ABAS-RAL: ADAPTIVE BATCH SIZE USING REINFORCED ACTIVE LEARNING

Lazaridis Ioannis *Psaltis Athanasios* *Dimou Anastasios* *Daras Petros*

Centre for Research and Technology Hellas, Thessaloniki, Greece

ABSTRACT

Active learning reduces annotation costs by prioritizing the most informative samples, however, methods that are using fixed batch sizes can be inefficient as they fail to adjust to variations in data and resource limitations. This challenge becomes even more critical in resource-constraint environments, such as IoT and edge devices, where both computational power and the availability of labeled data for training are limited. We propose Adaptive Batch Size using Reinforced Active Learning, a novel method that employs a dual-strategy design to dynamically optimize batch sizes. Firstly, the proposed method identifies performance benchmarks, including target precision and annotation budget thresholds, by exploring various batch sizes during an initialization phase. These benchmarks guide the second phase, where a reinforcement learning agent adjusts batch sizes in real-time, ensuring efficient resource usage while maintaining robust model performance. Evaluated on CIFAR-10, CIFAR-100, and MNIST datasets, the proposed method demonstrates significant annotation cost reductions and efficient learning, even in scenarios with constrained computational resources and limited labeled data.

***Index Terms*—** Reinforcement Learning, Active Learning, Batch Size Optimization

1. INTRODUCTION

Active Learning (AL) is an approach that reduces annotation costs by prioritizing the selection of informative samples. However, it often becomes resource-intensive when applied to large datasets or tasks requiring extensive annotations [1]. This challenge is particularly pronounced in low-resource environments, such as edge devices, where computational and annotation budgets are severely constrained [2]. Existing AL methods, including both uncertainty-based [3] and hybrid approaches [4], aim to maximize learning efficiency by selecting samples with high information gain. Techniques such as ALFA-Mix [5] attempt to balance annotation cost and variability but often overlook the practical constraints faced in resource-limited environments.

Reinforcement Learning (RL) provides a dynamic framework to optimize AL strategies and is effective in constrained scenarios [6]. Despite its potential, RL has not been fully utilized for dynamically optimizing batch sizes in AL. Reinforced Active Learning (RAL) utilizes RL to enable dynamic, context-aware sample selection and has demonstrated versatility across fields like image classification [7], and medical imaging [8].

Fixed-batch size methods select a constant number of samples per iteration, often resulting in inefficiencies—either underusing data, missing learning opportunities, or overusing resources, leading to wasted computation and time. Additionally, they struggle to adapt to varying data complexity, reducing the effectiveness of annotation budgets. In contrast, adaptive batch size optimization, driven by the learning state and resource constraints, overcomes these challenges by improving both efficiency and scalability. Specifically, those approaches [9, 10, 11] typically rely on fixed rules or heuristics, which limit adaptability and require manual tuning. More flexible methods, such as dynamic batch size scaling for improved convergence [12], adaptive AL strategies for sample selection [13, 14, 15], and probabilistic frameworks for batch size optimization [16], address some limitations, but fail to fully balance computational constraints and annotation precision in low-resource settings. Despite its potential, RAL for optimizing batch size remains underexplored.

To address the aforementioned challenges, we propose ABAS-RAL, a novel method that utilizes RAL to efficiently adjust batch sizes, enhancing adaptability and performance during the annotation - particularly in scenarios such as online adaptation, semi-supervised learning, or privacy-preserving settings on edge devices, where local training and annotation minimization are essential and traditional server-based pipelines with distillation may not be feasible. By modeling this process as a Markov Decision Process (MDP), ABAS-RAL learns an optimal policy that balances precision and budget while eliminating the need for manual tuning.

The proposed approach uses a Deep Q-Network (DQN) [17] to efficiently allocate annotation efforts, ensuring that resources are used optimally. The latter is achieved by a) **optimizing resource usage**, dynamically focusing annotation on impactful regions to reduce computational overhead; b) **reducing annotation costs**, prioritizing the selection of informative and uncertain samples to minimize labeling while limiting batch sizes to conserve resources; and c) **maintaining robust performance**, utilizing precision and budget thresholds set during initialization to meet target metrics under con-

straints. RL-driven adjustments enable ABAS-RAL to balance efficiency and feasibility, making it ideal for resource-limited devices.

2. METHODOLOGY

The RAL-based approach uses an agent with a dual strategy to select the optimal number of samples to annotate during an AL episode. The agent operates within a defined state-space, choosing actions (batch sizes) and receiving rewards based on improvements in model performance. By focusing on informative samples, this method reduces the number of annotations needed to achieve satisfactory performance.

2.1. Formulating AL as an MDP

An MDP consists of five components [18]: the **agent** (the decision-maker), the **environment** (the context in which the agent operates), the **state** (the current environment condition), the **action** (the choices available to the agent), and the **reward** (the feedback on action quality). The **policy** defines the strategy to map states to actions and to maximize cumulative rewards. In AL, an episode involves selection and annotation of a batch of samples and then retraining of the classifier so as to assess performance improvements.

Following Konyushkova et al. [10], ABAS-RAL uses a **DQN agent** that learns an optimal **policy** to select batch sizes in each AL iteration. The **environment** includes the unlabeled dataset and the classifier being trained. The **state** is defined by the margin scores of a selected subset of samples, reflecting the uncertainty of the model, and is updated after each iteration based on the re-trained classifier. The **action** is the selection of the batch size, and the **reward** evaluates the model's precision improvement relative to annotation cost, guiding the agent to prioritize informative samples.

ABAS-RAL optimizes batch selection in AL through a structured approach involving **performance benchmarking**, and **agent training**. **Performance benchmarking** acts as a *warm-start phase* in RL, where different batch sizes are tested not only to provide the agent with an overview of their effects, but also to identify target precision and budget thresholds. Finally, the **agent is trained** using a DQN to select batch sizes that optimize both performance and resource use, enabling ABAS-RAL to adapt dynamically to learning progress and constraints.

2.1.1. RL states

Initially, a subset V is randomly sampled from the unlabeled pool U_0 to define the state. The remaining unlabeled data is updated as $U = U_0 \setminus V$. At each iteration t of an AL episode, the classifier is retrained with the newly annotated batch of samples chosen from the unlabeled dataset. The state is represented by a vector $\mathbf{s}_t$ of sorted margin scores $m(x_i)$ for each $x_i \in V$, which are computed based on the classifier's predictions as the difference between the highest and the second-highest predicted class probabilities $m_i(x_i) = P(y^*|x_i) - P(y^{**}|x_i)$, reflecting prediction uncertainty. Sorting ensures structured input, allowing the RL agent to identify uncertain samples effectively. Margin scores are computationally efficient and provide a simple way to prioritize uncertain samples.

2.1.2. RL action selection

The DQN agent selects actions (batch sizes) that maximize the Q-value (1), guiding the agent to choose batch sizes that yield the highest cumulative reward. By updating Q-values based on received rewards, the DQN agent optimizes the annotation process for efficient learning. At each iteration t of an AL episode, the action a_t is defined as selecting k samples for annotation, where $k \in \{1, \ldots, \lfloor 0.1 \times |U_t| \rfloor\}$, and $|U_t|$ is the size of the unlabeled dataset. Limiting the action space to 10% of the dataset per iteration encourages thorough data exploration, prevents overfitting, and ensures efficient annotation. The Q-value at iteration t for a given state s_t and action a_t is computed using the following equation:

$$Q(s_t, \mathrm{a_t}) = \mathrm{r_t} + \gamma \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \tag{1}$$

where r_t is the immediate reward received after taking action a_t in state s_t, γ is the discount factor, which determines the importance of future rewards, and $\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$ is the maximum Q-value of the possible actions a_{t+1} in the next state s_{t+1}.

2.1.3. RL Rewards Calculation

The reward r_t for action a_t at iteration t is defined as:

$$r_t = \frac{P(B_{t+1})}{B_{t+1}} - \frac{P(B_t)}{B_t} \tag{2}$$

where B_t denotes the batch size used in iteration t and $P(B_t)$ the corresponding precision achieved by the classifier. This formulation measures the precision improvement per annotated sample, encouraging the agent to favor batch sizes that yield more efficient learning gains relative to annotation cost. For example, if precision increases from 0.50 to 0.55 when using batches of 100 and 10 samples respectively, the reward becomes 0.05, indicating that the smaller batch achieves a significantly higher precision gain per annotated sample. Conversely, if precision improves from 0.40 to 0.45 when increasing the batch size from 10 to 15 samples, the reward is -0.01, suggesting that the larger batch is less efficient despite the absolute precision improvement. This encourages efficiency while avoiding trivial batch sizes, as the DQN learns that larger, more informative batches are necessary to achieve the target precision and maximize cumulative rewards.

2.2. Performance Benchmarking (Initialization Phase)

According to Algorithm 1, the benchmarking phase is a restructured version of the *warm-start episodes* in RL. Traditionally, warm-start episodes involve testing random actions to observe how the environment responds. In our method, this phase is expanded to serve an additional purpose: defining the target precision and annotation budget in a way that reflects the capabilities of the device being used. By systematically testing various batch sizes from the unlabeled dataset, and using the classifier to compute precision, the benchmarking phase evaluates the trade-offs between precision and cost, ensuring efficient learning without overburdening the available resources. An episode terminates when two consecutive rewards decrease, signaling that the model is no longer improving, thus avoiding ineffective batches. In conclusion, this phase $a)$ stores **state-action-reward** transitions into the replay buffer, which provides a rich set of experiences to guide the agent's training and decision-making process, and $b)$ establishes meaningful performance benchmarks.

Algorithm 1 Performance Benchmarking Phase

Require: U_0 initial unlabeled data, L labeled data, U unlabeled data, D training set, a batch, c classifier, r reward, s state, E episodes number
Ensure: Classifier c, Target precision P_{target}, Target budget B_{target}
Sample a subset V to define the state data from the unlabeled data U_0.
Update unlabeled set $U = U_0 \setminus V$.
for $e = 0$ to $E - 1$ **do**
  Initialize the labeled data L and the unlabeled data U.
  repeat
    Randomly choose a number a of X samples to annotate from U.
    Retrieve labels Y for X from D.
    Update labeled set $L_{new} = L_{previous} + X$.
    Update unlabeled set $U_{new} = U_{previous} - X$.
    Re-train c with L_{new}.
    Compute the state s (margin scores for the V data) using c.
    Compute the reward r based on the precision of the c and the selected a size.
    Store transition (s, a, r) in the replay buffer.
  until two consecutive rewards decrease **or** the samples are exhausted.
end for
P_{target} is the maximum achieved precision.
B_{target} is the minimum budget.

return $c, P_{target}, B_{target}$

2.3. Training the DQN Agent

After setting performance benchmarks, the DQN agent dynamically selects batch sizes at each iteration according to Algorithm 2. The agent aims to maximize long-term reward by optimizing the precision and annotation budget based on state and rewards. It uses the transitions stored in the replay buffer to learn from past experiences and updates it with new ones. Initially, the agent selects random actions for *exploration*, but as training progresses, it shifts towards *exploitation*, relying more on the learned policy to maximize rewards and optimize decisions. The Q-network in ABAS-RAL approximates the Q-values to predict future rewards for batch sizes according to (1). It consists of an input layer for state data (margin scores), a fully connected layer (10 units, sigmoid), and a state-action concatenation step. A second layer (5 units, sigmoid) refines the representation, and the output layer generates Q-values for each batch size. The agent selects the optimal batch size based on the highest Q-value. The DQN architecture was intentionally kept minimal (10/5 units in the hidden layers) to ensure a negligible computational footprint. In our experiments, the inference time for the DQN agent was less than 1 ms per iteration, making it suitable for real-time execution on low-power edge devices without offsetting the annotation cost savings.

Figure 1 illustrates the ABAS-RAL process: the dataset is split into labeled and unlabeled data. The RL agent selects batch sizes based on state, the classifier annotates uncertain samples, and rewards are calculated based on (2). The agent updates its policy to optimize batch selection, iterating until precision and cost are balanced.

Algorithm 2 Agent Training Phase

Require: L labeled data, U unlabeled data, D training set, a batch, c classifier, r reward, s state, E agent's training epochs, T episodes per epoch
Ensure: Trained agent DQN_T, Trained classifier c_T
Start with the pre-trained classifier c.
for $e = 0$ to $E - 1$ **do**
  Start with the labeled L and the unlabeled U data.
  for $t = 0$ to $T - 1$ **do**
    Train classifier c_t using labeled data L_t.
    repeat
      Compute margin scores for samples in V using c_t.
      Sort margin scores to form state s_t.
      Select action a_t (batch size) using DQN policy.
      Compute $a_t \leftarrow \arg\max_a Q(s_t, a)$.
      Find the batch X_t by selecting a_t samples with highest uncertainty based on the c_t.
      Retrieve labels Y_t for X_t from D.
      Update labeled set $L_t \leftarrow L_t \cup (X_t, Y_t)$.
      Update unlabeled set $U_t \leftarrow U_t \setminus X_t$.
      Train c_t using labeled data L_t.
      Calculate reward r_t. {Precision gain per annotation.}
      Store transition (s_t, a_t, r_t) in the replay buffer.
      Update agent's DQN_t weights.
    until Precision $P_t \geq P_{target}$ **and** Budget $B_t \leq B_{target}$ **or** the samples are exhausted.
  end for
end for

return DQN_T, c_T

3. EXPERIMENTAL EVALUATION

Experiments are conducted on CIFAR-10, CIFAR-100 [19], and MNIST [20], with each dataset split into 10% for state data, 10% for warm-start data, 60% for DQN training, and 20% for evaluation. Evaluation metrics include precision, accuracy, recall, F1-score, and annotation budget. We compare ABAS-RAL with existing methods using fixed and non-fixed batch sizes under various conditions.

3.1. Comparison with Fixed-batch Size Methods

For this set of experiments, we use a ResNet50 model, as classifier, pre-trained on ImageNet [21] (TensorFlow 1.14, Python 3.6) to fine-tune on CIFAR-10, CIFAR-100, and MNIST. To

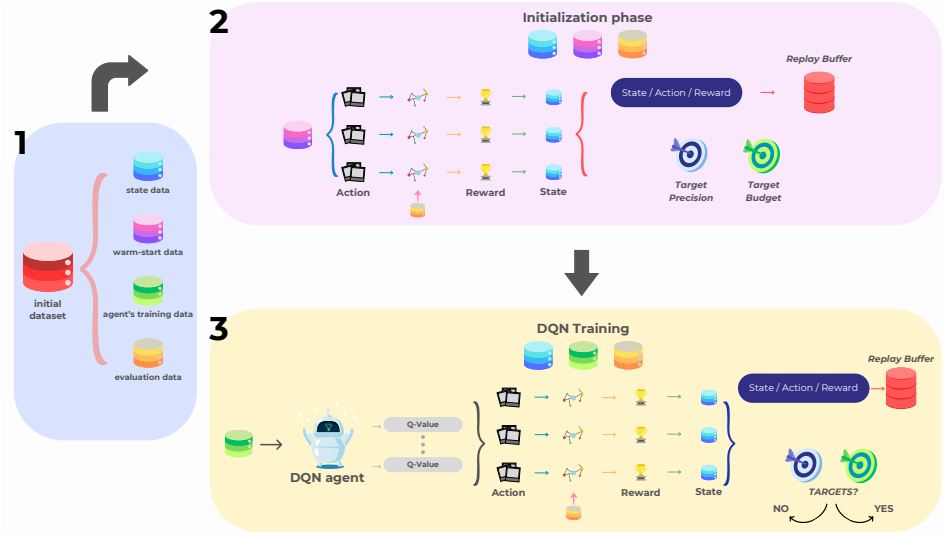

Fig. 1. Overview of the ABAS-RAL process for optimizing batch size selection: (1) Initial dataset is split into sub-datasets; (2) Initialization Phase benchmarks precision and annotation cost by testing batch sizes and storing experiences for efficient learning; (3) Training Phase employs a DQN agent that dynamically selects batch sizes to maximize long-term rewards based on learned Q-values and past experiences.

Table 1. Maximum precision achieved by the ResNet50 model after training for 10 epochs on the full training datasets.

Dataset	Maximum Precision Achieved
CIFAR-10	45%
CIFAR-100	29%
MNIST	95%

evaluate ABAS-RAL in stringent resource-constrained settings, we established an upper-bound performance baseline by training the ResNet50 classifier for only 10 epochs with a frozen backbone. As shown in Table 1, these baseline precisions (45%, 29%, and 95% for CIFAR-10, CIFAR-100, and MNIST, respectively) represent the full-information convergence targets for our experiments. Our goal is to demonstrate how ABAS-RAL can reach these deployment-specific targets with minimal annotation budget. All experiments run on an NVIDIA GTX 1080 Ti GPU.

The fixed-batch size methods include Random Sampling (RS) [22], Entropy Sampling (ES) [23], Uncertainty Sampling (US) [24], and ALFA-Mix [5]. ABAS-RAL consistently outperforms these methods across all datasets. Table 2 shows that, on CIFAR-10, ABAS-RAL achieves 46.17% precision with only 5.12% of the budget, while the other methods achieve precision with budgets ranging from 27% to 97.49%. On CIFAR-100, ABAS-RAL uses a mere 1.98% of the budget, and on MNIST, ABAS-RAL achieves 95.18% precision while using just 7.07% of the budget.

Classifiers were also trained from scratch to reach 85% of the target precision. As shown in Table 3, ABAS-RAL maintains its superiority; on CIFAR-10, it hits 39.69% precision

Table 2. Comparison with fixed-batch methods (mean of 25 runs) using the pretrained classifier. Target Precision and Budget are set during Performance Benchmarking. ABAS-RAL achieves the targets with low variance ($\sigma < 0.75\%$).

		Precision	Accuracy	Recall	F1-Score	Budget
CIFAR10 — *Target Precision: 44.66%, Target Budget: 98.03%*						
ABAS-RAL	**Mean**	**46.17%**	**43.99%**	**43.97%**	**43.67%**	**5.12%**
	Max	**50.00%**	**45.26%**	**45.26%**	**45.24%**	**11.17%**
RS [22]	**Mean**	45.89%	42.00%	43.00%	41.97%	50.79%
	Max	46.85%	43.62%	43.62%	44.08%	97.49%
ES [23]	**Mean**	45.04%	42.96%	42.95%	42.17%	30%
	Max	47.36%	44.14%	44.13%	43.67%	35%
US [24]	**Mean**	46.04%	43.96%	43.96%	43.17%	28%
	Max	48.36%	45.14%	45.14%	44.67%	32%
ALFA-Mix [5]	**Mean**	45.67%	43.54%	43.92%	42.08%	27%
	Max	47.44%	44.56%	45.13%	44.32%	38%
CIFAR100 — *Target Precision: 28.01%, Target Budget: 99.94%*						
		Precision	**Accuracy**	**Recall**	**F1-Score**	**Budget**
ABAS-RAL	**Mean**	**32.02%**	**32.14%**	**32.14%**	**31.54%**	**1.98%**
	Max	**32.52%**	**32.53%**	**32.55%**	**32.53%**	**1.98%**
RS	**Mean**	31.89%	32.02%	32.07%	31.48%	48.71%
	Max	32.47%	32.07%	33.02%	32.21%	99.22%
ES	**Mean**	31.64%	32.02%	32.01%	31.23%	25%
	Max	32.41%	32.11%	32.13%	31.43%	27%
US	**Mean**	31.73%	32.05%	32.05%	31.31%	37%
	Max	32.05%	32.30%	32.31%	31.53%	31%
ALFA-Mix	**Mean**	31.82%	32.09%	32.09%	31.53%	34%
	Max	32.43%	32.47%	32.47%	31.57%	30%
MNIST — *Target Precision: 94.11%, Target Budget: 96.34%*						
		Precision	**Accuracy**	**Recall**	**F1-Score**	**Budget**
ABAS-RAL	**Mean**	**95.18%**	**95.05%**	**94.98%**	**95.04%**	**7.07%**
	Max	**95.23%**	**95.24%**	**95.24%**	**95.23%**	**8.18%**
RS	**Mean**	95.02%	95.01%	94.90%	95.02%	60.13%
	Max	95.21%	95.20%	95.20%	95.19%	95.15%
ES	**Mean**	95.10%	94.09%	94.95%	94.99%	11%
	Max	95.22%	95.17%	95.23%	95.20%	13%
US	**Mean**	95.12%	94.13%	94.89%	94.98%	15%
	Max	95.22%	95.15%	95.20%	95.20%	18%
ALFA-Mix	**Mean**	95.01%	94.97%	94.79%	94.95%	16%
	Max	95.10%	95.10%	95.08%	95.08%	18%

using only 7.74% budget. Similar efficiency is observed on CIFAR-100 (2.98% budget) and MNIST (26.67% budget) to

Table 3. Comparison of ABAS-RAL (mean of 25 runs) with fixed-batch size methods when training the classifier from scratch. Target Precision being defined during the Performance Benchmarking phase. ABAS-RAL achieves the targets with low variance ($\sigma < 0.8\%$).

		Precision	Accuracy	Recall	F1-Score	Budget
CIFAR10 — *Target Precision: 37.96%*						
ABAS-RAL	**Mean**	**39.69%**	**36.17%**	**36.17%**	**34.52%**	**7.74%**
	Max	**44.73%**	**39.53%**	**39.53%**	**38.46%**	**27.67%**
ES [23]	**Mean**	35.30%	28.35%	28.24%	26.08%	87%
	Max	43.35%	32.96%	32.96%	32.44%	90%
US [24]	**Mean**	36.30%	29.35%	29.24%	27.08%	88%
	Max	44.35%	33.96%	33.96%	33.43%	96%
ALFA-MIX [5]	**Mean**	34.30%	27.35%	29.24%	27.24%	85%
	Max	42.35%	31.96%	31.44%	31.96%	87%
CIFAR100 — *Target Precision: 23.81%*						
		Precision	**Accuracy**	**Recall**	**F1-Score**	**Budget**
ABAS-RAL	**Mean**	**25.60%**	**25.73%**	**25.73%**	**24.57%**	**2.98%**
	Max	**27.20%**	**27.17%**	**27.17%**	**26.11%**	**2.98%**
ES	**Mean**	24.11%	25.14%	24.65%	23.08%	70%
	Max	25.43%	25.96%	26.12%	23.44%	92%
US	**Mean**	24.12%	25.14%	24.71%	23.23%	71%
	Max	25.50%	26.65%	24.91%	23.43%	91%
ALFA-MIX	**Mean**	22.55%	23.84%	23.34%	21.98%	61%
	Max	23.35%	23.96%	23.84%	22.12%	72%
MNIST — *Target Precision: 79.99%*						
		Precision	**Accuracy**	**Recall**	**F1-Score**	**Budget**
ABAS-RAL	**Mean**	**81.85%**	**81.65%**	**80.97%**	**80.91%**	**26.67%**
	Max	**82.97%**	**82.91%**	**81.25%**	**81.24%**	**29.18%**
ES	**Mean**	80.87%	79.23%	79.02%	78.01%	34.10 %
	Max	81.23%	79.46%	79.52%	79.06%	37.10 %
US	**Mean**	80.20%	79.30%	79.50%	78.90%	32.10%
	Max	81.00%	80.00%	80.30%	79.70%	39.10%
ALFA-MIX	**Mean**	80.21%	80.71%	80.71%	80.47%	40.32%
	Max	81.42%	81.40%	81.11%	81.21%	41.40%

Table 4. Training time breakdown for ABAS-RAL on CIFAR-10.

Phase	Time
Initialization Phase	$\sim$6 hours
DQN Agent's Training	$\sim$12 hours
Mean Total Run Duration	$\sim$407 seconds

reach the 25.60% and 81.85% precision marks, respectively.

These results underscore ABAS-RAL's exceptional efficiency, delivering high performance while dramatically reducing annotation costs, thereby enabling more effective use of resources and maximizing the impact of each annotated sample.

Table 4 shows the time breakdown for the ABAS-RAL training process on the CIFAR-10 dataset. It includes the duration of the initialization phase, the agent's training, and the mean time per experiment run across 25 runs. The total training time was approximately 18 hours, including the training of the DQN agent. However, this is a one-time cost, as the trained RL agent can be reused across devices without retraining. The mean total duration per experiment ($\sim$407 seconds) is almost entirely dominated by the classifier's retraining, as the DQN agent's decision-making overhead remains negligible ($<$1 ms).

3.2. Comparison with non-Fixed batch Size Methods

We evaluate ABAS-RAL against DRAL [14] and LAL-RL [10] on the CIFAR-10 dataset using ResNet-18 [25] to demonstrate the benefits of dynamic batch-size optimization through RL in a commonly used setting. Table 5 shows a significant reduction in annotation cost, with ABAS-RAL requiring approximately 25% fewer samples at 50% target accuracy and more than 70% fewer samples at 60% target accuracy compared to DRAL. This performance gain highlights the adaptability of ABAS-RAL in optimizing annotation effort by focusing on sample informativeness and uncertainty while balancing resource constraints.

Table 5. Number of labeled samples required to reach target accuracy on CIFAR-10 using ResNet-18 for ABAS-RAL, DRAL, and LAL-RL.

Target ACC	Method	Samples Used
50%	DRAL [14]	$\sim$1000
	LAL-RL [10]	*Not completed*
	ABAS-RAL	$\sim$**750**
60%	DRAL	$\sim$3000
	LAL-RL	*Not completed*
	ABAS-RAL	$\sim$**850**

LAL-RL [10], originally a single-sample binary selector, proved inefficient for image classification. Unlike our batch-optimized approach, LAL-RL's lack of batch-wise optimization resulted in excessive computational overhead, failing to reach target accuracies even after 3 days of training.

4. CONCLUSION AND FUTURE WORK

We present ABAS-RAL, an RL-based AL method that optimizes the balance between precision and annotation costs. Experimental results across multiple datasets demonstrate that ABAS-RAL consistently outperforms fixed and non-fixed batch methods, achieving superior performance with significantly fewer labeled samples. While current evaluations focus on standard datasets (CIFAR, MNIST), the agnostic nature of the margin-based state representation suggests scalability to more complex tasks. Future work will extend ABAS-RAL to high-resolution tasks, investigate alternative RL architectures, and evaluate scalability across diverse hardware and large-scale datasets.

Acknowledgements. This work is part of the TESTUDO project, which has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101121258. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

5. REFERENCES

[1] Xueying Zhan, Qingzhong Wang, Kuan hao Huang, Haoyi Xiong, Dejing Dou, and Antoni B Chan, "A comparative survey of deep active learning," *arXiv:2203.13450*, 2022.

[2] J. Rafid Mahmood, James Lucas, Jose M Alvarez, Sanja Fidler, and Marc T Law, "Optimising data collection for machine learning," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[3] Rebati Gaire and Arman Roohi, "Fdal: Leveraging feature distillation for efficient and task-aware active learning," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2025.

[4] Maite Puerta-Beldarrain, Oihane Gómez-Carmona, Diego Casado-Mansilla Rubén Sánchez-Corcuera, Diego López de-Ipiña González de Artaza, and Liming Chen, "A multifaceted vision of the human-ai collaboration: A comprehensive review," *IEEE Access*, 2025.

[5] Amin Parvaneh, Ehsan Abbasnejad, Damien Teney, Gholamreza Reza Haffari, Anton Van Den Hengel, and Javen Qinfeng Shi, "Active learning by feature mixing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[6] Rinyoichi Takezoe, Xu Liu, Shunan Mao, Marco Tianyu Chen, Zhanpeng Feng, Shiliang Zhang, and Xiaoyu Wang, "Deep active learning for computer vision: Past and future," *APSIPA Transactions on Signal and Information Processing.*, 2023.

[7] Qing Cai, Ran Tao, Xiufen Fang, Xiurui Xie, and Guisong Liu, "A deep reinforcement active learning method for multi-label image classification," in *Computer Vision and Image Understanding*, 2025.

[8] Masuda Begum Sampa, Nor Hidayati Abdul Aziz, Md Siddikur Rahman, Nor Azlina Ab Aziz, Rosli Besar, and Anith Khairunnisa Ghazali, "Reinforcement learning for medical image analysis: a systematic review of algorithms, engineering challenges, and clinical deployment," *Computer Assisted Surgery*, 2026.

[9] Guy Hacohen, Avihu Dekel, and Daphna Weinshall, "Active learning on a budget: Opposite strategies suit high and low budgets," *arXiv preprint arXiv:2202.02794*, 2022.

[10] Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua, "Discovering general-purpose active learning strategies," *arXiv preprint arXiv:1810.04114*, 2018.

[11] Nikos Fazakis, Vasileios G. Kanas, Christos K. Aridas, Stamatis Karlos, and Sotiris Kotsiantis, "Combination of active learning and semi-supervised learning under a self-training scheme," *Entropy*, 2019.

[12] Aditya Devarakonda, Maxim Naumov, and Michael Garland, "Adabatch: Adaptive batch sizes for training deep neural networks," *CoRR*, 2017.

[13] Zhenghao Zhao, Yuzhang Shang, Junyi Wu, and Yan Yan, "Dataset quantization with active learning based adaptive sampling," in *ECCV*, 2024.

[14] Mingyuan Jiu, Xuguang Song, Hichem Sahbi, Shupan Li, Yan Chen, Wei Guo, Lihua Guo, and Mingliang Xu, "Image classification with deep reinforcement active learning," in *Computer Science*, 2024.

[15] Qing Cai, Ran Tao, Xiufen Fang, Xiurui Xie, and Guisong Liu, "A deep reinforcement active learning method for multi-label image classification," *Computer Vision and Image Understanding*, 2025.

[16] Masaki Adachi, Satoshi Hayakawa, Martin Jørgensen, Xingchen Wan, Vu Nguyen, Harald Oberhauser, and Michael A Osborne, "Adaptive batch sizes for active learning: A probabilistic numerics approach," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[18] Kai Xie and Attila Szolnoki, "Reinforcement learning in evolutionary game theory: A brief review of recent developments," *Applied Mathematics and Computation*, 2026.

[19] Krizhevsky Alex and Geoffrey Hinton, "Learning multiple layers of features from tiny images," *Technical Report, University of Toronto*, 2009.

[20] Y. Lecun, Bottou, L., Bengio, Y., and Haffner P., "Gradient-based learning applied to document recognition," in *IEEE*, 1998.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the 2009 IEEE conference on computer vision and pattern recognition*, 2009.

[22] William Gemmell Cochran, *Sampling Techniques*, John Wiley & Sons, 1977.

[23] Claude Elwood Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, 1948.

[24] David D. Lewis and William A. Gale, "A sequential algorithm for training text classifiers," in *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and development in Information Retrieval*, 1994.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.