RETHINKING SAMPLE UTILITY FOR CORESET SELECTION UNDER EXTREME COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Coreset selection provides a practical alternative to dataset distillation by preserving real, inspectable samples, yet existing methods degrade sharply under extreme compression. We attribute this failure to two mismatches: criteria designed for larger budgets undervalue prototypical samples, while fixed evaluators cannot capture how sample utility changes with the evolving coreset. We propose Model-State-Conditioned Coreset Selection (MCCS), which initializes with prototypical samples, updates utility scores using a model trained on the current subset, and filters already-learned samples to prioritize complementary information. Across CIFAR-10 and ImageNet-1K, MCCS achieves the best or tied-best coreset performance at every budget from 1 to 200 images per class. MCCS also surpasses dataset distillation as the budget or dataset complexity increases and transfers effectively across downstream tasks and architectures. These results establish model-state-conditioned utility estimation as a key requirement for coreset selection under extreme compression.

1 INTRODUCTION

The growing data and compute demands of modern AI have made efficient storage and training increasingly important. Dataset compression addresses these demands by retaining the utility of a training set under a limited storage budget. Coreset selection (CS) preserves a subset of real samples from the original dataset (Guo et al., 2022), whereas dataset distillation (DD) synthesizes a compact training set (Wang et al., 2018; Zhao & Bilen, 2021). Most coreset studies consider moderate compression ratios that retain 30%–70% of the original data, while dataset distillation typically targets extreme budgets of 1–50 images per class (IPC). Consequently, DD has dominated research on extreme dataset compression, and the potential of real-image coreset selection in this regime remains less understood.

We formulate extreme coreset selection as a sample-utility estimation problem. Every selection strategy estimates how much each candidate contributes to the selected subset. At moderate budgets, approximate estimates can remain effective because the subset contains enough samples to tolerate redundancy or imperfect rankings. Under extreme budgets, each selection carries a substantially higher opportunity cost. Redundant samples consume limited capacity and reduce the complementary information available for training. This regime therefore provides a stringent test of whether a selection criterion captures marginal training utility. Notably, existing submodular, coverage-based, budget-aware, and multi-stage methods can deteriorate sharply and even underperform random selection under these constraints.

We identify two mechanisms behind this degradation. First, the utility criterion may be mismatched to the budget. Criteria that prioritize difficult or boundary samples can refine a model when the subset already provides sufficient class coverage. When only a few samples can be retained, however, the subset must first establish stable class-level representations. In this setting, prototypical samples can provide greater marginal utility than globally difficult examples (Sorscher et al., 2022). Second, the evaluator used to estimate utility may be mismatched to the model induced by the selected subset. Existing methods often account for previously selected samples while retaining a fixed model or representation. Their scores therefore describe a candidate relative to a fixed feature space rather than its contribution to the model trained on the evolving coreset. As the coreset changes, samples

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

that appeared useful initially may become redundant, while previously low-priority samples may become informative.

To address these limitations, we propose Model-State-Conditioned Coreset Selection (MCCS). MCCS first constructs a stable initial subset from prototypical samples. It then trains an evaluator on the current coreset, removes candidates that the evaluator already classifies correctly, and re-estimates the utility of the remaining samples. Repeating this process conditions each selection stage on the model state induced by the accumulated coreset. Representative initialization establishes the training foundation, while model-state-conditioned scoring and learned-concept filtering direct subsequent selections toward information that the current coreset does not yet support.

We evaluate MCCS on CIFAR-10 and ImageNet-1K across budgets from 1 to 200 IPC. MCCS achieves the best or on-par performance among coreset methods at every evaluated budget. On CIFAR-10 at 200 IPC, MCCS reaches 77.5% accuracy and exceeds the strongest coreset baseline by 2.0%pp. Dataset distillation remains advantageous at 1 and 10 IPC on CIFAR-10, but MCCS surpasses it from 50 IPC onward. On ImageNet-1K, MCCS outperforms the compared distillation methods at every budget. Fixed-evaluator and filtering ablations directly validate the contributions of model-state-conditioned scoring and learned-concept filtering. MCCS also retains its performance under soft-label, downstream-transfer, and cross-architecture evaluations. These findings show that the limitations of coreset selection under extreme compression arise from the selection strategy rather than an inherent deficiency of real data.

2 Related Work

Dataset Distillation. Dataset distillation constructs a compact synthetic training set that approximates the utility of the original data (Wang et al., 2018; Zhao & Bilen, 2021). Early approaches synthesize data by matching gradients, feature distributions, or optimization trajectories, as exemplified by MTT (Cazenavette et al., 2022) and DATM (Guo et al., 2023). Recent model-inversion methods such as SRe^{2}L (Yin et al., 2023) extend distillation to ImageNet-scale datasets, while RDED (Sun et al., 2024) combines information from multiple real images into each compressed sample. These methods achieve strong performance at very small budgets. However, many recent approaches rely on teacher-generated soft labels whose storage cost can equal or exceed that of the distilled images (Yu et al., 2025). We therefore evaluate dataset distillation and coreset selection using hard labels under a strict storage budget and report a separate soft-label comparison following the configuration used by recent distillation methods.

Coreset Selection. Coreset selection retains real samples according to their estimated contribution to downstream training. Sample-wise methods assign independent importance scores using gradient norms, prediction errors, or forgetting statistics (Killamsetty et al., 2021a; Paul et al., 2021). Set-conditioned approaches additionally account for relationships among selected samples through submodular objectives, diversity measures, or coverage criteria (Mirzasoleiman et al., 2020; Zheng et al., 2022). These methods primarily target moderate compression ratios, where the selected subset can tolerate some redundancy or imperfect utility estimates. Under extreme compression, each selection consumes a substantial fraction of the available budget. Effective utility estimation must therefore account for not only the properties of each sample, but also the information already represented by the current coreset.

Recent methods introduce budget-aware, multi-stage, or model-guided selection mechanisms. DQ partitions the dataset into multiple bins and samples from each bin (Zhou et al., 2023), while ADQ modifies bin allocation using representativeness, diversity, and importance estimates (Li et al., 2025). CCS and DUAL incorporate coverage or training dynamics into selection, and DQAS combines class-dependent budget allocation with expected-error-reduction sampling (Zhao et al., 2024). DQAS also retrains its evaluator on the growing subset and is therefore the closest prior method to MCCS. MCCS differs in both its target setting and its formulation. It retains the class-balanced IPC protocol used in extreme dataset compression and conditions both sample eligibility and utility estimation on the model trained on the current coreset. Learned-concept filtering constructs a state-dependent candidate pool, while model-state-conditioned scoring re-evaluates candidate utility within that pool. Consequently, the candidate set and its utility estimates evolve jointly with the accumulated coreset.

Relation to Model-Guided Training Strategies. MCCS is conceptually related to curriculum learning, boosting, and hard-example mining, which also use model feedback to modify the treatment of training samples (Bengio et al., 2009; Schapire, 2013; Shrivastava et al., 2016). Their objectives nevertheless differ. Curriculum learning orders an available dataset, boosting reweights examples and combines learners, and hard-example mining emphasizes difficult samples during training. MCCS instead constructs a permanently stored coreset for training an independent downstream model. Starting from a representative initialization, MCCS uses the model trained on the accumulated coreset to identify unsupported information and reassess candidate utility. The resulting selection trajectory evolves with the coreset rather than following a predefined curriculum.

The contribution of MCCS therefore lies not in multi-stage selection or model feedback alone, but in formulating extreme coreset selection as model-state-conditioned utility estimation. MCCS couples representative initialization, learned-concept filtering, and model-state-conditioned scoring to limit redundant selection under severe budget constraints. This distinction is consequential in a regime where recent budget-aware, multi-stage, and model-guided methods frequently fail to outperform random selection.

3 Problem Formulation

We formulate extreme coreset selection as a problem of estimating sample utility under severe budget constraints. While coreset methods differ in their selection rules, they all rely on an estimate of marginal utility of samples. We first define the ideal coreset objective and the corresponding marginal utility. We then show that existing methods can fail under extreme budgets because their utility estimates are mismatched to the budget regime and the model state induced by the selected subset.

3.1 Preliminaries

Given a dataset $\mathcal{D} = (x_i, y_i)_{i=1}^N$ with labels $y_i \in \mathcal{Y}$, coreset selection seeks a subset $\mathcal{S} \subset \mathcal{D}$ under a storage budget B. The budget includes the storage of both images and their associated labels. Thus, although some dataset distillation methods use soft labels, their storage must also be accounted for in the budget.

Let $\mathcal{A}$ denote a training algorithm, and let $\mathcal{A}(\mathcal{S})$ denote the model obtained by training on $\mathcal{S}$. We define $U(\mathcal{S})$ as the utility of $\mathcal{S}$, measured by the generalization performance of $\mathcal{A}(\mathcal{S})$ under the target evaluation protocol. The ideal coreset objective is

$$\mathcal{S}^* = \underset{\mathcal{S}\subset\mathcal{D},|\mathcal{S}|=B}{\arg,\max}\ U(\mathcal{S}). \tag{1}$$

Directly optimizing this objective is intractable because it requires training and evaluating models for many candidate subsets. Practical methods therefore use a sample utility estimator q. For a candidate sample $z = (x, y) \in \mathcal{D} \setminus \mathcal{S}$, the ideal sample utility is its marginal contribution when added to the current subset,

$$q^\star(z \mid \mathcal{S}) = U(\mathcal{S} \cup z) - U(\mathcal{S}). \tag{2}$$

Since $q^\star$ is generally unavailable, practical methods approximate it using an estimator,

$$q(z \mid \mathcal{S}; \phi) \approx q^\star(z \mid \mathcal{S}), \tag{3}$$

where ϕ denotes the evaluator used to estimate utility.

Coreset strategies instantiate Eq. 3 in various ways. Sample-wise methods ignore the conditioning set and use $q(z; \phi)$, as in gradient norm, EL2N, and forgetting counts. Set-conditioned methods use $q(z \mid \mathcal{S}; \phi)$ to model dependencies among samples, as in submodular, coverage, and multi-stage selection. These strategies differ in how they estimate utility, but their effectiveness ultimately depends on how closely q matches the ideal marginal utility in Eq. 2.

The importance of this approximation depends on the budget regime. At moderate compression ratios, approximate utility estimates remain effective because selecting similar samples is not necessarily harmful; redundancy can strengthen the empirical support for learned features and improve generalization. Under extreme budgets, however, redundancy consumes scarce capacity and limits the complementary information available for training, making accurate sample-utility estimation critical. We next identify two failure modes of existing strategies under severe budget constraints.

Table 1: Validation accuracy (%) on CIFAR-10 (50 IPC). H and L denote selecting from high and low forgetting scores first, respectively.

Type	Arch. train/test	Synth.		Real		
		MTT	DATM	Rand.	Forget (H)	Forget (L)
Self	ConvNet/ConvNet	71.60	**76.10**	52.59	29.28	59.23
Cross	ConvNet/RN-18	60.67	**61.71**	53.78	28.48	59.73
	ConvNet/RN-50	42.41	30.37	32.99	24.88	**43.62**
	ConvNet/AlexNet	43.59	50.66	45.97	27.91	**54.52**
	ConvNet/MLP	34.77	34.30	36.25	23.82	**41.04**
	Avg.	50.61	50.62	44.28	26.87	**51.63**

3.2 Budget-Mismatched Utility Criteria

The first failure mode is the mismatch between the utility criterion and the budget. Many selection criteria assign high utility to hard samples because these examples can help refine a decision boundary once a sufficient training foundation is available. However, when only a few samples can be selected, the subset must first provide stable anchors for training. We observe this effect in direct comparison with dataset distillation. As shown in Table 1, reversing the conventional Forgetting criterion to prioritize easy samples substantially narrows the gap to DATM, a representative dataset distillation baseline. The advantage of real images is especially visible under cross-architecture evaluation, where distilled images can degrade on architectures different from the one used during synthesis. These results indicate that poor performance at small budgets can arise from a budget-mismatched utility criterion rather than from an inherent limitation of real images.

3.3 Evaluator-Mismatched Utility Estimation

The second failure mode is the mismatch between the utility estimator and the model induced by the evolving coreset. After a representative subset has been selected, the challenge is to identify samples that provide additional training signal beyond what the current subset already provides. Existing strategies partly address this by conditioning utility on previously selected samples. However, they compute this utility with a fixed evaluator,

$$q(z \mid \mathcal{S}_t; \phi_{\text{fix}}), \tag{4}$$

where $\mathcal{S}_t$ is the subset selected at stage t, and ϕ_{fix} denotes fixed evaluator parameters. Although the score depends on $\mathcal{S}_t$, the evaluator itself is independent of the model that would be induced from $\mathcal{S}_t$. Thus, q may reflect a candidate's relation to the selected samples in a fixed representation, rather than its marginal training value for the current coreset-induced model. Consequently, a candidate can receive high utility under the fixed evaluator while contributing little when added to the coreset.

The relevant evaluator should therefore reflect the model state induced by the current subset. Let

$$\theta_t = \mathcal{A}(\mathcal{S}_t) \tag{5}$$

denote the model obtained by training on $\mathcal{S}_t$. A model-state-conditioned estimator takes the form

$$q(z \mid \mathcal{S}_t; \theta_t). \tag{6}$$

Unlike Eq. 4, this estimator changes as the subset changes. It evaluates whether a candidate provides additional benefit to the model induced by the current coreset, rather than whether it remains useful under a fixed representation.

To demonstrate this, we present a toy example tracking the top 5% of samples with the highest quality scores in MCCS (samples with lowest classification loss with respect to models trained on subsets) across 5 selection stages (no filtering is involved in this procedure). Table 2 reveals that the overlap between these sets can be as low as 13%, indicating that samples considered important drastically varies between different training stages. This dynamic behavior confirms that static evaluators cannot accurately estimate marginal quality relative to the evolving model under low budgets.

Table 2: Overlap (%) of top 5% quality samples in CIFAR-10 between five consecutive training stages.

Transition	IPC Budget	
	10	50
Stage $1 \rightarrow 2$	13.00	23.28
Stage $2 \rightarrow 3$	38.76	46.76
Stage $3 \rightarrow 4$	49.48	50.96
Stage $4 \rightarrow 5$	49.32	56.28

4 METHODOLOGY

In this section, we present **Model-State-Conditioned Coreset Selection (MCCS)**, a framework that evaluates candidate utility relative to the model state induced by the current coreset. We first introduce representative initialization in Sec. 4.1, which constructs a stable initial subset using representative samples. We then present the model-state-conditioned selection criterion in Sec. 4.2, which combines model-state scoring and learned-concept filtering. Model-state scoring evaluates candidates using a model trained on the current coreset, while learned-concept filtering excludes samples that this model already predicts correctly. Together, these components ensure that newly selected samples complement the existing coreset under severe budget constraints.

4.1 REPRESENTATIVE INITIALIZATION

MCCS begins by constructing an initial coreset from representative samples. This step addresses the budget-dependent role of prototypes discussed in Sec. 3.2. When the budget is extremely small, the first selected samples should provide stable class-level information rather than focus on difficult or boundary samples.

Let $\mathcal{S}_0$ denote the initial coreset. Given an initialization budget B_0, MCCS selects representative samples according to a score $r(z)$:

$$\mathcal{S}_0 = \underset{\substack{\mathcal{S} \subset \mathcal{D} \\ |\mathcal{S}| = B_0}}{\arg\max} \sum_{z \in \mathcal{S}} r(z). \tag{7}$$

The representative score can be instantiated by a prototype-based criterion. This initialization is not intended to solve the full selection problem. Instead, it provides a stable initial subset from which later stages can estimate additional marginal utility relative to the current coreset-trained model.

4.2 MODEL-STATE-CONDITIONED SELECTION CRITERION

After initialization, MCCS allocates the remaining budget using a model-state-conditioned selection criterion. At stage t, let $\mathcal{S}_t$ denote the current coreset. MCCS trains a model on this subset,

$$\theta_t = \mathcal{A}(\mathcal{S}_t), \tag{8}$$

where $\mathcal{A}$ is the target training procedure. The resulting model θ_t serves as a state-dependent evaluator of what the current coreset already supports.

Model-state scoring. Given the current model state θ_t, MCCS estimates the utility of each remaining candidate through a model-state-conditioned score $q(z \mid \mathcal{S}_t; \theta_t)$. Unlike fixed utility estimates, this score is recomputed after the coreset changes, since the evaluator is the model trained on the current subset. Thus, the score reflects whether a candidate provides additional training value relative to the current coreset-induced model.

Learned-concept filtering. Before selecting the next batch, MCCS removes samples already supported by the current model. This yields the residual candidate pool

$$\mathcal{C}_t = \{(x, y) \in \mathcal{D} \setminus \mathcal{S}_t \mid \underset{k}{\arg\max}\, f_{\theta_t}(x)_k \neq y\}. \tag{9}$$

This filtering step does not define utility by global hardness. Instead, it defines residual candidates relative to the current model state. Samples excluded from $\mathcal{C}_t$ are treated as already supported by the current coreset, while samples remaining in $\mathcal{C}_t$ are candidates for additional utility.

Within the residual pool, MCCS selects the next batch using the model-state-conditioned utility scores. Given a stage budget b_t, MCCS selects

$$\mathcal{B}_t = \underset{\substack{\mathcal{B}\subset\mathcal{C}_t \\ |\mathcal{B}|=b_t}}{\arg\max} \sum_{z\in\mathcal{B}} q(z \mid \mathcal{S}_t; \theta_t). \tag{10}$$

The coreset is then updated as

$$\mathcal{S}_{t+1} = \mathcal{S}_t \cup \mathcal{B}_t. \tag{11}$$

The model is retrained on the updated coreset, producing a new state θ_{t+1} and a new residual candidate pool. By repeating this process, MCCS ensures that newly selected samples complement the existing coreset, while avoiding samples already supported by the current coreset-trained model.

5 Experiments

5.1 Implementation Details

Datasets. Following standard practice in dataset distillation and coreset selection literature, we evaluate all methods on CIFAR-10 (Krizhevsky et al., 2009) and ImageNet-1K (Deng et al., 2009). CIFAR-10 provides a standard small-scale benchmark for extreme dataset compression, while ImageNet-1K evaluates whether the same trends hold in a large-scale setting.

Baselines. We compare MCCS against two major classes of dataset compression methods: dataset distillation (DD) and coreset selection (CS). For DD, we include MTT (Cazenavette et al., 2022), DATM (Guo et al., 2023), SRe^{2}L (Yin et al., 2023), and RDED (Sun et al., 2024) in the main hard-label comparison. We additionally compare against recent soft-label distillation methods, including CDA (Yin & Shen, 2024), CUDD (Cui et al., 2025), EDC (Chen et al., 2024), NRR-DD (Tran et al., 2025), CV-DD (Chen et al., 2025), and Info Util (Wang et al., 2026).

For CS, we compare against Herding (Welling, 2009), K-means (Guo et al., 2022), GradMatch (Killamsetty et al., 2021a), Glister (Killamsetty et al., 2021b), Forgetting (Toneva et al., 2018), EL2N (Paul et al., 2021), GraphCut (Mirzasoleiman et al., 2020), Coverage-Centric Coreset Selection (Zheng et al., 2022), Dataset Quantization (Zhou et al., 2023), DUAL (Cho et al., 2025), DQAS (Zhao et al., 2024), and ADQ (Li et al., 2025). Unless otherwise stated, all coreset methods select samples using a ResNet-18 backbone. We use official implementations when available, including the DeepCore library (Guo et al., 2022) and the benchmark codebase of Moser et al. (Moser et al., 2025). Following the standard Glister setup, we assume access to a 1% validation set.

Training. All datasets are evaluated using the same downstream training protocol whenever possible. We train ResNet-18 (He et al., 2016) and ViT-Tiny (Dosovitskiy et al., 2020) classifiers to evaluate both same-architecture and cross-architecture performance. The learning rate is fixed for each IPC setting, except for MTT and DATM, where we use the learning rates specified by their official implementations. In the hard-label setting, all methods are trained using only image-label pairs under the same storage budget. In the soft-label setting, we apply the same soft labels used by the distillation baselines to the images selected by MCCS, enabling a direct comparison under the label configuration used in recent DD benchmarks. Full training details are provided in Appendix B.

MCCS Implementation. MCCS follows the stage-wise procedure described in Sec. 4. We initialize the coreset with representative samples, using training loss for CIFAR-10 and EL2N norm for ImageNet-1K. At each stage, we train an intermediate model on the current coreset and employ the negative classification loss as the model-state-conditioned utility score for the remaining candidates. The residual candidate pool is constructed by filtering out samples already correctly classified by the intermediate model. For 1 IPC, we use a single selection stage; for larger budgets, we increase the number of stages so that utility estimates are updated as the coreset grows. Detailed stage schedules and hyperparameters are provided in Appendix B.

Table 3: Hard-label validation accuracy (%) of ResNet-18 for representative and strongest baselines. **MCCS** is bold, and the strongest displayed non-MCCS coreset result is underlined. Values are mean ± standard deviation over three runs; the exhaustive comparison is in Appendix Table 7.

Method	1 IPC	10 IPC	50 IPC	100 IPC	200 IPC
CIFAR-10					
MTT	$35.5_{\pm 1.2}$	$48.7_{\pm 0.7}$	$60.7_{\pm 0.9}$	–	–
DATM	$30.6_{\pm 1.0}$	$51.0_{\pm 0.5}$	$61.7_{\pm 0.7}$	$70.3_{\pm 0.4}$	$74.3_{\pm 0.4}$
Random	$16.7_{\pm 1.1}$	$32.9_{\pm 0.8}$	$53.8_{\pm 0.6}$	$63.9_{\pm 1.2}$	$71.7_{\pm 0.3}$
Forgetting	$17.7_{\pm 1.0}$	$37.5_{\pm 1.3}$	$59.7_{\pm 0.4}$	$69.1_{\pm 0.1}$	$75.5_{\pm 0.3}$
EL2N	$23.7_{\pm 0.9}$	$42.5_{\pm 0.2}$	$53.4_{\pm 0.0}$	$62.3_{\pm 0.2}$	$67.8_{\pm 0.1}$
ADQ	$15.7_{\pm 0.9}$	$26.2_{\pm 0.7}$	$39.8_{\pm 0.6}$	$49.6_{\pm 0.5}$	$60.4_{\pm 0.4}$
DQAS	$15.0_{\pm 1.0}$	$26.7_{\pm 0.8}$	$38.5_{\pm 0.7}$	$52.6_{\pm 0.5}$	$65.4_{\pm 0.4}$
MCCS	$\mathbf{24.0}_{\pm 0.6}$	$\mathbf{43.5}_{\pm 0.7}$	$\mathbf{63.2}_{\pm 0.4}$	$\mathbf{70.8}_{\pm 0.4}$	$\mathbf{77.5}_{\pm 0.3}$
ImageNet-1K					
SRe^{2}L	$0.3_{\pm 0.1}$	$2.3_{\pm 0.7}$	$5.7_{\pm 0.5}$	–	–
RDED	$1.8_{\pm 0.4}$	$12.5_{\pm 1.0}$	$29.8_{\pm 1.2}$	$36.3_{\pm 0.6}$	$38.9_{\pm 0.9}$
Random	$1.0_{\pm 0.2}$	$6.6_{\pm 1.1}$	$31.1_{\pm 0.7}$	$46.2_{\pm 0.5}$	$57.3_{\pm 0.8}$
Forgetting	$1.4_{\pm 0.3}$	$14.7_{\pm 0.6}$	$33.3_{\pm 0.9}$	$45.8_{\pm 0.8}$	$55.5_{\pm 0.4}$
EL2N	$2.1_{\pm 0.5}$	$17.9_{\pm 0.7}$	$40.2_{\pm 1.3}$	$46.9_{\pm 0.4}$	$52.7_{\pm 1.0}$
ADQ	$1.3_{\pm 0.3}$	$9.6_{\pm 0.8}$	$37.6_{\pm 0.9}$	$50.5_{\pm 0.6}$	$58.1_{\pm 0.5}$
DQAS	$1.1_{\pm 0.2}$	$2.8_{\pm 0.4}$	$33.2_{\pm 0.8}$	$41.6_{\pm 0.7}$	$56.9_{\pm 0.6}$
MCCS	$\mathbf{2.1}_{\pm 0.5}$	$\mathbf{17.9}_{\pm 0.7}$	$\mathbf{42.3}_{\pm 0.5}$	$\mathbf{51.2}_{\pm 0.2}$	$\mathbf{59.8}_{\pm 0.6}$

5.2 Main Results

Strict storage-budget evaluation. We first evaluate MCCS under a strict storage budget, where both dataset distillation and coreset selection methods are trained with hard labels. Table 3 compares MCCS with state-of-the-art methods on CIFAR-10 and ImageNet-1K using ResNet-18 for evaluation. MCCS achieves the best or tied-best coreset performance at every evaluated budget. On CIFAR-10, it outperforms all coreset baselines from 1 to 200 IPC, exceeding the strongest alternative by 3.5%pp at 50 IPC and 2.0%pp at 200 IPC. On ImageNet-1K, MCCS ties the best coreset result at 1 and 10 IPC and leads by 2.1%pp, 0.7%pp, and 1.7%pp at 50, 100, and 200 IPC, respectively. These consistent gains show that model-state-conditioned selection remains effective as both the dataset complexity and budget increase.

Importantly, these gains are not explained merely by incorporating diversity, budget awareness, or multiple selection stages. Recent methods such as DQ, CCS, DQAS, and DUAL employ coverage-based, budget-aware, adaptive, or multi-stage mechanisms, yet they do not provide consistent improvements under extreme compression and frequently underperform random selection. This suggests that adapting the selection procedure or modeling dependencies within a fixed representation is insufficient. As formulated in Sec. 3, utility must instead be re-estimated with respect to the model induced by the evolving coreset. MCCS combines this model-state-conditioned evaluation with learned-concept filtering, allowing each newly selected sample to complement what the current coreset already supports.

Dataset distillation nevertheless remains effective in specific regimes. On CIFAR-10, MTT and DATM outperform MCCS at 1 and 10 IPC, demonstrating the advantage of synthetic data when only a few images can be retained. As the budget increases, however, MCCS closes this gap and surpasses DATM from 50 IPC onward, reaching a 3.2%pp advantage at 200 IPC. On ImageNet-1K, MCCS outperforms the compared distillation methods at every budget, with the gap widening as the budget grows. Overall, dataset distillation is particularly competitive at the most extreme budgets on smaller datasets, whereas real-image coreset selection scales more reliably with increasing budget and dataset complexity.

Moderate compression regimes. Beyond extreme IPC budgets, MCCS remains competitive at coreset ratios from 10% to 50% and reaches 94.4% accuracy at the 50% budget (Table 8 in the

Table 4: Ablation study of different selection strategies on CIFAR-10 and ImageNet-1K across various IPC budgets. Values are presented as mean ± standard deviation. **MCCS** (highlighted in gray) consistently outperforms baselines or matches the variants.

	CIFAR-10					ImageNet-1K				
Method	1 IPC	10 IPC	50 IPC	100 IPC	200 IPC	1 IPC	10 IPC	50 IPC	100 IPC	200 IPC
No filtering	$24.0_{\pm 0.9}$	$35.0_{\pm 0.8}$	$51.4_{\pm 0.5}$	$58.7_{\pm 0.4}$	$65.1_{\pm 0.6}$	$2.1_{\pm 0.5}$	$17.9_{\pm 0.7}$	$23.9_{\pm 0.8}$	$30.8_{\pm 0.5}$	$44.9_{\pm 0.4}$
Fixed evaluator	$24.0_{\pm 0.9}$	$39.6_{\pm 0.7}$	$56.1_{\pm 0.5}$	$65.2_{\pm 0.4}$	$72.6_{\pm 0.4}$	$2.1_{\pm 0.5}$	$17.9_{\pm 0.7}$	$34.8_{\pm 0.6}$	$33.5_{\pm 0.4}$	$50.2_{\pm 0.5}$
Hard initialization	$12.8_{\pm 1.0}$	$37.2_{\pm 0.9}$	$57.4_{\pm 0.6}$	$66.3_{\pm 0.5}$	$74.1_{\pm 0.4}$	$0.4_{\pm 0.0}$	$1.0_{\pm 0.3}$	$7.2_{\pm 0.9}$	$16.9_{\pm 0.6}$	$40.1_{\pm 0.5}$
Hard prioritization	$12.8_{\pm 1.0}$	$20.2_{\pm 0.8}$	$29.8_{\pm 0.7}$	$36.5_{\pm 0.5}$	$43.9_{\pm 0.4}$	$0.4_{\pm 0.0}$	$1.0_{\pm 0.3}$	$6.8_{\pm 0.8}$	$18.5_{\pm 0.7}$	$47.1_{\pm 0.6}$
MCCS	$\mathbf{24.0_{\pm 0.9}}$	$\mathbf{43.5_{\pm 0.7}}$	$\mathbf{63.2_{\pm 0.4}}$	$\mathbf{70.8_{\pm 0.4}}$	$\mathbf{77.5_{\pm 0.3}}$	$\mathbf{2.1_{\pm 0.5}}$	$\mathbf{17.9_{\pm 0.7}}$	$\mathbf{42.3_{\pm 0.5}}$	$\mathbf{51.2_{\pm 0.2}}$	$\mathbf{59.8_{\pm 0.6}}$

appendix). This indicates that model-state-conditioned utility estimation also remains effective in conventional coreset regimes.

Soft-label evaluation. Under the soft-label configuration commonly used by recent distillation methods, MCCS also outperforms all compared methods, achieving 60.9%, 63.6%, and 65.9% at 50, 100, and 200 IPC, respectively (Table 9 in the appendix). These results exceed the strongest baselines by 0.7%pp, 1.1%pp, and 0.9%pp while retaining real images and avoiding image synthesis.

Generalization to Downstream Tasks. Linear probing on frozen ViT-DINO backbones shows that MCCS outperforms all coreset baselines at every budget (Table 12 in the appendix). At 10 IPC, it achieves 80.3% on CIFAR-10 and 64.5% on ImageNet-1K, exceeding the strongest coreset baselines by 1.7%pp and 0.2%pp; it also surpasses RDED from 5 IPC onward without combining or synthesizing images.

Transferability Across Architectures. MCCS achieves the best performance in all but one bidirectional transfer configuration between MLP and ResNet-18 on CIFAR-10 and between ViT-Tiny (Dosovitskiy et al., 2020) and ResNet-18 on ImageNet-1K (Table 13 in the appendix). At 50 IPC, it reaches 36.10% under ImageNet-1K ViT-to-ResNet transfer, 4.3%pp above the strongest baseline, without architecture-specific tuning of the number of stages.

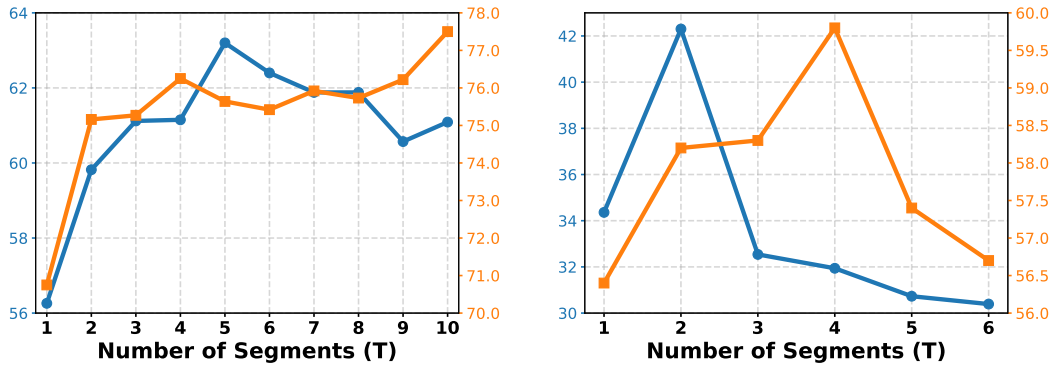
5.3 Ablation Studies

In this section, we conduct ablation studies on the the individual components of the proposed selection strategy and the impact of the number of selection stages.

Efficacy of Model-State-Conditioned Scoring and Filtering. Table 4 isolates the contributions of representative initialization, model-state-conditioned scoring, and learned-concept filtering. The **no filtering** variant retains samples already classified correctly by the intermediate model. The **fixed evaluator** variant fixes the scoring model as pretrained teacher. The **hard initialization** variant replaces the initial prototypes with difficult samples, while **hard prioritization** selects high-loss samples throughout selection.

The fixed-evaluator results demonstrate the importance of conditioning utility on the current model state. From 50 to 200 IPC, MCCS improves over this variant by 4.9–7.1%pp on CIFAR-10 and 7.5–17.7%pp on ImageNet-1K. The methods coincide in single-stage settings, where MCCS does not update the evaluator. Learned-concept filtering provides a separate benefit. At 200 IPC, MCCS exceeds the no-filtering variant by 12.4%pp on CIFAR-10 and 14.9%pp on ImageNet-1K, showing that redundancy becomes increasingly detrimental as the coreset grows. The difficulty-based variants further confirm the importance of representative initialization. On CIFAR-10 at 200 IPC, hard initialization and hard prioritization underperform MCCS by 3.4%pp and 33.6%pp, respectively. Together, these results validate the three central design choices of MCCS. Prototypical initialization establishes a stable foundation, model-state-conditioned scoring updates sample utility as the coreset evolves, and learned-concept filtering directs subsequent selection toward complementary samples.

Influence of Stage Granularity. We examine the effect of the number of selection stages T at 50 and 200 IPC in Fig. 1. An appropriately chosen multi-stage configuration improves over single-



(a) CIFAR-10 (b) ImageNet-1K

Figure 1: Effect of the number of selection stages T at 50 and 200 IPC. The optimal number of stages increases with the budget and is smaller on ImageNet-1K than on CIFAR-10.

stage selection, but the optimal granularity depends on the total budget. On CIFAR-10, performance peaks at $T = 5$ for 50 IPC and $T = 10$ for 200 IPC. On ImageNet-1K, the corresponding optima are $T = 2$ and $T = 4$.

The per-stage budget per class governs this trade-off. If each stage contains too few samples, the intermediate model cannot learn sufficiently reliable features for learned-concept filtering. Conversely, larger budgets support finer-grained updates as the coreset evolves. In practice, allocating 2–20 images per class per stage on CIFAR-10 and 25–50 on ImageNet-1K yields competitive performance. Tables 5 and 6 report the configurations used in our experiments.

5.4 Computational Analysis

We compare the selection time of MCCS with Forgetting, ADQ, and DQAS on a single RTX 4090 GPU (NVIDIA, 2022). Forgetting represents static scoring based on full-dataset training statistics, whereas ADQ and DQAS employ adaptive or multi-stage selection. As shown in Table 10 in the appendix, ADQ and DQAS require comparable or longer selection times than MCCS. Compared with Forgetting, MCCS requires additional computation in most settings because it trains intermediate models on the evolving coreset.

MCCS remains substantially more efficient than dataset distillation at scale. ImageNet-level distillation commonly requires hundreds of GPU hours to optimize synthetic images (Cui et al., 2023), whereas MCCS completes selection within 14.8 hours even at 200 IPC. MCCS therefore provides a practical approach to large-scale dataset compression without requiring image synthesis.

6 Conclusion

We revisit coreset selection under extreme compression and show that its degradation arises from mismatches among the budget, utility criterion, and evaluator. Model-State-Conditioned Coreset Selection (MCCS) addresses these mismatches through representative initialization, model-state-conditioned scoring, and learned-concept filtering. Across CIFAR-10 and ImageNet-1K, MCCS achieves the best or tied-best coreset performance at every evaluated budget from 1 to 200 IPC. Fixed-evaluator and filtering ablations show that updating utility with the evolving model state and excluding already-learned samples both contribute substantially. Dataset distillation remains effective at the smallest CIFAR-10 budgets, while MCCS becomes increasingly competitive as the budget and dataset complexity grow and retains its performance under soft-label, downstream-transfer, and cross-architecture evaluations. These results establish model-state-conditioned utility estimation as an effective principle for coreset selection under severe budget constraints.

AI USE STATEMENT

The authors used generative AI to assist with writing, providing overviews and reviews, and preparing tables and related materials.

REPRODUCIBILITY STATEMENT

The objective, selection rule, and model-state updates are defined in Sections 3 and 4. The appendix provides complete pseudocode, classwise budget allocation, fallback behavior, and the hyperparameters and training schedules used for every dataset and budget in Appendix B. All datasets and architectures used in this work are publicly available.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, 2009.

Gabriel Cazenavette, Tony Testuggine, Bo Wang, Rishi Kumar, Huawei Ma, and S. Vivek Kumar. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11130–11139, 2022.

Jiacheng Chen et al. Dataset distillation via committee voting. *arXiv preprint*, 2025.

Zhaorun Chen et al. Elucidating the design space of dataset condensation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yeseul Cho, Baekrok Shin, Changmin Kang, and Chulhee Yun. Lightweight dataset pruning without full training via example difficulty and prediction uncertainty. *arXiv preprint arXiv:2502.06905*, 2025.

Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Scaling up dataset distillation to imagenet-1k with trajectory matching. In *International Conference on Machine Learning (ICML)*, 2023.

Z. Cui et al. Curriculum dataset distillation. *IEEE Transactions on Image Processing*, 2025.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *Transactions on Machine Learning Research*, 2023.

Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*, pp. 181–195. Springer, 2022.

Ziyao Guo, Kai Wang, George Cazenavette, Hui Li, Kaipeng Zhang, and Yang You. Towards lossless dataset distillation via difficulty-aligned trajectory matching. *arXiv preprint arXiv:2310.05773*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pp. 5464–5474. PMLR, 2021a.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 8110–8118, 2021b.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 and CIFAR-100 datasets. https://www.cs.toronto.edu/~kriz/cifar.html, 2009. Accessed: 2025-07-28.

Muquan Li, Dongyang Zhang, Qiang Dong, Xiurui Xie, and Ke Qin. Adaptive dataset quantization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(11):12093–12101, 2025. doi: 10.1609/aaai.v39i11.33317.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning (ICML)*, 2020.

Brian B Moser, Arundhati S Shanbhag, Stanislav Frolov, Federico Raue, Joachim Folz, and Andreas Dengel. A coreset selection of coreset selection literature: Introduction and recent advances. *arXiv preprint arXiv:2505.17799*, 2025.

NVIDIA. NVIDIA Ada Lovelace Architecture: Professional Graphics Powering the Metaverse and Beyond, 2022. URL https://www.nvidia.com/en-us/design-visualization/ada-lovelace-architecture/. Accessed: 2026-03-03.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Vasiljevic, Puxuan Sun, Piotr Procopio, Piotr Bojanowski, Nicolas Lajugie, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34:20596–20607, 2021.

Robert E Schapire. Explaining adaboost. In *Empirical inference: festschrift in honor of vladimir N. Vapnik*, pp. 37–52. Springer, 2013.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 761–769, 2016.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, S. Surya, and Morcos Ari. Beyond neural scaling laws: Beating power law scaling via data pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 19523–19536, 2022.

Peng Sun, Bei Shi, Daiwei Yu, and Tao Lin. On the diversity and realism of distilled dataset: An efficient dataset distillation paradigm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9390–9399, 2024.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Minh-Tuan Tran et al. Enhancing dataset distillation via non-critical region refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Shaobo Wang et al. Grounding and enhancing informativeness and utility in dataset distillation. *arXiv preprint*, 2026.

Tongzhou Wang, Jun-Yan Liu, Chongruo Zhu, Jianbo Gong, Song Liu, and Jing Ma. Dataset distillation. In *NIPS Workshop on Meta-Learning*, 2018.

Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1121–1128, 2009.

Zeyuan Yin and Zhiqiang Shen. Dataset distillation via curriculum data synthesis in large data era. *Transactions on Machine Learning Research (TMLR)*, 2024.

Zeyuan Yin, Eric Xing, and Zhiqiang Shen. Squeeze, recover and relabel: Dataset condensation at imagenet scale from a new perspective. *Advances in Neural Information Processing Systems*, 36: 73582–73603, 2023.

Ruonan Yu, Songhua Liu, Zigeng Chen, Jingwen Ye, and Xinchao Wang. Heavy labels out! dataset distillation with label space lightening. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5017–5026, 2025.

Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pp. 12674–12685. PMLR, 2021.

Zhenghao Zhao, Yuzhang Shang, Junyi Wu, and Yan Yan. Dataset quantization with active learning based adaptive sampling. In *Computer Vision – ECCV 2024*, pp. 346–362. Springer, 2024. doi: 10.1007/978-3-031-73027-6_20.

Haizhong Zheng, Rui Liu, Fan Lai, and Atul Prakash. Coverage-centric coreset selection for high pruning rates. *arXiv preprint arXiv:2210.15809*, 2022.

Daquan Zhou, Kai Wang, Jianyang Gu, Xiangyu Peng, Dongze Lian, Yifan Zhang, Yang You, and Jiashi Feng. Dataset quantization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17205–17216, 2023.

Algorithm 1 Model-State-Conditioned Coreset Selection (MCCS)

Input: Dataset $\mathcal{D}$ with C classes, total budget B, number of stages T, and initial parameters θ_0.
Output: Final coreset $\mathcal{S}$ of size B.
Distribute the per-class budget across stages as $\{b_t\}_{t=1}^{T}$, assigning any remainder to the earlier stages.
Train the initial model with parameters θ_0 on the full dataset $\mathcal{D}$.
for each class $c \in \{1, \ldots, C\}$ **do**
    Select b_1 samples $\mathcal{S}_{0,c} \subset \mathcal{D}_c$ that maximize the representative score $r(z)$.
end for
Initialize the coreset $\mathcal{S} \leftarrow \mathcal{S}_0 = \bigcup_{c=1}^{C} \mathcal{S}_{0,c}$.
for $t = 2$ **to** T **do**
    Train a model with parameters θ_t on the current coreset $\mathcal{S}$.
    Construct the residual candidate pool $\mathcal{C}_t$ by removing selected and correctly classified samples.
    for each class $c \in \{1, \ldots, C\}$ **do**
        Select a batch $\mathcal{B}_{t,c} \subset \mathcal{C}_{t,c}$ of size b_t that maximizes $q(z \mid \mathcal{S}; \theta_t)$.
    end for
    Set $\mathcal{B}_t \leftarrow \bigcup_{c=1}^{C} \mathcal{B}_{t,c}$.
    Update the coreset $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{B}_t$.
end for
return $\mathcal{S}$.

A MCCS Algorithm Pseudocode

Algorithm 1 presents the complete Model-State-Conditioned Coreset Selection (MCCS) procedure. In practice, we allocate an equal number of images to every class, following the images-per-class protocol commonly used in dataset distillation under extreme budgets. This class-balanced allocation ensures that coreset selection and dataset distillation use the same number of images from each class and prevents class allocation from confounding the comparison.

MCCS distributes the per-class budget $K = B/C$ across T stages as evenly as possible. Each stage first receives $\lfloor K/T \rfloor$ samples per class. If the budget is not divisible by T, we allocate one additional sample per class to each of the earliest stages until the remainder is exhausted. This procedure ensures that stage budgets differ by at most one sample and that their sum equals K. For example, when $K = 10$ and $T = 3$, MCCS allocates 4, 3, and 3 samples per class across the three stages.

MCCS begins by constructing a representative initial coreset. We train an initial model with parameters θ_0 on the full dataset and compute a representative score $r(z)$ for each sample $z = (x, y)$. On CIFAR-10, we use the negative cross-entropy loss

$$r(z) = -\ell_{\mathrm{CE}}(z; \theta_0), \qquad \ell_{\mathrm{CE}}(z; \theta_0) = -\log p_{\theta_0}(x)_y,$$

where $p_{\theta_0}(x)_y$ denotes the predicted probability of the ground-truth class. Maximizing $r(z)$ therefore selects samples with the lowest classification loss.

On ImageNet-1K, we define the representative score using the negative EL2N norm

$$r(z) = -\left\| p_{\theta_0}(x) - \mathbf{e}_y \right\|_2,$$

where $p_{\theta_0}(x)$ is the predicted class-probability vector and $\mathbf{e}_y$ is the one-hot encoding of the ground-truth label. A lower EL2N norm indicates that the prediction is closer to the ground-truth target. MCCS selects the samples with the highest $r(z)$ within each class, which corresponds to selecting those with the lowest EL2N norms. Both initialization scores prioritize samples that the initial model represents with high confidence, providing stable class-level prototypes for the first coreset.

During each subsequent stage, MCCS trains a new model with parameters θ_t on the accumulated coreset $\mathcal{S}$. Learned-concept filtering then constructs the residual candidate pool

$$\mathcal{C}_t = \left\{ (x, y) \in \mathcal{D} \setminus \mathcal{S} \;\middle|\; \arg\max_k p_{\theta_t}(x)_k \neq y \right\}.$$

Table 5: Hyperparameters for MCCS experiments on CIFAR-10. Augmentations include Differentiable Siamese Augmentation (DSA) and CutMix (CM).

Hyperparameter	**1 IPC**	**10 IPC**	**50 IPC**	**100 IPC**	**200 IPC**
Learning Rate	0.025	0.015	0.03	0.025	0.05
Optimizer	SGD	SGD	SGD	SGD	SGD
Epochs	1000	1000	1000	1000	1000
Batch Size	64	64	64	64	64
Augmentations	DSA, CM	DSA, CM	DSA, CM	DSA, CM	DSA, CM
Segments (T)	1	5	5	8	10

Table 6: Hyperparameters for MCCS on ImageNet-1K, where the augmentation used is Random-sized Recrop (RR).

Hyperparameter	**1 IPC**	**10 IPC**	**50 IPC**	**100 IPC**	**200 IPC**
Learning Rate	0.001	0.001	0.001	0.001	0.001
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Epochs	300	300	300	300	300
Batch Size	128	128	128	128	128
Augmentations	RR	RR	RR	RR	RR
Segments (T)	1	1	2	3	4

This pool excludes both previously selected samples and unselected samples that the current model already classifies correctly.

For each remaining candidate, MCCS computes the model-state-conditioned score

$$q(z \mid \mathcal{S}; \theta_t) = -\ell_{\mathrm{CE}}(z; \theta_t).$$

MCCS selects the b_t candidates with the highest scores within each class. Because $\mathcal{C}_t$ contains only misclassified samples, maximizing q selects the lowest-loss samples among the concepts that the current coreset does not yet support. If a classwise residual pool contains fewer than b_t candidates, MCCS selects all available candidates and fills the remaining positions with the highest-scoring samples from the corresponding unselected class pool $\mathcal{D}_c \setminus \mathcal{S}$. This fallback preserves the specified classwise budget, although it was not triggered in any of our experiments.

Retraining the evaluator after every update changes both the candidate pool and the relative ordering of its samples. This process produces a curriculum that begins with representative prototypes and progressively incorporates more difficult samples as the model learns from the growing coreset. Learned-concept filtering and model-state-conditioned scoring therefore reduce redundant selection and direct each subsequent stage toward information not yet captured by the current model.

B MCCS Implementation

We provide the full hyperparameters needed to implement MCCS in Table 5 and Table 6

C Additional Evaluation Settings

C.1 Complete Hard-Label Comparison

Table 7 reports the complete set of hard-label baselines summarized by Table 3 in the main paper.

C.2 Moderate Compression Regimes

The main experiments focus on extreme compression using IPC budgets. We further evaluate whether MCCS remains effective beyond this regime by considering moderate coreset ratios of 10%, 20%, 30%, 40%, and 50%. In this setting, we remove class-wise IPC constraints and select subsets according to the target ratio.

Table 7: Full comparison of dataset distillation and coreset selection methods on CIFAR-10 and ImageNet-1K. We report the validation accuracy (%) of ResNet-18 trained on subsets generated by each method. We mark methods that perform worse than random selection in red. The best results within each category (Dataset Distillation or Coreset Selection) are in **bold**, and the second best are underlined. Values are presented as mean ± standard deviation over three runs.

		CIFAR-10					ImageNet-1K					
Category	**Method**	**1 IPC**	**10 IPC**	**50 IPC**	**100 IPC**	**200 IPC**	**Method**	**1 IPC**	**10 IPC**	**50 IPC**	**100 IPC**	**200 IPC**
Dataset Distillation	MTT	**35.5**$_{\pm 1.2}$	48.7$_{\pm 0.7}$	60.7$_{\pm 0.9}$	-	-	SRe^{2}L	0.3$_{\pm 0.1}$	2.3$_{\pm 0.7}$	5.7$_{\pm 0.5}$	-	-
	DATM	30.6$_{\pm 1.0}$	**51.0**$_{\pm 0.5}$	**61.7**$_{\pm 0.7}$	**70.3**$_{\pm 0.4}$	**74.3**$_{\pm 0.4}$	RDED	**1.8**$_{\pm 0.4}$	**12.5**$_{\pm 1.0}$	**29.8**$_{\pm 1.2}$	**36.3**$_{\pm 0.6}$	**38.9**$_{\pm 0.9}$
Coreset Selection	Random	16.7$_{\pm 1.1}$	32.9$_{\pm 0.8}$	53.8$_{\pm 0.6}$	63.9$_{\pm 1.2}$	71.7$_{\pm 0.3}$	Random	1.0$_{\pm 0.2}$	6.6$_{\pm 1.1}$	31.1$_{\pm 0.7}$	46.2$_{\pm 0.5}$	57.3$_{\pm 0.8}$
	Herding	20.0$_{\pm 0.4}$	34.8$_{\pm 0.4}$	51.0$_{\pm 0.8}$	58.6$_{\pm 0.4}$	68.5$_{\pm 0.4}$	Herding	0.4$_{\pm 0.2}$	4.0$_{\pm 0.8}$	19.1$_{\pm 0.6}$	29.5$_{\pm 0.3}$	49.8$_{\pm 0.7}$
	K-Means	19.9$_{\pm 0.7}$	37.5$_{\pm 1.1}$	53.6$_{\pm 0.2}$	63.7$_{\pm 0.8}$	72.2$_{\pm 0.5}$	K-Means	1.1$_{\pm 0.3}$	3.7$_{\pm 0.4}$	15.7$_{\pm 0.9}$	34.0$_{\pm 0.2}$	52.8$_{\pm 0.5}$
	GradMatch	13.8$_{\pm 1.1}$	26.8$_{\pm 0.6}$	37.8$_{\pm 0.7}$	50.8$_{\pm 0.5}$	56.4$_{\pm 0.8}$	GradMatch	0.8$_{\pm 0.2}$	3.8$_{\pm 0.3}$	11.3$_{\pm 1.0}$	19.9$_{\pm 0.7}$	36.9$_{\pm 0.6}$
	Glister	12.7$_{\pm 0.2}$	10.0$_{\pm 0.0}$	10.2$_{\pm 0.1}$	19.4$_{\pm 0.0}$	23.5$_{\pm 0.1}$	Glister	0.5$_{\pm 0.2}$	3.1$_{\pm 0.5}$	26.8$_{\pm 1.1}$	41.2$_{\pm 0.7}$	39.7$_{\pm 1.2}$
	Forgetting	17.7$_{\pm 1.0}$	37.5$_{\pm 1.3}$	59.7$_{\pm 0.4}$	69.1$_{\pm 0.1}$	75.5$_{\pm 0.3}$	Forgetting	1.4$_{\pm 0.3}$	14.7$_{\pm 0.6}$	33.3$_{\pm 0.9}$	45.8$_{\pm 0.8}$	55.5$_{\pm 0.4}$
	EL2N	23.7$_{\pm 0.9}$	42.5$_{\pm 0.2}$	53.4$_{\pm 0.0}$	62.3$_{\pm 0.2}$	67.8$_{\pm 0.1}$	EL2N	**2.1**$_{\pm 0.5}$	**17.9**$_{\pm 0.7}$	40.2$_{\pm 1.3}$	46.9$_{\pm 0.4}$	52.7$_{\pm 1.0}$
	GraphCut	18.9$_{\pm 1.0}$	36.3$_{\pm 0.8}$	54.6$_{\pm 0.3}$	64.5$_{\pm 0.6}$	71.7$_{\pm 0.3}$	GraphCut	1.8$_{\pm 0.1}$	12.0$_{\pm 0.9}$	38.2$_{\pm 0.8}$	49.1$_{\pm 1.2}$	57.6$_{\pm 0.4}$
	DQ	16.8$_{\pm 0.8}$	34.5$_{\pm 0.5}$	51.3$_{\pm 0.5}$	61.5$_{\pm 0.4}$	70.7$_{\pm 0.3}$	DQ	0.8$_{\pm 0.1}$	5.3$_{\pm 0.3}$	23.8$_{\pm 0.9}$	39.7$_{\pm 0.8}$	53.6$_{\pm 1.1}$
	CCS	23.1$_{\pm 0.9}$	42.3$_{\pm 0.7}$	56.2$_{\pm 0.4}$	62.2$_{\pm 0.3}$	68.6$_{\pm 0.6}$	CCS	1.1$_{\pm 0.3}$	6.7$_{\pm 1.0}$	33.1$_{\pm 0.9}$	47.8$_{\pm 0.1}$	56.8$_{\pm 0.6}$
	DUAL	14.6$_{\pm 1.2}$	20.0$_{\pm 0.5}$	28.3$_{\pm 0.4}$	33.6$_{\pm 0.6}$	40.3$_{\pm 0.2}$	DUAL	0.6$_{\pm 0.0}$	3.5$_{\pm 0.1}$	21.3$_{\pm 0.6}$	42.6$_{\pm 0.2}$	57.2$_{\pm 0.7}$
	ADQ	15.7$_{\pm 0.9}$	26.2$_{\pm 0.7}$	39.8$_{\pm 0.6}$	49.6$_{\pm 0.5}$	60.4$_{\pm 0.4}$	ADQ	1.3$_{\pm 0.3}$	9.6$_{\pm 0.8}$	37.6$_{\pm 0.9}$	50.5$_{\pm 0.6}$	58.1$_{\pm 0.5}$
	DQAS	15.0$_{\pm 1.0}$	26.7$_{\pm 0.8}$	38.5$_{\pm 0.7}$	52.6$_{\pm 0.5}$	65.4$_{\pm 0.4}$	DQAS	1.1$_{\pm 0.2}$	2.8$_{\pm 0.4}$	33.2$_{\pm 0.8}$	41.6$_{\pm 0.7}$	56.9$_{\pm 0.6}$
	MCCS (Ours)	**24.0**$_{\pm 0.6}$	**43.5**$_{\pm 0.7}$	**63.2**$_{\pm 0.4}$	**70.8**$_{\pm 0.4}$	**77.5**$_{\pm 0.3}$	**MCCS (Ours)**	**2.1**$_{\pm 0.5}$	**17.9**$_{\pm 0.7}$	**42.3**$_{\pm 0.5}$	**51.2**$_{\pm 0.2}$	**59.8**$_{\pm 0.6}$

Table 8: Validation accuracy (%) of ResNet-18 on CIFAR-10 under medium-budget coreset selection. MCCS is adapted by removing class-wise constraints to accommodate larger subset ratios. The best results are in **bold**. Values are presented as mean ± standard deviation.

Method	**10%**	**20%**	**30%**	**40%**	**50%**
Random	74.3$_{\pm 0.3}$	86.6$_{\pm 0.2}$	90.0$_{\pm 0.2}$	92.2$_{\pm 0.2}$	92.9$_{\pm 0.1}$
DQ	76.0$_{\pm 0.4}$	86.6$_{\pm 0.3}$	89.7$_{\pm 0.2}$	92.1$_{\pm 0.2}$	93.2$_{\pm 0.1}$
CCS	78.0$_{\pm 0.4}$	**89.9**$_{\pm 0.2}$	92.3$_{\pm 0.2}$	93.2$_{\pm 0.1}$	93.5$_{\pm 0.1}$
DUAL	77.6$_{\pm 0.4}$	89.6$_{\pm 0.2}$	92.3$_{\pm 0.2}$	93.4$_{\pm 0.1}$	93.7$_{\pm 0.1}$
MCCS (Ours)	**79.1**$_{\pm 0.3}$	89.2$_{\pm 0.2}$	**92.4**$_{\pm 0.2}$	**93.5**$_{\pm 0.1}$	**94.4**$_{\pm 0.1}$

Table 8 shows that MCCS remains competitive across moderate compression ratios. Although the gap between selection methods and random sampling typically narrows as the budget increases, MCCS maintains consistent gains and reaches 94.4% accuracy at the 50% budget. These results suggest that model-state-conditioned utility estimation is not limited to extreme IPC settings, but also remains effective in more conventional coreset selection regimes.

C.3 Soft-Label Evaluation

The strict-budget comparison uses hard labels to control storage cost, whereas recent dataset distillation methods often report their strongest performance with soft labels. For a direct comparison under this setting, we apply the same soft-label configuration to the real images selected by MCCS and compare against recent distillation and coreset baselines.

Table 9 reports results from 50 to 200 IPC. MCCS consistently outperforms all compared methods, achieving 60.9%, 63.6%, and 65.9% at 50, 100, and 200 IPC, respectively. These results exceed the strongest baselines by 0.7%pp, 1.1%pp, and 0.9%pp, showing that MCCS remains effective under the label configuration commonly used by recent distillation methods while retaining real images and avoiding image synthesis.

C.4 Selection Time

Table 10 gives the wall-clock selection times used in the computational analysis of the main paper.

Table 9: Validation accuracy (%) of ResNet-18 using soft labels across varying Images Per Class (IPC) budgets on ImageNet-1K. Dataset distillation and coreset selection methods are reported separately. Values are presented as mean ± standard deviation.

Category	Method	50 IPC	100 IPC	200 IPC
Dataset Distillation	SRe^{2}L	$46.8_{\pm 0.8}$	$52.8_{\pm 0.7}$	$57.0_{\pm 0.6}$
	CDA	$53.5_{\pm 0.7}$	$58.0_{\pm 0.6}$	$63.3_{\pm 0.5}$
	RDED	$56.5_{\pm 0.6}$	$58.2_{\pm 0.5}$	$62.5_{\pm 0.6}$
	CUDD	$57.4_{\pm 0.5}$	$61.3_{\pm 0.4}$	$65.0_{\pm 0.4}$
	EDC	$58.0_{\pm 0.7}$	-	-
	NRR-DD	$60.2_{\pm 0.5}$	-	-
	CV-DD	$59.5_{\pm 0.6}$	-	-
	Info Util	$57.6_{\pm 0.6}$	$58.8_{\pm 0.5}$	$63.4_{\pm 0.5}$
Coreset Selection	DQ	$54.8_{\pm 0.9}$	$58.7_{\pm 0.8}$	$62.6_{\pm 1.1}$
	CCS	$57.1_{\pm 0.9}$	$59.8_{\pm 0.1}$	$63.2_{\pm 0.6}$
	DUAL	$51.3_{\pm 0.6}$	$62.6_{\pm 0.2}$	$63.4_{\pm 0.7}$
	ADQ	$57.6_{\pm 0.7}$	$62.5_{\pm 0.5}$	$64.1_{\pm 0.5}$
	DQAS	$53.2_{\pm 0.8}$	$61.6_{\pm 0.6}$	$63.9_{\pm 0.5}$
	MCCS (Ours)	$\mathbf{60.9}_{\pm \mathbf{0.4}}$	$\mathbf{63.6}_{\pm \mathbf{0.3}}$	$\mathbf{65.9}_{\pm \mathbf{0.3}}$

Table 10: Wall-clock selection time on a single RTX 4090 GPU.

Dataset	IPC	Forgetting	ADQ	DQAS	MCCS
CIFAR-10	100	10m	11m	11m	37m
CIFAR-10	200	10m	11m	11m	1h
ImageNet-1K	100	7.7h	30.2h	17.3h	6h
ImageNet-1K	200	7.7h	30.2h	20.2h	14.8h

D Diversity Analysis

To evaluate whether the adaptive selection of MCCS translates into diversity, we compute the Vendi score (Friedman & Dieng, 2023) for each coreset, which measures the effective number of distinct elements in a set. We compare against GraphCut (GC), Coverage-Centric Selection (CCS), and Dataset Quantization (DQ). As shown in Table 11, GC and CCS consistently achieve lower diversity scores, indicating that these conditioning strategies struggle to promote diversity under lower budgets. Notably, DQ achieves marginally higher scores than MCCS at 50 IPC and above, due its multi-stage selection strategy which is very similar to MCCS. However, as shown in Table 3, DQ fails to translate this diversity into downstream performance. This suggests that diversity alone is insufficient without a model-state-conditioned scoring mechanism, as DQ relies on a fixed score, while MCCS does not. MCCS achieves competitive diversity while consistently converting it into performance gains, confirming that its adaptive strategy effectively balances diversity with informativeness.

Table 11: Vendi score comparison of subsets selected by diversity-maximizing methods. The best results are in bold, and the second-best are underlined.

Method	10 IPC	50 IPC	100 IPC	200 IPC
GC	5.12	5.93	6.07	6.07
CCS	5.05	5.66	5.82	5.94
DQ	<u>5.26</u>	**6.06**	**6.19**	**6.21**
MCCS	**5.38**	<u>6.02</u>	**6.19**	<u>6.20</u>



(a) Images from the first segment. (b) Images from the last segment.

Figure 2: Qualitative results of images selected in the first and last segment using MCCS.

E Qualitative Analysis

To understand the characteristics of samples selected by MCCS, we visualize images from the first and last stages in Figure 2. The images are selected under the 100 IPC and $T = 3$ setting on ImageNet. The first subset (Figure 2a) contains canonical samples with clear, distinctive features, such as whole strawberries with typical appearance, establishing foundational patterns for the coreset. In contrast, the last subset (Figure 2b) incorporates challenging hard samples that represent edge cases, such as strawberries with atypical appearance or partial occlusion. These samples capture decision boundary complexities that complement the foundational patterns. This demonstrates how MCCS maintains diversity through adaptive selection, ensuring a balanced representation of both prototypical and challenging samples in the coreset.

F Generalization to Downstream Tasks

We evaluate the transfer learning performance of synthetic and real datasets using a ViT-DINO backbone. The training is done for 10 epochs using the AdamW optimizer, and MCCS initializes the first subset according to the lowest loss and employs 2 stages for coresets larger than 1 IPC. Table 12 shows that MCCS outperforms all coreset baselines at every budget, confirming that its gains extend to transfer learning settings. At 10 IPC, MCCS achieves 80.3% on CIFAR-10 and 64.5% on ImageNet-1K, exceeding the strongest coreset baselines by 1.7%pp and 0.2%pp, respectively.

The results also reveal distinct transfer behaviors among distillation methods. On CIFAR-10, MTT and DATM perform near chance, suggesting that their synthetic images encode information specific to the distillation and training procedure rather than broadly transferable semantic features. In contrast, RDED transfers effectively on ImageNet-1K, particularly at 1 IPC, because it combines patches from four real images into each synthetic image and exposes the classifier to more visual content under a small nominal image budget. Nevertheless, MCCS surpasses RDED from 5 IPC on-

Table 12: Transfer learning performance on CIFAR-10 and ImageNet-1K by linearly probing a ViT-Base-DINO model. Red denotes performance below the random baseline. The best results are in **bold**, and the second best are underlined. Values are presented as mean ± standard deviation over three runs.

	CIFAR-10				ImageNet-1K			
Category	**Method**	**1 IPC**	**5 IPC**	**10 IPC**	**Method**	**1 IPC**	**5 IPC**	**10 IPC**
Dataset Distillation	MTT	11.7±0.4	12.4±0.2	9.1±0.5	SRe^{2}L	25.8±0.4	45.5±0.3	49.8±0.2
	DATM	10.9±0.3	11.0±0.1	11.7±0.2	RDED	**51.6±0.2**	58.8±0.4	60.2±0.3
Coreset Selection	Random	35.5±0.3	68.2±0.2	75.4±0.1	Random	35.0±0.3	58.3±0.2	63.9±0.1
	Herding	26.8±0.5	45.4±0.2	43.1±0.5	Herding	36.5±0.2	58.9±0.3	63.7±0.4
	K-Means	34.0±0.1	71.6±0.2	77.8±0.1	K-Means	35.2±0.4	58.7±0.2	64.3±0.1
	GraphCut	50.1±0.2	70.3±0.1	75.0±0.2	GraphCut	36.4±0.3	59.4±0.1	64.2±0.3
	GradMatch	32.6±0.5	63.6±0.3	71.5±0.2	GradMatch	36.1±0.5	58.9±0.4	63.8±0.5
	Glister	38.1±0.4	58.0±0.5	67.8±0.3	Glister	35.8±0.4	58.9±0.2	64.0±0.4
	Forgetting	30.6±0.5	62.9±0.4	69.4±0.2	Forgetting	36.9±0.3	58.6±0.2	64.0±0.2
	EL2N	34.5±0.4	59.0±0.5	65.5±0.3	EL2N	36.3±0.2	59.0±0.1	64.2±0.2
	DQ	34.9±0.3	67.1±0.2	74.6±0.2	DQ	35.7±0.3	58.5±0.2	63.6±0.3
	CCS	51.4±0.2	72.8±0.1	78.6±0.1	CCS	36.8±0.2	58.1±0.1	63.4±0.1
	DUAL	28.4±0.4	50.7±0.3	58.3±0.3	DUAL	34.1±0.4	57.6±0.3	62.8±0.3
	ADQ	34.7±0.3	66.9±0.2	74.8±0.2	ADQ	34.6±0.3	57.9±0.2	63.4±0.2
	DQAS	35.1±0.3	66.8±0.3	73.9±0.2	DQAS	34.8±0.3	58.0±0.2	63.5±0.3
Ours	**MCCS**	**52.9±0.2**	**74.4±0.1**	**80.3±0.1**	**MCCS**	37.2±0.2	**59.7±0.1**	**64.5±0.1**

Table 13: Cross-architecture evaluation accuracy on CIFAR-10 and ImageNet-1K. We report transfer performance from the source architecture used for selection to the target architecture used for evaluation. Values are mean ± standard deviation. Best results are **bold**; second-best are underlined.

	Arch 1 → Arch 2			Arch 2 → Arch 1		
Method	1 IPC	10 IPC	50 IPC	1 IPC	10 IPC	50 IPC
CIFAR-10 (Arch 1: MLP, Arch 2: ResNet-18)						
CCS	18.22±0.8	33.89±1.1	53.95±0.6	20.33±0.7	**28.31±0.9**	31.72±0.5
DQ	19.70±0.9	33.27±0.7	51.35±0.5	18.64±0.6	25.79±0.8	37.36±0.4
GC	23.55±0.6	38.86±0.8	49.83±0.4	19.02±1.0	26.94±0.7	35.09±0.6
MCCS	**23.90±0.4**	**42.10±0.5**	**54.04±0.3**	**20.44±0.5**	27.12±0.6	**37.42±0.4**
ImageNet-1K (Arch 1: ViT, Arch 2: ResNet-18)						
CCS	1.04±0.2	7.85±0.4	30.01±0.6	0.84±0.1	4.39±0.3	15.99±0.5
DQ	1.38±0.2	9.27±0.5	31.80±0.7	0.97±0.2	5.83±0.4	17.92±0.6
GC	1.61±0.3	9.65±0.4	31.39±0.5	1.06±0.2	6.10±0.3	18.04±0.5
MCCS	**1.80±0.2**	**9.90±0.3**	**36.10±0.4**	**1.24±0.1**	**6.28±0.2**	**18.71±0.4**

ward without combining or synthesizing images. These results show that MCCS selects real samples that are representative and provides sufficient variation across downstream evaluation settings.

G TRANSFERABILITY ACROSS ARCHITECTURES.

Table 13 reports bidirectional transfer between MLP and ResNet-18 on CIFAR-10 and between ViT-Tiny (Dosovitskiy et al., 2020) and ResNet-18 on ImageNet-1K. We retain the same number of selection stages T across architectures without architecture-specific tuning.

MCCS achieves the best performance in all but one configuration. When using MLP or ViT-Tiny for selection and ResNet-18 for evaluation, MCCS outperforms all baselines across every budget. On ImageNet-1K, MCCS achieves 36.10% at 50 IPC under ViT-to-ResNet transfer, exceeding the strongest baseline by 4.3%pp. MCCS maintains comparable advantages in the reverse direction when we select with ResNet-18 and evaluate with MLP or ViT-Tiny. These results indicate that MCCS is able to capture class-discriminative and complementary information that remains effective across architectures, rather than overfitting the selection criterion to the inductive bias of a specific model family.

Table 14: Performance comparison between standard MCCS, MCCS utilizing an ImageNet adapted DINO teacher for the first stage selection, and an early stop configuration. Values are presented as mean ± standard deviation.

	CIFAR-10					ImageNet-1K				
Method	1 IPC	10 IPC	50 IPC	100 IPC	200 IPC	1 IPC	10 IPC	50 IPC	100 IPC	200 IPC
MCCS	$24.0_{\pm 0.6}$	$43.5_{\pm 0.7}$	$63.2_{\pm 0.4}$	$70.8_{\pm 0.4}$	$77.5_{\pm 0.3}$	$2.1_{\pm 0.5}$	$17.9_{\pm 0.7}$	$42.3_{\pm 0.5}$	$51.2_{\pm 0.2}$	$59.8_{\pm 0.6}$
MCCS (DINO)	$22.7_{\pm 0.5}$	$40.3_{\pm 0.6}$	$61.6_{\pm 0.5}$	$70.1_{\pm 0.4}$	$76.9_{\pm 0.3}$	$1.3_{\pm 0.2}$	$10.2_{\pm 0.7}$	$38.7_{\pm 0.6}$	$48.9_{\pm 0.3}$	$57.9_{\pm 0.5}$
MCCS (Early stop)	$23.5_{\pm 0.5}$	$43.0_{\pm 0.6}$	$62.9_{\pm 0.4}$	$70.7_{\pm 0.4}$	$77.3_{\pm 0.3}$	$2.1_{\pm 0.5}$	$17.9_{\pm 0.7}$	$41.2_{\pm 0.3}$	$50.4_{\pm 0.4}$	$58.9_{\pm 0.5}$

H Teacher Model Variations

MCCS uses a model trained on the full dataset to identify prototypical samples during initialization. To reduce this initial cost, we evaluate two alternatives: training the teacher for half of the standard epochs and using a pretrained DINO model (Oquab et al., 2023) adapted to each dataset. Table 14 compares these configurations with the standard MCCS teacher.

Both alternatives retain most of the performance of standard MCCS. At 200 IPC, early stopping achieves 77.3% on CIFAR-10 and 58.9% on ImageNet-1K, corresponding to decreases of only 0.2%pp and 0.9%pp. The DINO teacher achieves 76.9% and 57.9%, respectively. These results show that MCCS can use partially trained or pretrained models for initialization without substantially reducing coreset quality. Moreover, the initial teacher constitutes a one-time cost, and its scores can be reused across coreset budgets.