.36	74.62	0.26	0.57	1.3
	GRADMATCH		68.34	74.63	72.36	0.22	0.48	1.22
	MILO		72.36	74.66	75.60	0.15	0.44	0.82
	RHO-LOSS		71.37	74.82	75.74	0.53	0.86	1.46
	BOSS		71.73	73.77	75.41	0.27	0.53	0.85
	ONLINESUBMOD (ours)		73.67	75.46	75.78	0.165	0.47	0.82
SVHN	FULL (skyline for test accuracy)		96.49	96.49	96.49	6.436	6.436	6.436
	RANDOM (skyline for training time)		93.47	95.31	95.84	0.6383	1.90	3.19
	CRAIG		95.27	96.15	96.40	0.934	2.332	4.17
	GLISTER		95.52	95.69	96.42	0.83	2.42	4.26
	GRADMATCH		95.64	96.4	96.42	0.789	2.398	4.19
	MILO		95.62	96.36	96.41	0.69	2.09	3.25
	RHO-LOSS		94.64	94.27	94.85	1.08	2.56	3.94
	BOSS		94.31	95.75	96.01	0.76	2.39	3.56
	ONLINESUBMOD (ours)		95.68	96.38	96.46	0.68	2.12	3.28
TINYIMAGENET	FULL (skyline for test accuracy)		64.36	64.36	64.36	15.4	15.4	15.4
	RANDOM (skyline for training time)		19.61	35.68	43.84	1.82	4.92	6.12
	CRAIG		52.42	55.56	61.48	3.27	6.46	9.23
	GLISTER		51.54	56.37	62.15	2.84	5.93	9.47
	GRADMATCH		52.63	58.19	61.93	2.63	5.94	7.24
	MILO		53.24	59.36	62.28	1.81	4.97	6.16
	RHO-LOSS		54.46	59.78	62.15	3.16	6.38	7.94
	BOSS		52.63	60.17	62.13	2.85	5.47	7.26
	ONLINESUBMOD (ours)		55.3	60.74	62.58	1.84	5.16	6.14

E.4 Additional Experiments on Large Language Models

We evaluate the evolution of test perplexity during pretraining on the MMLU benchmark using the LLAMA-2-7B model under different **online batch selection strategies**. Each method is trained under identical hyperparameter and compute budgets to ensure fair comparison.

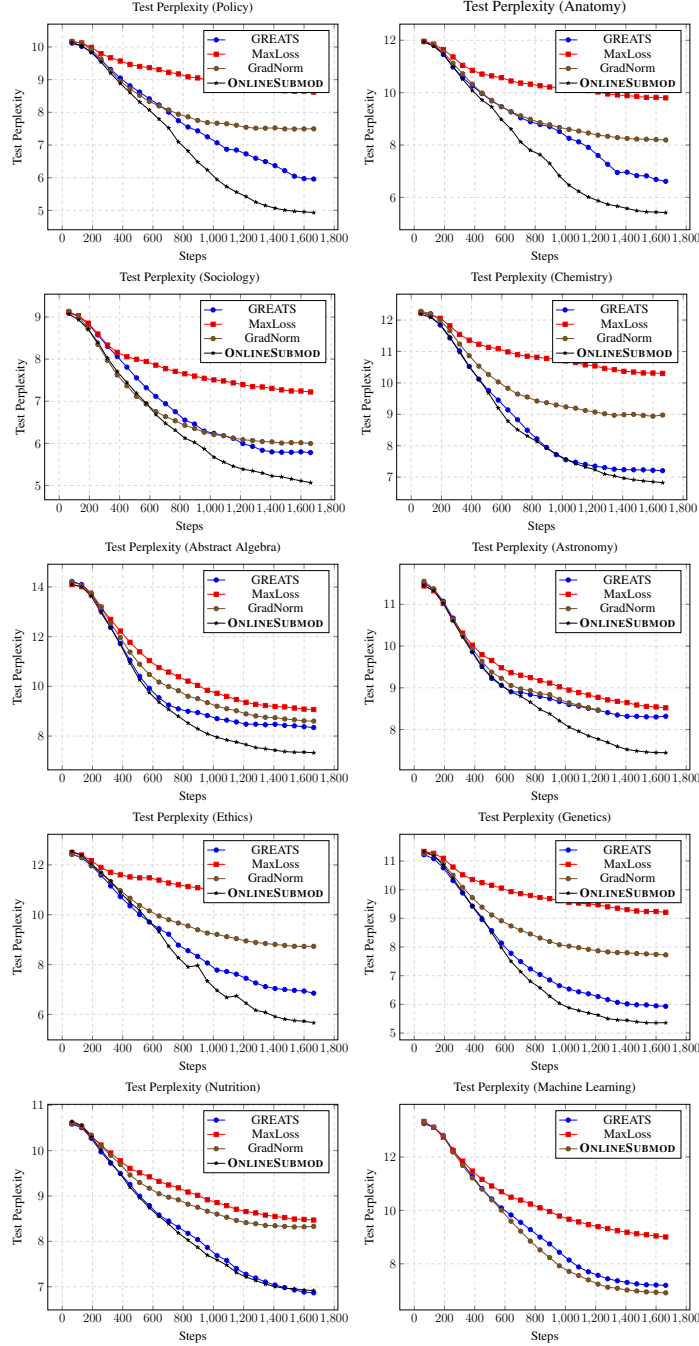


Figure 8: **Test perplexity dynamics** on LLAMA-2-7B during training with various **online batch selection strategies** on MMLU. ONLINESUBMOD significantly outperforms baselines.

F Details of Submodular Function used in all our training settings

We describe here the submodular functions we broadly utilised for all our experiments.

F.1 Diversity based Submodular Function

Here we share the details on the diversity based submodular functions we used for our training purposes.

Definition 1. Log-determinant Function is a diversity-based submodular function. It is non-monotone in nature. Let \mathbf{L} denote a positive semidefinite kernel matrix and \mathbf{L}_S denote the subset of rows and columns indexed by set S . Log-determinant function f is specified as:

$$f(S) = \log \det(\mathbf{L}_S) \quad (10)$$

The log-det function models diversity and is closely related to a determinantal point process.

Definition 2. Disparity Sum Function characterizes diversity by considering the sum of distances between every pair of points in a subset S . For any two points $i, j \in S$, let d_{ij} denote the distance between them.

$$f(S) = \sum_{i,j \in S} d_{ij} \quad (11)$$

The aim is to select a subset S such that $f(S)$ is maximized.

Definition 3. Disparity Min Function characterizes diversity by considering the minimum distance between any two non-similar points in a subset S .

$$f(S) = \min_{i,j \in S, i \neq j} d_{ij} \quad (12)$$

The aim is to select a subset S such that $f(S)$ is maximized.

F.2 Representative based Submodular Function

Here we share the details on the representative based submodular functions we used for our training purposes.

Definition 4. Facility Location Function characterizes the representativeness in the dataset by considering the minimum distance between any two non-similar points in a subset S .

$$f(S) = \sum_{i \in V} \max_{j \in S} d_{ij} \quad (13)$$

The aim is to select a subset S such that $f(S)$ is maximized.

Definition 5. Graph Cut Function characterizes representativeness by using the parameter λ which governs the tradeoff between representation and diversity. When λ becomes large, graph cut function also tries to model diversity in the subset. S .

$$f(S) = \sum_{i \in V, j \in S} d_{ij} - \lambda \sum_{i,j \in S} d_{ij} \quad (14)$$

The aim is to select a subset S such that $f(S)$ is maximized.

Submodular Mutual Information We first provide a definition of Submodular Mutual Information:

$$\mathcal{I}_f(A; B) = f(A) + f(B) - f(A \cup B)$$

Definition 6. Log-Determinant Mutual Information Function is an instantiation of a submodular mutual information function using a `LogDeterminantFunction`. Let $S_{A,B}$ be the cross-similarity matrix between the items in sets A and B . Also, denote $S_{AB} = S_{A \cup B}$. We construct a similarity matrix S^η (on a base matrix S) such that the cross-similarity between A and Q is multiplied by η (i.e., $S_{A,Q}^\eta = \eta S_{A,Q}$) to control the trade-off between query relevance and diversity. Higher values of η ensure greater query-relevance while lower values favor diversity. Using a similarity matrix defined above and with $f(A) = \log \det(S_A^\eta)$, we have:

$$I_f(A; Q) = \log \det(S_A) - \log \det(S_A - \eta^2 S_{A,Q} S_Q^{-1} S_{A,Q}^T) \quad (15)$$

Definition 7 (Generalized Submodular Mutual Information). Let Ω be a ground set and $\mathcal{V} \subseteq \Omega$ be a domain of interest. Let $f : 2^\Omega \rightarrow \mathbb{R}_{\geq 0}$ be a restricted submodular function, i.e., submodular when restricted to subsets of \mathcal{V} . A Submodular Mutual Information (SMI) function defined via such a function f is called a Generalized Submodular Mutual Information (GMI) function.

A notable instance of GMI is the Concave Over Modular (COM) function [21], defined for subsets $\mathcal{A} \subseteq \mathcal{V}$ and $\mathcal{Q} \subseteq \mathcal{V}'$ as:

$$I_{f_\eta}(\mathcal{A}; \mathcal{Q}) = \eta \sum_{i \in \mathcal{A}} \psi \left(\sum_{j \in \mathcal{Q}} s_{ij} \right) + \sum_{j \in \mathcal{Q}} \psi \left(\sum_{i \in \mathcal{A}} s_{ij} \right),$$

where $\eta \in \mathbb{R}_{\geq 0}$ controls the trade-off between query-relevance and diversity, $\psi : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ is a concave function, $S = [s_{ij}]$ is a kernel similarity matrix such that $s_{ij} = \mathbb{1}(i = j)$ for $i, j \in \mathcal{V}$ or $i, j \in \mathcal{V}'$.

Definition 8 (Facility Location Mutual Information (FLIMI)). Let $I_f(\mathcal{A}; \mathcal{Q})$ denote a Submodular Mutual Information (SMI) function. An instantiation of SMI using the `FacilityLocationFunction` is known as the Facility Location Mutual Information (FLIMI) function.

Formally, given subsets $\mathcal{A} \subseteq \mathcal{V}$ and $\mathcal{Q} \subseteq \mathcal{V}'$, FLIMI is defined as:

$$I_f(\mathcal{A}; \mathcal{Q}) = \sum_{i \in \mathcal{V}} \min \left(\max_{j \in \mathcal{A}} s_{ij}, \eta \max_{j \in \mathcal{Q}} s_{ij} \right),$$

where: $\eta \in \mathbb{R}_{\geq 0}$ is a relevance-diversity trade-off parameter, s_{ij} denotes similarity between elements i and j in the kernel similarity matrix S , \mathcal{V} is the candidate set and \mathcal{V}' is the query set domain.

Definition 9 (Graph Cut Mutual Information (GCMI)). Let $I_f(\mathcal{A}; \mathcal{Q})$ denote a Submodular Mutual Information (SMI) function. An instantiation of SMI using the `GraphCutFunction` is called the Graph Cut Mutual Information (GCMI) function.

Formally, for subsets $\mathcal{A} \subseteq \mathcal{V}$ and $\mathcal{Q} \subseteq \mathcal{V}'$, GCMI is defined as:

$$I_f(\mathcal{A}; \mathcal{Q}) = 2\lambda \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{Q}} s_{ij},$$

where $\lambda \in \mathbb{R}_{\geq 0}$ controls the scale of mutual information, s_{ij} denotes the similarity between elements i and j in the kernel similarity matrix S , \mathcal{V} is the candidate set and \mathcal{V}' is the query set domain.

G Additional Related Work

G.1 Online Submodular Maximization

A growing body of research has advanced our understanding of online submodular maximization under diverse feedback models and constraint classes. A notable contribution is the recent work on [13], which introduces a principled framework leveraging first-order regret bounds from online linear optimization to derive improved guarantees in submodular settings. At each round t , the algorithm selects a feasible set $S_t \in \mathcal{C} \subseteq 2^V$, where \mathcal{C} encodes combinatorial constraints such as matroids or cardinality bounds, and observes an adversarially chosen submodular function f_t . For monotone submodular functions under matroid constraints, the method achieves a $(1 - c/e - \epsilon)$ -approximate regret bound of $\mathcal{O}(kT \log(n/k))$, improving on earlier results by Streeter and Golovin [45], and Golovin and Krause [11], even in the absence of curvature (i.e., $c = 1$). For non-monotone unconstrained submodular functions, a novel algorithm based on Blackwell approachability achieves a $1/2$ -regret of $\mathcal{O}(n\sqrt{T})$, extending Roughgarden and Wang [41].

These developments complement recent advances in bandit and semi-bandit feedback models, including those by Hassani et al. [14] and [47], who analyze online submodular optimization in stochastic and adversarial environments, obtaining nearly minimax optimal regret bounds. Related efforts have also explored limited feedback settings with structure-aware exploration strategies (e.g., combinatorial Thompson sampling or optimism-based approaches), enhancing sample efficiency in large-scale decision spaces.

In the full-information setting, the online continuous greedy algorithm of Bian et al. [5] offers near-optimal $(1 - 1/e)$ -regret for monotone submodular functions under matroid constraints, while extensions [56] tackle delayed feedback. Meanwhile, online versions of lazy greedy [31] and distributionally robust submodular maximization [44] have enabled scalable implementations in real-world domains such as streaming recommendation and dataset summarization.

Although submodular maximization is NP-hard in general, approximation algorithms often yield near-optimal performance in practice across diverse applications [45, 11]. These include influence maximization, budget-constrained recommendation, and online resource allocation, all of which benefit from the expressive yet structured nature of submodular objectives. As such, the aforementioned theoretical advances not only deepen our algorithmic understanding but also broaden the applicability of online submodular optimization frameworks to practical domains involving limited feedback, combinatorial constraints, and dynamic inputs.

G.2 Pruning Mechanisms

Several recent works have explored data pruning and subset selection for efficient training, including D2PRUNING [28], INFOMAX [48], and CCS [58]. While these methods offer valuable insights into coreset selection and dataset reduction, they predominantly operate in static, full-dataset settings, in contrast to our dynamic, batch-level framework.

INFOMAX formulates an objective that can be interpreted as a reformulation of the Graph Cut function, which is monotone submodular, and leverages similarity kernels such as DINO embeddings (for image tasks) or gradient-based features. This aligns conceptually with our approach, where Graph Cut is explicitly implemented as a bandit arm. However, INFOMAX selects samples over the entire dataset in a static manner, whereas our framework is modular and dynamic: an InfoMax-like objective can be treated as a bandit arm and applied in batch-level pruning during training. This flexibility enables more scalable deployment in real-world training pipelines where adaptive, online selection is crucial.

D2PRUNING frames data subset selection as a subgraph pruning task over the dataset, representing data points as nodes in a similarity graph. Selection is performed using message passing algorithms, which can be computationally intensive and challenging to scale to large datasets. Like INFOMAX, D2PRUNING operates at a static, dataset-wide level, making it less suitable for dynamic batch-level pruning.

Similarly, CCS addresses static selection by optimizing for both coverage and diversity. A key contribution of CCS is its theoretical characterization of the pruning budget, identifying thresholds beyond which accuracy degradation becomes catastrophic. While our method does not primarily operate at the full-dataset level, we note that analogous effects could, in principle, occur in batch-level selection; investigating such phenomena is a potential avenue for future work.

In summary, although INFOMAX, D2PRUNING, and CCS provide important foundations for data pruning and coreset strategies, they are largely static and dataset-wide in nature. Our approach extends these ideas to a dynamic, scalable setting by leveraging a bandit-driven curriculum where batch-level pruning decisions are guided by validation performance. Moreover, the modularity of our framework allows for seamless integration of alternative reward signals, such as forgetting scores or other criteria discussed in prior works, enabling flexible and adaptive training in large-scale environments.

H Main Theoretical Results

H.1 Proof for permutation invariance of Expected Marginal Gain

Lemma 1 (Permutation Invariance of Expected Marginal Gain). *Let Π denote the set of all permutations over the elements of $\mathcal{B}_t^{(<i)}$. Then the expected marginal gain $\mathbb{E}_{\mathbf{z}_i \in \mathcal{B}_t^{(<i)}} [\Delta \mathcal{U}_t(\mathbf{z}_i \mid \mathcal{B}_t^{(<i)}, \mathbf{z}_t^{\text{val}})]$ is invariant under any permutation $\pi \in \Pi$, i.e.,*

$$\mathbb{E}_{\mathbf{z}_i \in \mathcal{B}_t^{(<i)}} [\Delta \mathcal{U}_t(\mathbf{z}_i \mid \mathcal{B}_t^{(<i)}, \mathbf{z}_t^{\text{val}})] = \mathbb{E}_{\mathbf{z}_i \in \pi(\mathcal{B}_t^{(<i)})} [\Delta \mathcal{U}_t(\mathbf{z}_i \mid \pi(\mathcal{B}_t^{(<i)}), \mathbf{z}_t^{\text{val}})].$$

Proof. Let $S = \mathcal{B}_t^{(<i)}$, with $|S| = n$, and let $\mathbf{z}_{\text{val}} = \mathbf{z}_t^{\text{val}}$. We denote $\mathbf{g}_i := \mathbf{g}_{\theta_t}(\mathbf{z}_i)$, $\mathbf{g}_v := \mathbf{g}_{\theta_t}(\mathbf{z}_{\text{val}})$, and $\mathcal{H}_v := \mathcal{H}_{\mathbf{z}_{\text{val}}}(\theta_t)$.

Then the expected marginal gain as per Eq 4 is given by:

$$\frac{1}{n} \sum_{\mathbf{z}_i \in S} \left[\eta_t \mathbf{g}_i \cdot \mathbf{g}_v - \eta_t^2 \mathbf{g}_i^\top \mathcal{H}_v \left(\frac{1}{n} \sum_{\mathbf{z} \in S} \mathbf{g}_z \right) \right].$$

Let $\bar{\mathbf{g}} := \frac{1}{n} \sum_{\mathbf{z} \in S} \mathbf{g}_z$. Then:

$$= \eta_t \bar{\mathbf{g}} \cdot \mathbf{g}_v - \eta_t^2 \left(\frac{1}{n} \sum_{\mathbf{z}_i \in S} \mathbf{g}_i^\top \mathcal{H}_v \bar{\mathbf{g}} \right) = \eta_t \bar{\mathbf{g}} \cdot \mathbf{g}_v - \eta_t^2 \bar{\mathbf{g}}^\top \mathcal{H}_v \bar{\mathbf{g}}.$$

This expression depends only on the multiset S , not the order of its elements. Therefore, for any permutation $\pi(S)$, the same value holds:

$$\mathbb{E}_{\mathbf{z}_i \in \pi(S)} [\Delta \mathcal{U}_t(\mathbf{z}_i \mid \pi(S), \mathbf{z}_{\text{val}})] = \eta_t \bar{\mathbf{g}} \cdot \mathbf{g}_v - \eta_t^2 \bar{\mathbf{g}}^\top \mathcal{H}_v \bar{\mathbf{g}}.$$

Hence,

$$\mathbb{E}_{\mathbf{z}_i \in S} [\Delta \mathcal{U}_t(\mathbf{z}_i \mid S, \mathbf{z}_{\text{val}})] = \mathbb{E}_{\mathbf{z}_i \in \pi(S)} [\Delta \mathcal{U}_t(\mathbf{z}_i \mid \pi(S), \mathbf{z}_{\text{val}})],$$

which proves the claim. \square

H.2 Theorem: Capacity-Controlled Risk Convergence Theorem

Theorem 2 (Capacity-Controlled Risk Convergence). *[[12)] **Theorem 16.3**] Let \mathcal{M}_Θ be a neural network with d parameters belonging to the parameter space Θ with an objective to minimize the empirical risk over the training data, $\mathcal{D} = \{(\mathcal{X}_i, \mathcal{Y}_i)\}_{i=1}^n$ where $\mathcal{X}_i \in \mathbb{R}^m$ and \mathcal{Y} are almost surely bounded and where $\mathcal{Y}_i = \vartheta(\mathbf{z}_i) \sim \mathcal{N}(\mu_{\mathbf{z}_i}, \sigma_{\mathbf{z}_i})$ where $\vartheta : \mathbb{R}^m \rightarrow \mathbb{R}$ and \mathbb{P} denotes the data distribution. Then for d large enough, we have the following, for any $\epsilon > 0$.*

$$\mathbb{E}_{\mathcal{D}} \int_{\mathbf{z}} \|\mathcal{M}_\Theta(\mathbf{z}) - \mathbb{E}[\vartheta(\mathbf{z})]\|^2 d\mathbb{P}(\mathbf{z}) \leq \epsilon \sqrt{\frac{\ln(d)}{d}} \quad (16)$$

The above theorem follows from [39] and [12] and is useful to prove further bounds in our case as below.

H.3 No-Regret Bounds under Constant $\lambda(\cdot)$

We first state here the main theorem under the following assumptions as stated in our main text:

Let $\tau_{(a)}^R(t)$ denote the number of times the a -th submodular function $\mathbf{f}^{(a)}$ is chosen in the first $t - 1$ steps by the uniform branch of the algorithm.

Assumption (a) (Constant Fractional Exploration Dampening): The exploration dampening parameter $\lambda(t)$ is time-invariant $\lambda(t) = \epsilon$ where $\epsilon \in (0, 1)$.

Assumption (b) (Optimality Gap): There exists an optimality gap ϱ such that for every suboptimal arm $a_t \in \mathcal{A} \setminus \{a^*\} : 0 \leq \varrho \leq \Delta_{(a_t)}(\mathcal{B}_t)$.

Assumption (c) (Fractional Exploration Sharpness): The exploration sharpness parameter $\pi(t)$ is a bounded quantity $\pi(t) \in (0, 1)$.

Assumption (d) (Utility Metric Approximation): The utility metric $\mathcal{U}_t(\cdot, \cdot)$ satisfies the approximation bound as per Theorem 2 (Appendix) with constants $\mathfrak{C}_{(a)}$ for each arm $a \in \mathcal{A}$ and let n_a be a specific constant associated with arm a such that Theorem 2 (Appendix) holds true.

Theorem 1 (Regret Guarantees). Under Assumptions **a - d**, for all $t > t_0$, with probability at least

$$1 - \mathcal{K} \exp \left(-\frac{3(t-2)(1+(1-\pi)\epsilon)}{28\mathcal{K}(2-\pi)} \right),$$

the expected instantaneous regret incurred by the arm selection policy satisfies

$$\begin{aligned} \mathbb{E}[\text{Regret}_t] &:= \mathbb{E}_{\mathcal{B}_t} \mathbb{E}_{\hat{a}_t \in \mathcal{A}} \mathbb{E}_{\boldsymbol{\vartheta}} [\boldsymbol{\vartheta}(a_t^* | \mathcal{B}_t) - \boldsymbol{\vartheta}(\hat{a}_t | \mathcal{B}_t)] \\ &= O\left(\frac{1}{t}\right) + O\left(\frac{\mathcal{K}^{3/2}(\max_a \mathfrak{C}_{(a)} + \mathfrak{C}_*)}{\varrho} \sqrt{\frac{\log t}{t}}\right), \end{aligned} \quad (8)$$

where \mathfrak{C}_* is the approximation constant corresponding to the optimal arm a^* .

Proof.

$$\begin{aligned} &\mathbb{E}_{\mathcal{B}_t} \mathbb{E}_{a_t \in [\mathcal{K}]} \mathbb{E}_{\boldsymbol{\vartheta}} [\boldsymbol{\vartheta}(f^{(a_t^*)} | \mathcal{B}_t) - \boldsymbol{\vartheta}(f^{(a_t)} | \mathcal{B}_t)] \\ &= \mathbb{E}_{\mathcal{B}_t} \left[\mu_{(*)}(\mathcal{B}_t) - \mathbb{E}_{a \in [\mathcal{K}]} \mathbb{E}_{\boldsymbol{\vartheta}} [\boldsymbol{\vartheta}(f^{(a_t)} | \mathcal{B}_t)] \right] \quad \text{where } \mu_{(\bullet)} = \mathbb{E}_{\boldsymbol{\vartheta}}[\boldsymbol{\vartheta}(\bullet)] \\ &= \mathbb{E}_{\mathcal{B}_t} \left[\mu_{(*)}(\mathcal{B}_t) - \sum_{j=1}^{\mathcal{K}} \mu_a(\mathcal{B}_t) \mathbb{P}(f^{(a_t)} = f^{(a)} | \mathcal{B}_t) \right] \\ &= \mathbb{E}_{\mathcal{B}_t} \sum_a \Delta_a(\mathcal{B}_t) \mathbb{P}(f^{(a_t)} = f^{(a)} | \mathcal{B}_t) \\ &= \sum_a \mathbb{E}_{\mathcal{B}_t} \Delta_a(\mathcal{B}_t) \mathbb{P}(f^{(a_t)} = f^{(a)} | \mathcal{B}_t) \end{aligned}$$

$$\begin{aligned} &\mathbb{E}_{\mathcal{B}_t} \Delta_a(\mathcal{B}_t) \mathbb{P}(f^{(a_t)} = f^{(a)} | \mathcal{B}_t) \\ &\leq \mathbb{E}_{\mathcal{B}_t} \Delta_a(\mathcal{B}_t) \left[\Xi_t/\mathcal{K} + \mathbb{P}(\mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t)}} \geq \mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t^*)}}) \right] \end{aligned}$$

Let $\mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t)}}$ indicates the trained neural network in accordance to [39] for action $f^{(a_t)}$. By Markov's inequality

$$\begin{aligned} \mathbb{P}(\mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t)}} \geq \mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t^*)}}) &\leq \mathbb{P}\left(\mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t)}} \geq \mu_{(a)}(\mathcal{B}_t) + \Delta_a(\mathcal{B}_t)/2\right) + \\ &\quad \mathbb{P}\left(\mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t^*)}} \leq \mu_{(*)}(\mathcal{B}_t) - \Delta_a(\mathcal{B}_t)/2\right) \\ &= \int_{\mathbb{1}\{\mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t)}} \geq \mu_{(a)}(\mathcal{B}_t) + \Delta_a(\mathcal{B}_t)/2\}} \partial \mathbb{P}_a + \int_{\mathbb{1}\{\mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t^*)}} \leq \mu_{(*)}(\mathcal{B}_t) - \Delta_a(\mathcal{B}_t)/2\}} \partial \mathbb{P}_* \quad (17) \\ &\leq \int_{\mathbb{1}\{|\mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t)}} - \mu_{(a)}(\mathcal{B}_t)| \geq \Delta_{(a)}(\mathcal{B}_t)/2\}} \partial \mathbb{P}_a + \int_{\mathbb{1}\{|\mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t^*)}} - \mu_{(*)}(\mathcal{B}_t)| \geq \Delta_{(a)}(\mathcal{B}_t)/2\}} \partial \mathbb{P}_* \\ &\leq \int 4 \frac{|\mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t)}} - \mu_{(a)}(\mathcal{B}_t)|^2}{\Delta_{(a)}(\mathcal{B}_t)^2} \partial \mathbb{P}_a + \int 4 \frac{|\mathcal{M}_{\boldsymbol{\Theta}, f^{(a_t^*)}} - \mu_{(*)}(\mathcal{B}_t)|^2}{\Delta_{(a)}(\mathcal{B}_t)^2} \partial \mathbb{P}_* \end{aligned}$$

Based on Proposition 1 we have $\tau_{(a)}^R(t) \geq \frac{t-2}{2\mathcal{K}(2-\pi)}(1+(1-\pi)\epsilon)$ for all $a \in \mathcal{A}$. Let $\mathfrak{C}_{(a)}$ indicate the constant from Theorem 2 and let n_a be the minimal training data size. We choose $t_0 > e^{(2\mathcal{K} \max\{e, \max_a n_a\})}$. Since the $x \rightarrow \sqrt{\frac{\ln(x)}{x}}$ is monotone decreasing for $x > e$, the above expression is further bounded by

$$\begin{aligned}
&\leq \mathbb{E}_{\mathcal{B}_t} \Delta_{(a)}(\mathcal{B}_t) \frac{\epsilon_t}{\mathcal{K}} + \frac{4}{\varrho} \mathfrak{C}_{(a)} \sqrt{\frac{\ln(\tau_{(a)}^R(t))}{\tau_{(a)}^R(t)}} + \frac{4}{\varrho} \mathfrak{C}_* \sqrt{\frac{\ln(\tau_*^R(t))}{\tau_*^R(t)}} \\
&\leq \mathbb{E}_{\mathcal{B}_t} \Delta_{(a)}(\mathcal{B}_t) \frac{\epsilon_t}{\mathcal{K}} + \frac{4}{\varrho} \mathfrak{C}_{(a)} \sqrt{\frac{\ln(\tau_i^R(t))}{\tau_i^R(t)}} + \frac{4}{\varrho} \mathfrak{C}_* \sqrt{\frac{\ln(\tau_*^R(t))}{\tau_*^R(t)}} \\
&\leq \frac{\mathbb{E}_{\mathcal{B}_t} \Delta_{(a)}(\mathcal{B}_t)}{t\mathcal{K}} + \frac{4}{\varrho} \left[\mathfrak{C}_{(a)} + \mathfrak{C}_* \right] \sqrt{\frac{\ln\left(\frac{t-2}{2\mathcal{K}(2-\pi)}(1+(1-\pi)\epsilon)\right)}{\frac{t-2}{2\mathcal{K}(2-\pi)}(1+(1-\pi)\epsilon)}}
\end{aligned} \tag{18}$$

Thus we have the following:

$$\begin{aligned}
&\sum_{\mathbf{f}^{(a_t)}} \mathbb{E}_{\mathcal{B}_t} \Delta_a(\mathcal{B}_t) \mathbb{P}(\mathbf{f}^{(a_t)} = \mathbf{f}^{(a)} \mid \mathcal{B}_t) \\
&\leq \frac{\max_{\mathbf{f}^{(a_t)} \in \mathcal{F}_{\text{sub}}} \mathbb{E}_{\mathcal{B}_t} \Delta_a(\mathcal{B}_t)}{t} + \\
&\quad \mathcal{K}^{3/2} \sqrt{2(2-\pi)} \frac{4}{\varrho} \left[\max_a \mathfrak{C}_{(a)} + \mathfrak{C}_* \right] \\
&\quad \sqrt{\frac{\ln((t-2)(1+\epsilon-\pi\epsilon)) - \ln(2\mathcal{K}(2-\pi))}{(t-2)(1+\epsilon-\pi\epsilon)}}
\end{aligned} \tag{19}$$

To showcase the lower bound, we have for a -th arm not optimal that,

$$\mathbb{E}_{\mathcal{B}_t} \Delta_a(\mathcal{B}_t) \mathbb{P}(\mathbf{f}^{(a_t)} = \mathbf{f}^{(a)} \mid \mathcal{B}_t) \geq \mathbb{E}_{\mathcal{B}_t} \Delta_a(\mathcal{B}_t) \frac{\Xi_t}{\mathcal{K}} \geq \mathbb{E}_{\mathcal{B}_t} \varrho \frac{\Xi_t}{\mathcal{K}} \geq \frac{\varrho}{t\mathcal{K}} \tag{20}$$

□

Lemma 2 (Bound on Uniform Arm Selection Frequency). Since $\tau_{(a)}^R(t)$ denotes the number of times the a -th submodular function $\mathbf{f}^{(a)}$ is chosen in the first $t-1$ steps by the uniform branch of the algorithm, we have the following:

$$\begin{aligned}
&\mathbb{P}\left(\bigcap_{a=1}^{\mathcal{K}} \{\tau_{(a)}^R(t) \geq \frac{t-2}{2\mathcal{K}(2-\pi)}(1+(1-\pi)\epsilon)\}\right) \\
&\geq 1 - \mathcal{K} \exp\left(-\frac{3(t-2)(1+(1-\pi)\epsilon)}{28\mathcal{K}(2-\pi)}\right)
\end{aligned}$$

Proof.

$$\begin{aligned}
&\mathbb{E}(\tau_{(a)}^R(t)) = \sum_{r=1}^{t-1} \mathbb{P}(\zeta < \Xi_r \cap \mathbf{f}^{(a_t)} = \mathbf{f}^{(a)}) \\
&= \sum_{r=1}^{t-1} \mathbb{P}(\zeta < \Xi_r) \mathbb{P}(\mathbf{f}^{(a_t)} = \mathbf{f}^{(a)}) = \sum_{r=1}^{t-1} \frac{\Xi_r}{\mathcal{K}} = \frac{1}{\mathcal{K}} \sum_{r=1}^{t-1} \frac{r}{(r+\lambda(r))^\pi} \\
&\geq \frac{1}{\mathcal{K}} \int_{x=1}^{x=t-1} \frac{x}{(x+\lambda(x))^\pi} \partial x \geq \frac{t-2}{\mathcal{K}(2-\pi)}(1+(1-\pi)\epsilon)
\end{aligned} \tag{21}$$

where, the last inequality is based on *Proposition 1*. We define the variance of $\tau_{(a)}^R(t)$ as $\sigma(\tau_{(a)}^R(t))$ and the corresponding upperbound as $\mathcal{Z}(\sigma(t))$

$$\sigma(\tau_{(a)}^R(t)) = \sum_{r=1}^{t-1} \frac{\Xi_r}{\mathcal{K}} (1 - \frac{\Xi_r}{\mathcal{K}}) \leq \frac{1}{\mathcal{K}} \sum_{r=1}^{t-1} \Xi_r = \frac{1}{\mathcal{K}} \sum_{r=1}^{t-1} \frac{r}{(r+\lambda(r))^\pi} = \mathcal{Z}(\sigma(t))$$

Using Bernstein's inequality

$$\begin{aligned}
\mathbb{P}\left(\tau_{(a)}^R(t) \leq \frac{\mathcal{Z}(\sigma(t))}{2}\right) &= \mathbb{P}\left(\tau_{(a)}^R(t) - \mathcal{Z}(\sigma(t)) \leq -\frac{\mathcal{Z}(\sigma(t))}{2}\right) \\
&\leq \exp\left(\frac{\frac{-\mathcal{Z}(\sigma(t))^2}{8}}{\sigma(\tau_{(a)}^R(t)) + \frac{1}{3}\frac{\mathcal{Z}(\sigma(t))}{2}}\right) \leq \exp\left(\frac{\frac{-\mathcal{Z}(\sigma(t))^2}{8}}{\mathcal{Z}(\sigma(t)) + \frac{1}{3}\frac{\mathcal{Z}(\sigma(t))}{2}}\right) \\
&\leq \exp\left(-\frac{3\mathcal{Z}(\sigma(t))}{28}\right) \leq \exp\left(-\frac{3(t-2)(1+(1-\pi)\epsilon)}{28\mathcal{K}(2-\pi)}\right)
\end{aligned}$$

By union bound method

$$\begin{aligned}
&\mathbb{P}\left(\bigcup_{a=1}^{\mathcal{K}} \{\tau_{(a)}^R(t) \leq \frac{t-2}{2\mathcal{K}(2-\pi)}(1+(1-\pi)\epsilon)\}\right) \\
&\leq \mathcal{K}\mathbb{P}\left(\tau_{(1)}^R(t) \leq \frac{t-2}{2\mathcal{K}(2-\pi)}(1+(1-\pi)\epsilon)\right) \leq \mathcal{K}\exp\left(-\frac{3(t-2)(1+(1-\pi)\epsilon)}{28\mathcal{K}(2-\pi)}\right)
\end{aligned}$$

Therefore

$$\mathbb{P}\left(\bigcap_{a=1}^{\mathcal{K}} \{\tau_{(a)}^R(t) \geq \frac{t-2}{2\mathcal{K}(2-\pi)}(1+(1-\pi)\epsilon)\}\right) \geq 1 - \mathcal{K}\exp\left(-\frac{3(t-2)(1+(1-\pi)\epsilon)}{28\mathcal{K}(2-\pi)}\right)$$

□

Proposition 1 (Integral Lower Bound (Constant λ)). *Let $\lambda(t) = \epsilon$ with $0 < \epsilon < 1$, and $0 < \pi < 1$. Then, for*

$$I_t = \int_{x=1}^{t-1} \frac{x}{(x + \lambda(x))^\pi} dx,$$

we have

$$I_t \geq \int_{x=1}^{t-1} \frac{x}{(x + \epsilon)^\pi} dx \geq \frac{t-2}{2-\pi} (1 + (1-\pi)\epsilon).$$

Proof.

$$\begin{aligned}
\int_{x=1}^{x^{t-1}} \frac{x}{(x + \lambda(x))^\pi} dx &= \int_{x=1}^{x=t-1} \frac{x}{(x + \epsilon)^\pi} dx && \text{(Substitute } \lambda(x) = \epsilon \text{)} \\
&= \frac{[t - (1 - \epsilon)]^{2-\pi}}{2 - \pi} - \frac{(1 + \epsilon)^{2-\pi}}{2 - \pi} \\
&\quad - \epsilon \cdot \frac{[t - (1 - \epsilon)]^{1-\pi}}{1 - \pi} + \epsilon \cdot \frac{(1 + \epsilon)^{1-\pi}}{1 - \pi} \\
&= [t - (1 - \epsilon)]^{1-\pi} \left(\frac{(1 - \pi)(t - 1) - \epsilon}{(1 - \pi)(2 - \pi)} \right) \\
&\quad - [1 + \epsilon]^{1-\pi} \left(\frac{1 - \pi - \epsilon}{(1 - \pi)(2 - \pi)} \right) \\
&= [m + \epsilon]^{1-\pi} \left(\frac{(1 - \pi)m - \epsilon}{(1 - \pi)(2 - \pi)} \right) \\
&\quad - [1 + \epsilon]^{1-\pi} \left(\frac{(1 - \pi) - \epsilon}{(1 - \pi)(2 - \pi)} \right) && \text{(Let } m = t - 1 \text{)} \\
&= [m + \epsilon]^k \left(\frac{km - \epsilon}{k(k + 1)} \right) - [1 + \epsilon]^k \left(\frac{k - \epsilon}{k(k + 1)} \right) && \text{(Let } k = 1 - \pi \text{)} \\
&= \frac{1}{k(k + 1)} ([m + \epsilon]^k (km - \epsilon) + (\epsilon - k)[1 + \epsilon]^k) \\
&\geq \frac{1}{k(k + 1)} ([1 + \epsilon]^k (km - \epsilon) + (\epsilon - k)[1 + \epsilon]^k) \\
&\quad \text{(Since } m + \epsilon \geq 1 + \epsilon \text{)} \\
&= \frac{1}{k(k + 1)} [1 + \epsilon]^k (km - \epsilon + \epsilon - k) \\
&= \frac{1}{k(k + 1)} [1 + \epsilon]^k (km - k) \\
&= \frac{1}{k + 1} [1 + \epsilon]^k (m - 1) \\
&\geq \frac{1}{k + 1} [1 + k\epsilon] (m - 1) && \text{(Using } (1 + \epsilon)^k \geq 1 + k\epsilon \text{ for small } \epsilon \text{)} \\
&= \frac{1}{2 - \pi} [1 + (1 - \pi)\epsilon] (t - 2) && \text{(Substitute } k = 1 - \pi, m = t - 1 \text{)} \\
&= \frac{t - 2}{2 - \pi} (1 + (1 - \pi)\epsilon)
\end{aligned}$$

(22)

□

The above integral computation is used in the main paper for our proofs.

H.4 Regret bounds in the case of growing with time exploration dampening function

Proposition 2 (Integral Lower Bound (Exponential Growing λ)). *Let $\lambda(t) = 1 - e^{-it}$ be a time-growing function with rate $i > 0$, and let $0 < \pi < 1$. Then, for*

$$I_t = \int_{x=1}^{t-1} \frac{x}{(x + \lambda(x))^\pi} dx,$$

the following lower bound holds:

$$I_t \geq \int_{x=1}^{t-1} \frac{x}{(x + 1 - e^{-ix})^\pi} dx \geq \left(\frac{1}{2i} \left[\ln(2e^{i(t-1)} - 1) - \ln(2e^i - 1) \right] \right)^\pi.$$

Proof.

$$\begin{aligned}
\int_{x=1}^{t-1} \frac{x}{(x + \lambda(x))^\pi} dx &= \int_{x=1}^{t-1} \frac{x}{(x + 1 - e^{-ix})^\pi} dx \\
&\geq \int_{x=1}^{t-1} \frac{x^\pi}{(x + 1 - e^{-ix})^\pi} dx \\
&\geq \int_{x=1}^{t-1} \left(\frac{xe^{ix}}{xe^{ix} + e^{ix} - 1} \right)^\pi dx \\
&\geq \int_{x=1}^{t-1} \left(\frac{xe^{ix}}{xe^{ix} + xe^{ix} - x} \right)^\pi dx \\
&= \int_{x=1}^{t-1} \left(\frac{e^{ix}}{2e^{ix} - 1} \right)^\pi dx && \text{Using Jensen's Inequality} \\
&\geq \left[\int_{x=1}^{t-1} \frac{e^{ix}}{2e^{ix} - 1} dx \right]^\pi \\
&= \left(\frac{1}{2i} \left[\ln(2e^{i(t-1)} - 1) - \ln(2e^i - 1) \right] \right)^\pi
\end{aligned}$$

□

Lemma 3 (Exploration Dampening: Annealing). *Since $\tau_{(a)}^R(t)$ denotes the number of times the a -th submodular action is chosen in the first $t - 1$ steps by the uniform branch of the algorithm, $a \in [\mathcal{K}]$, then in the case of $\lambda(t) = 1 - e^{-it}$, for $i \geq 0$ (i.e. growing exploration dampening probability), we have the following:*

$$\begin{aligned}
&\mathbb{P} \left(\bigcap_{j=1}^{\mathcal{K}} \{ \tau_{(a)}^R(t) \geq \frac{1}{2\mathcal{K}} \left(\frac{1}{2i} \left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1) \right] \right)^\pi \} \right) \\
&\geq 1 - \mathcal{K} \exp \left(- \frac{3}{28\mathcal{K}} \left(\frac{1}{2i} \left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1) \right] \right)^\pi \right)
\end{aligned}$$

Proof.

$$\begin{aligned}
\mathbb{E}(\tau_a^R(t)) &= \sum_{r=1}^{t-1} \mathbb{P}(\zeta < \Xi_r \cap f_t = f^j) \\
&= \sum_{r=1}^{t-1} \mathbb{P}(\zeta < \Xi_r) \mathbb{P}(f^{(a_t)} = f^{(a)}) = \sum_{r=1}^{t-1} \frac{\Xi_r}{\mathcal{K}} = \frac{1}{\mathcal{K}} \sum_{r=1}^{t-1} \frac{r}{(r + \lambda(r))^\pi} \\
&\geq \frac{1}{\mathcal{K}} \int_{x=1}^{x=t-1} \frac{x}{(x + \lambda(x))^\pi} dx = \frac{1}{\mathcal{K}} \int_{x=1}^{x=t-1} \frac{x}{(x + 1 - e^{-ix})^\pi} dx \\
&\geq \frac{1}{\mathcal{K}} \left(\frac{1}{2i} \left[\ln(2e^{i(t-1)} - 1) - \ln(2e^a - 1) \right] \right)^\pi
\end{aligned}$$

The last inequality comes from [Proposition 2](#)

We define the variance of $\tau_{(a)}^R(t)$ as $\sigma(\tau_{(a)}^R(t))$ and the corresponding upperbound as $\mathcal{Z}(\sigma(t))$

$$\sigma(\tau_a^R(t)) = \sum_{r=1}^{t-1} \frac{\Xi}{\mathcal{K}} \left(1 - \frac{\Xi}{\mathcal{K}} \right) \leq \frac{1}{\mathcal{K}} \sum_{r=1}^{t-1} \Xi_r = \frac{1}{\mathcal{K}} \sum_{r=1}^{t-1} \frac{r}{(r + 1 - e^{-ir})^\pi} = \mathcal{Z}(\sigma(t))$$

Using Bernstein's inequality

$$\begin{aligned}
\mathbb{P}\left(\tau_a^R(t) \leq \frac{\mathcal{Z}(\sigma(t))}{2}\right) &= \mathbb{P}\left(\tau_a^R(t) - \mathcal{Z}(\sigma(t)) \leq -\frac{\mathcal{Z}(\sigma(t))}{2}\right) \\
&\leq \exp\left(\frac{\frac{-\mathcal{Z}(\sigma(t))^2}{8}}{\sigma(\tau_a^R(t)) + \frac{1}{3}\frac{\mathcal{Z}(\sigma(t))}{2}}\right) \leq \exp\left(\frac{\frac{-\mathcal{Z}(\sigma(t))^2}{8}}{\mathcal{Z}(\sigma(t)) + \frac{1}{3}\frac{\mathcal{Z}(\sigma(t))}{2}}\right) \\
&\leq \exp\left(-\frac{3\mathcal{Z}(\sigma(t))}{28}\right) \leq \exp\left(-\frac{3}{28\mathcal{K}}\left(\frac{1}{2i}\left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1)\right]\right)^\pi\right)
\end{aligned}$$

By union bound method

$$\begin{aligned}
&\mathbb{P}\left(\bigcup_{j=1}^{\mathcal{K}}\{\tau_{(a)}^R(t) \leq \frac{1}{2\mathcal{K}}\left(\frac{1}{2i}\left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1)\right]\right)^\pi\}\right) \\
&\leq \mathcal{K}\mathbb{P}\left(\tau_1^R(t) \leq \frac{1}{2\mathcal{K}}\left(\frac{1}{2i}\left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1)\right]\right)^\pi\right) \\
&\leq \mathcal{K}\exp\left(-\frac{3}{28\mathcal{K}}\left(\frac{1}{2i}\left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1)\right]\right)^\pi\right)
\end{aligned}$$

Therefore

$$\begin{aligned}
&\mathbb{P}\left(\bigcap_{j=1}^{\mathcal{K}}\{\tau_{(a)}^R(t) \geq \frac{1}{2\mathcal{K}}\left(\frac{1}{2i}\left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1)\right]\right)^\pi\}\right) \\
&\geq 1 - \mathcal{K}\exp\left(-\frac{3}{28\mathcal{K}}\left(\frac{1}{2i}\left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1)\right]\right)^\pi\right)
\end{aligned}$$

□

Theorem 3 (Regret Guarantees). Under Assumptions **b - d**, for all $t > t_0$ and with $\lambda(t) = 1 - e^{-it}$, with probability at least

$$1 - \mathcal{K} \exp \left(- \frac{3}{28\mathcal{K}} \left(\frac{1}{2i} \left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1) \right] \right)^\pi \right)$$

the expected instantaneous regret incurred by the arm selection policy satisfies

$$\begin{aligned} \mathbb{E}[\text{Regret}_t] &:= \mathbb{E}_{\mathcal{B}_t} \mathbb{E}_{\hat{a}_t \in \mathcal{A}} \mathbb{E}_{\vartheta} [\vartheta(a_t^* | \mathcal{B}_t) - \vartheta(\hat{a}_t | \mathcal{B}_t)] \\ &= O\left(\frac{1}{t}\right) + O\left(\frac{4(\mathfrak{C}_{(a)} + \mathfrak{C}_*)}{\varrho} t^{-\pi/2} \sqrt{\ln t}\right), \end{aligned} \quad (23)$$

where \mathfrak{C}_* is the approximation constant corresponding to the optimal arm a^* .

Proof. Continuing from the same step in Section H.3 Theorem 1, we have the following:

For the case of $\tau_{(a)}^R(t) \geq \frac{1}{2\mathcal{K}} \left(\frac{1}{2i} \left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1) \right] \right)^\pi$ for all a via Proposition 2. Let $\mathfrak{C}_{(a)}$ indicate the constant from Theorem 2 and let n_a be the minimal training data size. We choose $t_0 > e^{(2\mathcal{K} \max\{e, \max_a n_a\})}$. Since the $x \rightarrow \sqrt{\frac{\ln(x)}{x}}$ is monotone decreasing for $x > e$, the above expression is further bounded by

$$\begin{aligned} &\leq \mathbb{E}_{\mathcal{B}_t} \Delta_{(a)}(\mathcal{B}_t) \frac{\epsilon_t}{\mathcal{K}} + \frac{4}{\varrho} \mathfrak{C}_{(a)} \sqrt{\frac{\ln(\tau_{(a)}(t))}{\tau_{(a)}(t)}} + \frac{4}{\varrho} \mathfrak{C}_* \sqrt{\frac{\ln(\tau_*(t))}{\tau_*(t)}} \\ &\leq \mathbb{E}_{\mathcal{B}_t} \Delta_{(a)}(\mathcal{B}_t) \frac{\epsilon_t}{\mathcal{K}} + \frac{4}{\varrho} \mathfrak{C}_{(a)} \sqrt{\frac{\ln(\tau_i^R(t))}{\tau_i^R(t)}} + \frac{4}{\varrho} \mathfrak{C}_* \sqrt{\frac{\ln(\tau_*^R(t))}{\tau_*^R(t)}} \\ &\leq \frac{\mathbb{E}_{\mathcal{B}_t} \Delta_{(a)}(\mathcal{B}_t)}{t\mathcal{K}} + \frac{4}{\varrho} \left[\mathfrak{C}_{(a)} + \mathfrak{C}_* \right] \sqrt{\frac{\ln\left(\frac{1}{2\mathcal{K}} \left(\frac{1}{2i} \left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1) \right] \right)^\pi\right)}{\frac{1}{2\mathcal{K}} \left(\frac{1}{2i} \left[\ln(2e^{i(t-1)} - 1) - \ln(2e^{(a)} - 1) \right] \right)^\pi}} \end{aligned} \quad (24)$$

□

I Broader Impact

The primary aim of our work is to improve the data efficiency of machine learning training pipelines via submodular subset selection. By leveraging principled selection algorithms—such as monotone submodular functions, we can reduce the number of training examples needed without sacrificing model performance. This contributes directly to more sustainable and accessible machine learning, especially in scenarios where training data or compute is limited.

Societal and Environmental Benefits: Reducing the amount of data required for training has multiple practical benefits. For large-scale models, this can translate into lower energy consumption and a reduced carbon footprint. For smaller research labs or applications in low-resource settings, our approach can make training state-of-the-art models more feasible.

Equity and Fairness: By allowing for careful and task-informed selection of training data, our methods could help surface underrepresented or domain-critical samples early in training. However, care must be taken to ensure that the subset selection process does not reinforce existing dataset biases. We encourage practitioners to combine our framework with fairness-aware selection techniques and to audit the resulting models for any performance disparities across groups.

Scientific Impact: More broadly, this work highlights the growing role of data-centric approaches in machine learning research, particularly for compute efficient machine learning research.