TiCRL: Textual Image Classification with Reinforcement Learning-Based Curriculum Learning

Gayoung Kim[1] and Yuncheol Kang[2]*

[1] Graduate School of Big Data Analytics, Ewha Womans University, Seoul, 03760 South Korea
[2] School of Business, Ewha Womans University, Seoul, 03760 South Korea
gy.kim@ewha.ac.kr, yckang@ewha.ac.kr

Abstract. Textual image classification remains challenging due to the diversity of document layouts and textual structures. We propose TiCRL (Textual Image Classification with Reinforcement Learning-based Curriculum Learning), which combines a text-independent difficulty measurer with an RL-based training scheduler. The measurer estimates sample difficulty using visual-textual features and loss signals without requiring OCR, while the RL agent dynamically adapts the curriculum based on the learner's state. TiCRL substantially improves both accuracy and data efficiency. On a lightweight CNN trained from scratch, it achieves 83.67% accuracy using only 82% of the training data, outperforming curriculum learning baselines by up to 13.8%p. Applied to a strong transformer baseline (DiT), TiCRL reaches 93.02% accuracy using only 58% of the data, surpassing transformer baselines trained on 80%. The learned curriculum policy generalizes to new datasets and architectures without retraining, demonstrating that RL-based curriculum learning is an effective and efficient training strategy for textual image classification.

1 Introduction

As digital transformation accelerates, large volumes of paper-based documents are being digitized, making automated document classification essential for downstream tasks such as retrieval, information extraction, and text recognition [1, 4]. However, document images created through scanning or photography vary widely—from structured contracts to unstructured free-form layouts—making classification challenging, especially when relying only on structural cues.

While traditional image-based classifiers such as Convolutional Neural Network (CNN) perform well on standardized layouts [16], their gains do not always transfer to diverse documents. Recent work addresses this through heavier architectures or multi-modal inputs [2, 12, 15, 23].

* Corresponding author

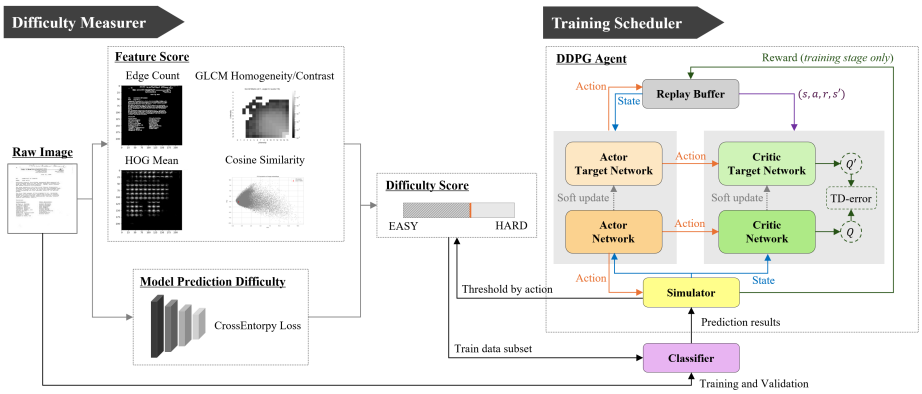

Fig. 1: Overview of the proposed TiCRL framework.

Our approach shifts the focus to the learning process itself without altering the backbone model or adding costly inputs. We hypothesize that a difficulty score built from visual-textual features and loss signals will raise classification accuracy. We further expect a Reinforcement Learning (RL)-based scheduler to outperform fixed curricula when both use the same difficulty metric. Finally, we anticipate that the learned policy will transfer to new textual image datasets without extra tuning. A key insight of our approach is that strategic selection of training samples—focusing on easier examples early—can improve generalization even on hard-to-classify categories, as the agent learns an adaptive curriculum that balances exploration and exploitation. To test these ideas, we adopt curriculum learning [3], which naturally fits document images where difficulty varies with structural formality and visual detail.

Curriculum learning (CL) depends on how difficulty is measured (a.k.a. difficulty measurer) and how the training schedule is controlled (a.k.a. training scheduler) [34]. For difficulty scoring, we design a text-independent hybrid measurer that combines visual-textual features with pre-trained model loss. For scheduling, we employ a learner-aware RL agent that dynamically adjusts the curriculum based on real-time training feedback. We call this framework **TiCRL** (Textual Image Classification with Reinforcement Learning based Curriculum Learning) (Figure 1).

The contributions of this study are summarized as follows:

- A TiCRL framework for textual image classification that tackles the accuracy–efficiency trade-off without modifying the base classifier, applicable to both from-scratch training and fine-tuning scenarios
- A text-independent difficulty measurer that captures structural layout complexity without OCR, combining hand-crafted visual-textual features with model-based signals, validated through comprehensive ablation study

- A learner-aware RL scheduler that dynamically adapts curriculum based on training feedback, surpassing predetermined schedules and generalizing across diverse architectures (CNNs and transformers) and unseen datasets
- An analysis of early stopping criteria for RL-driven curricula, including practical guidance on monitoring actor loss and Q-value stability

2 Related Work

Curriculum Learning was proposed by [3] to mimic human study habits, moving from easy to hard samples. It has improved convergence and generalization in computer vision, natural language processing, medical report generation, and graph tasks [13, 28, 34], with particular benefits under noisy or imbalanced data conditions [9]. Two pieces matter most, how difficulty is scored (difficulty measurer) and how the schedule is run (training scheduler). CL differs from static coreset selection (e.g., CRAIG [25], EL2N [26]), which selects a fixed subset prior to training and often requires per-backbone recomputation; CL methods, including TiCRL, instead govern the order and timing of sample introduction dynamically.

Difficulty scoring falls into prior knowledge and data-driven groups. Prior knowledge approaches define difficulty using domain-specific heuristics such as object size for segmentation [35]. Data-driven approaches rely on model-dependent signals: Self-Paced Learning (SPL) uses current model loss [20], CurriculumNet employs teacher network predictions to cluster data by confidence [8], MentorNet uses a separate mentor network to weight samples [14], and Data-IQ analyzes gradient-based importance scores [30]. While effective, these methods depend on model training state, requiring recomputation for different architectures. In contrast, our approach measures difficulty using text-independent visual-textual features that remain valid across architectures—from CNNs to transformers—without recomputation. Schedulers decide when harder material appears. Discrete rules switch at fixed epochs or after subset convergence, seen in Baby Step [31], One-Pass [3], and Lambda Step [20]. Continuous rules raise the used ratio with functions such as linear or root families [34]. Multi-task curriculum learning determines task ordering based on transfer relationships [27]. These preset schedules are easy to code yet often ignore task-specific signals. Automated CL reduces manual tuning by learning the scorer, the schedule, or both from feedback. Adaptive Curriculum Learning (ACL) [17] addresses this by combining dynamic difficulty assessment with pre-trained model knowledge, updating difficulty scores throughout training via knowledge distillation.

Recent work frames scheduling as a teacher–student problem in reinforcement learning [34]. A teacher watches the student model and picks data or tasks, then uses performance gains as reward. Learning to Teach cast this as a Markov Decision Process (MDP) where the teacher can also choose loss functions or learning rates [6]. RL has been used to balance skewed clinical data [5], optimize neural machine translation [18, 36], and order tasks in multitask settings [24], yielding faster and steadier training.

Despite these advances, textual image classification rarely employs CL. Prior efforts focus on model design or multi-modal fusion [2, 12, 15, 23]. Our work fills this gap with TiCRL, which differs from existing curriculum methods in two key ways: (1) our difficulty measurer uses text-independent visual-textual features rather than model-dependent signals (loss, predictions, gradients), enabling architecture-agnostic curriculum construction, and (2) our RL scheduler dynamically adapts to learner state rather than following fixed or teacher-driven schedules. The result is strong accuracy with moderate cost and transfer to unseen datasets without extra tuning.

3 Method

3.1 Overview of the Proposed Framework

To implement a dynamic curriculum learning strategy, we design our TiCRL framework with two primary components: **difficulty measurer** and **training scheduler**, as illustrated in Figure 1. The difficulty measurer computes sample-wise difficulty scores, while the training scheduler uses reinforcement learning to dynamically adjust the curriculum based on real-time performance feedback.

3.2 Visual-Textual Features-based Difficulty Measurer

We designed a difficulty score that integrates visual-textual complexity and model-based prediction difficulty. Our hybrid approach combines traditional features (Canny, HOG, GLCM) capturing fine-grained visual patterns with modern deep representations (ResNet50 embeddings) for text-independent difficulty measurement. This design avoids OCR-dependent methods (e.g., LayoutLM) that require text extraction. Ablation study (Table 4) validates each component's contribution. The computation process consists of the following four steps:

Step 1: Extraction of visual-textual features. We extract five visual-textual features that describe the structural variety and visual density of document images. First, the number of edge pixels detected by OpenCV's Canny operator reflects text density and structural segmentation. Second, the average gradient magnitude from the Histogram of Oriented Gradients (HOG) captures text orientation and layout patterns. Third, the contrast value from the Gray-Level Co-occurrence Matrix (GLCM) measures local intensity variation. Fourth, the homogeneity value from GLCM represents pixel similarity and local uniformity. Finally, the cosine similarity between an image embedding (from a ResNet50) and its class representative vector indicates how typical the sample is for its class.

Step 2: Feature normalization and weighted aggregation. We normalize the five visual-textual features and compute a weighted sum $d_i = \sum_{j=1}^{5} w_j f_{ij}$, where f_{ij} is the normalized j-th feature for sample i and w_j is the feature weight. Following standard practice for weight initialization, we sample the weight vector $\boldsymbol{w}$ from a symmetric Dirichlet distribution $\mathrm{Dir}([1, 1, 1, 1, 1])$, which provides an

unbiased prior over the feature importance space. The final feature score is min-max normalized to $[0, 1]$.

Step 3: Model-based prediction difficulty. While the feature score captures visual complexity, it may not fully reflect actual prediction difficulty. To address this, we use an ImageNet-pretrained ResNet18 with its final layer replaced by a randomly initialized 16-class head, computing sample-wise cross-entropy loss via a single forward pass (no weight updates). This serves as a fixed, one-time scoring tool, fully decoupled from the target classifier; scores are computed once and reused across all backbones (CNN, DiT, UDOP, LayoutLMv3) without re-computation.

Step 4: Final difficulty score calculation. To integrate both components, we convert each score to a percentile rank and compute the average.

The resulting score guides the RL agent during training. The agent interprets its action as an adjustment to the difficulty threshold, thereby steering the average difficulty of the data it selects. Full details of normalization, Dirichlet weight sampling, and rank-based aggregation appear in Suppl. Sec. 1.

3.3 RL-based Training Scheduler

To automate the data selection process, we design a training scheduler based on DDPG [22], which learns to dynamically select sample difficulty levels throughout training. DDPG outperformed alternatives including PPO [29] in our experiments, consistent with prior findings on curriculum learning with continuous action spaces [5].

Formulation as Markov Decision Process (MDP). We define this problem as a MDP, $\mathcal{M} = (S, A, R, T, \gamma)$, where each component is defined as follows. The state $s_t \in \mathbb{R}^8$ encodes the current training status of the classifier, including the difficulty threshold d_t, its change Δd_t, episode index e_t, training/validation loss $(L_{\text{train}}, L_{\text{val}})$, accuracy $(\text{Acc}_{\text{train}}, \text{Acc}_{\text{val}})$, and a clipping flag indicating whether d_t was constrained to $[0.001, 1.0]$. The agent's action $a_t \in A$ is a continuous scalar value for threshold adjustment, defined as $a_t \in [-0.01, 0.01]$, and the threshold is updated by:

$$d_{t+1} = \text{clip}(d_t + a_t, 0.001, 1.0) \tag{1}$$

A positive action ($a_t > 0$) increases the threshold and leads to the selection of more difficult samples, whereas a negative action ($a_t < 0$) decreases the threshold, favoring easier samples. The reward $r_t = R(s_t, a_t)$ is designed to reflect the classifier's improvement. We compute the reward using relative improvement in both training and validation loss:

$$r_t = \max\left(0, \frac{L_{\text{train}}^{\text{prev}} - L_{\text{train}}^{\text{curr}}}{L_{\text{train}}^{\text{prev}} + \epsilon}\right) + \lambda_t \cdot \max\left(0, \frac{L_{\text{val}}^{\text{prev}} - L_{\text{val}}^{\text{curr}}}{L_{\text{val}}^{\text{prev}} + \epsilon}\right) \tag{2}$$

where $\epsilon = 10^{-6}$ prevents division by zero and λ_t is a weight that increases linearly according to p_t. Here, p_t denotes the training progress, defined as the fraction of training data that has been selected up to the current step, according to the

difficulty threshold determined by the agent's action. This design emphasizes training loss improvement early and shifts to validation performance later. Validation loss is used exclusively for the reward signal and never updates classifier weights directly; the test set remains held out throughout both stages. We selected this approach that achieved the most stable performance among multiple reward function designs (Suppl. Sec. 2). The transition function $T(s_{t+1} \mid s_t, a_t)$ captures how the environment evolves after an action. After the classifier learns with data selected by the agent's action, the simulator calculates a new state s_{t+1} and transmits it to the agent. In this process, all state elements including threshold, loss, and accuracy are updated and used as input for the next episode.

Based on this MDP structure, the DDPG agent learns a progressive curriculum policy using feedback from classifier learning. In the early stages of learning, it induces fast convergence focusing on easy data, and as learning progresses, it selects increasingly difficult samples to maximize the model's generalization performance. This structure not only enables complete automation of curriculum design but also provides higher flexibility and adaptability than existing fixed scheduling approaches by allowing fine-grained control of difficulty-based learning.

Two-stage learning procedure. To implement the DDPG agent within our curriculum learning framework, we define two separate stages:

(1) Agent Training Stage. The DDPG agent interacts repeatedly with the classifier. At each episode, the agent adjusts the difficulty threshold through a continuous action based on feedback from the classifier's learning performance and selects samples with difficulty scores below this threshold to train the classifier. The threshold represents the selection boundary from easy data (EASY) to hard data (HARD) on the difficulty distribution of the entire training dataset. This process constitutes one episode, and episodes are repeated until all data are exhausted to complete one cycle. The entire training process consists of multiple cycles, where the classifier is initialized in each cycle to enable learning curriculum policies in various learning situations. The simplified version is presented in Algorithm 1, and the full details are provided in Suppl. Sec. 3.1.

(2) Curriculum Application Stage. The trained DDPG agent is used as a scheduler to guide the training of the classifier. In this stage, the agent selects the data dynamically according to its learned policy but does not update the policy or receive any rewards. This corresponds to the inference phase of reinforcement learning. The pseudo-code is summarized in Suppl. Sec. 3.2.

These two stages fully separate the design and application of curriculum learning: the DDPG agent can learn effective data selection strategies and reliably apply them to classifier training. Importantly, the learned curriculum policy can be flexibly reused across various backbone architectures (from CNNs to transformers) and learning paradigms (from-scratch training and fine-tuning), enabling broad applicability without task-specific modifications.

DDPG agent design and training. The DDPG agent follows the standard actor-critic architecture with target networks for stable training. The actor network $\mu(s_t)$ receives the state vector s_t as input and outputs a continuous action

Algorithm 1 Training DDPG Agent as Curriculum Scheduler (Simplified)

Require: Training dataset D_{train}, validation dataset D_{val}, DDPG agent $\mathcal{A}$ (actor μ, critic Q), classifier M, Replay buffer $\mathcal{B}$, #cycles C, epochs/episode u, batch size b

Ensure: Trained agent $\mathcal{A}$

Pre-compute sample-wise difficulty scores for D_{train} with the *Difficulty Measurer*.
for cycle $c = 1$ to C **do**
  Initialize classifier M; set threshold $d_0 \leftarrow 0.5$.
  while $d_t < 1.0$ **do**
    Observe current state s_t.
    Select action $a_t \leftarrow \mu(s_t) + \mathcal{N}(0, \sigma)$: OU noise
    Update threshold d_{t+1} by Eq. (1).
    Select training subset $D_t \subseteq D_{\text{train}}$ where $difficulty\ score \leq d_{t+1}$.
    Train classifier M for u epochs on D_t.
    Compute reward r_t by Eq. (2) on D_t and D_{val}.
    Observe next state s_{t+1}.
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$.
    if $|\mathcal{B}| \geq b$ **then**
      Sample a mini-batch from $\mathcal{B}$.
      Update Q, μ, and soft update by Eq. (3).
    end if
    $s_t \leftarrow s_{t+1}$, $d_t \leftarrow d_{t+1}$.
  end while
end for
return Trained agent $\mathcal{A}$.

a_t, while the critic network $Q(s_t, a_t)$ estimates the Q-value for a state-action pair.

The actor network consists of two hidden layers followed by an output layer with a Tanh activation to constrain the output within $[-1, 1]$. This output is then scaled by a factor of 0.01 to produce the final action value within $[-0.01, 0.01]$. The critic network takes the state s_t and the action a_t as inputs and predicts the corresponding Q-value $Q(s_t, a_t)$. It is trained to minimize the Bellman loss function:

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_{(s,a,r,s')} \left[\left(Q(s, a) - y \right)^2 \right],$$

$$\text{where } y = r + \gamma Q_{\text{target}}(s', \mu_{\text{target}}(s'))$$

Here, Q_{target} and μ_{target} denote the target critic and actor networks, respectively, and γ is the discount factor.

To improve exploration during training, we add Ornstein–Uhlenbeck (OU) noise [33] to the actor's output. The final action becomes $a_t = \mu(s_t) + N_t$, where the OU noise N_t follows:

$$N_{t+1} = N_t + \theta(\mu - N_t) + \sigma \cdot \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, I)$$

with mean reversion rate $\theta = 0.2$, mean $\mu = 0$, and standard deviation $\sigma = 0.8$.

For stable critic training, inspired by the TD3 framework [7], we add clipped Gaussian noise to target actions during Q-value computation:

$$\tilde{a}' = \mu_{\text{target}}(s') + \text{clip}(\mathcal{N}(0, 0.2), -0.5, 0.5)$$

Target networks are updated using soft updates at each step with a soft update coefficient $\tau = 0.01$:

$$\begin{aligned} \mu_{\text{target}} &\leftarrow \tau\mu + (1-\tau)\mu_{\text{target}} \\ Q_{\text{target}} &\leftarrow \tau Q + (1-\tau)Q_{\text{target}} \end{aligned} \tag{3}$$

Both actor and critic networks use the Adam optimizer with learning rate 3×10^{-4}. Through this architecture, the agent learns a dynamic and context-adaptive threshold selection policy. Unlike static scheduling, it can respond to ongoing changes during training, continually refining the curriculum to match the current state of the learner.

4 Experiments

In this section, we evaluate the performance of the proposed framework through a series of experiments. We first introduce the experimental setup, followed by quantitative and qualitative analyses.

4.1 Experimental Setup

Datasets. We conducted experiments on the RVL-CDIP dataset [10], which consists of 400,000 grayscale document images categorized into 16 document types. To ensure class balance, we sampled an equal number of images from each class, resulting in a total of 100,000 images used for our experiments. The dataset was split into training, validation, and test sets with a ratio of 64:16:20, respectively. While document images often have high resolution, large input dimensions lead to increased computational costs and overfitting risks. Moreover, layout information is generally more important than detailed characters in document image classification [1]. Therefore, all images were resized to 224×224 pixels prior to input.

Classifier network architecture. For from-scratch training, we use a CNN with six 3×3 convolutional layers, three max pooling layers, batch normalization, and dropout, ending with 16-class classification (additional details in Suppl. Sec. 4). The proposed CNN contains approximately 1.35M parameters, making it a lightweight baseline compared to transformer-based models. To evaluate TiCRL's generalizability across different architectures and learning paradigms, we additionally conduct fine-tuning experiments on pretrained transformer-based document classifiers: DiT [21], UDOP [32], and LayoutLMv3 [11]. Each model is fine-tuned on the RVL-CDIP dataset using configurations validated through preliminary trials: 150 epochs for DiT, 15 for UDOP, and 10 for LayoutLMv3.

Compared methods. To validate the effectiveness of our proposed method (TiCRL), we compared it against various baselines organized into four categories:

- *Without Curriculum.* A standard deep learning model that uses all training data without curriculum. CNN and ResNet34 were included for comparison.
- *Pre-defined Curriculum Learning.* Data are sorted by difficulty scores and gradually introduced into training according to a fixed scheduling function (e.g., linear, root, step), without agent learning.
- *Automated Curriculum Learning.* These approaches adapt sample selection or weighting based on the model's learning state rather than predefined scores. Self-Paced Learning (SPL) [20] prioritizes easier, lower-loss samples early and progressively incorporates harder ones. Adaptive Curriculum Learning (ACL) [17] uses a pretrained teacher model to estimate difficulty, dynamically adjusting curriculum weights via pacing functions.
- *TiCRL.* TiCRL sorts training data by difficulty score and uses a trained DDPG agent to adjust thresholds for progressive sample selection.

4.2 Results and Analysis

Quantitative analysis (CNN-based models). Table 1 summarizes accuracy and average training samples used across all methods. Unlike pre-defined and automated CL methods that follow progressive but fixed data selection patterns, TiCRL dynamically adjusts exposure by reducing it during convergence and increasing it during plateaus, resulting in an average usage of 81.92% of training data (per-epoch trajectories in Suppl. Sec. 5). TiCRL achieved the highest mean accuracy (83.67%), outperforming pre-defined CL by 6.7–9.5%p and automated CL by 7.8–13.8%p, surpassing the CNN baseline (80.43%) despite reduced data exposure. These results indicate that fixed or loss-based curricula, which do not account for the learner's state, lead to suboptimal sample ordering even when an appropriate difficulty measurer is employed. To ensure fairness in data utilization, additional experiments with CNN and ResNet34 trained on 80% data confirmed substantially lower accuracies (78.69% and 77.68%), with all TiCRL improvements statistically significant (paired t-test, $p < 0.01$, Cohen's d: 2.17–7.62). All statistical comparisons are based on 5 independent runs per method using different random seeds; no correction for multiple comparisons was applied. As RVL-CDIP is class-balanced, accuracy and macro-F1 are nearly identical in practice; we confirm empirically a difference of less than 0.1 percentage points across our TiCRL runs (CNN- and DiT-based).

Ablation of the difficulty measurer. To evaluate the contribution of each component in the hybrid difficulty measurer, we remove individual features while keeping all other settings identical. Table 2 reports the results using a representative seed.

Removing any single component consistently degrades performance by 2.70–3.98 percentage points. GLCM Contrast produces the largest drop (−3.98%p), suggesting its importance for capturing layout complexity. Even the smallest drop (GLCM Homogeneity: −2.70%p) remains significant. These results indicate that the features provide complementary signals for difficulty estimation, supporting our design choice of combining multiple visual-textual cues rather than relying on a single feature or loss-based difficulty alone.

Table 1: Summary of compared methods in terms of difficulty source, curriculum strategy, classification accuracy (Mean ± Std), and average number of training samples used. All TiCRL improvements are statistically significant (paired t-test, $p < 0.01$)

Method	Difficulty	Strategy	Acc. (%)	Data Used
w/o Curriculum Learning (100% data and 80% data):				
CNN	–	None	80.43 ± 0.33	64,000 (100%)
ResNet34	–	None	78.48 ± 0.39	64,000 (100%)
CNN	–	None	78.69 ± 0.46	51,200 (80%)
ResNet34	–	None	77.68 ± 0.24	51,200 (80%)
Pre-defined CL:				
Linear	Score	Linear fn.	74.16 ± 1.43	48,106 (75.17%)
Step	Score	Step fn.	74.22 ± 0.98	48,528 (75.83%)
Root	Score	Root fn.	76.96 ± 0.75	53,435 (83.49%)
Automated CL:				
SPL	Loss	Self-paced	69.89 ± 1.82	48,212 (75.33%)
ACL	Model+Loss	Pacing	75.84 ± 0.48	46,416 (72.53%)
TiCRL	**Score**	**RL-based**	**83.67 ± 1.61**	**52,429 (81.92%)**

Table 2: Ablation of the hybrid difficulty measurer. Removing any component consistently degrades performance. All experiments use identical CNN architecture and training settings.

Component Removed	Accuracy (%)	Drop (%p)
None (Full TiCRL)	85.05	-
w/o Canny Edge	81.40	-3.65
w/o HOG	81.63	-3.42
w/o GLCM Contrast	81.07	-3.98
w/o GLCM Homogeneity	82.35	-2.70
w/o Cosine Similarity	81.23	-3.82

Ablation on action space. We tested a 2D-action variant where the agent controls both starting threshold and data ratio, achieving 72.30±10.18% accuracy with high variance. This confirms that our streamlined 1D threshold adjustment provides better learning stability and performance than multi-dimensional curriculum control.

Ablation on scheduler design. We compared TiCRL against a non-RL EMA-based threshold heuristic (5-seed average), which improves over the no-CL baseline (81.49% vs. 80.43%) but underperforms TiCRL (83.67%), confirming that the agent's multi-signal state representation provides benefits beyond simple adaptive heuristics.

Quantitative analysis (Transformer-based models). Table 3 summarizes performance of transformer-based document classifiers under TiCRL. Each model

Table 3: Performance comparison of transformer-based document classifiers and TiCRL inference.

Model	Acc. (Mean ± Std)	% Data Used
DiT	91.07 ± 0.35	80.00%
UDOP	74.00 ± 0.95	80.00%
LayoutLMv3	91.53 ± 0.26	80.00%
DiT + TiCRL	**93.02 ± 2.05**	57.66% (avg.)

was fine-tuned on the RVL-CDIP dataset under empirically validated configurations that ensure convergence: 150 epochs for DiT, 15 for UDOP, and 10 for LayoutLMv3. These epoch settings were determined through preliminary trials to reflect typical convergence points, ensuring fair comparisons. When applied to DiT, the DDPG agent adaptively adjusted the difficulty threshold based on the learner's evolving state, achieving 93.02% accuracy using only 57.66% of training data and surpassing all transformer baselines including LayoutLMv3 (91.53%) and DiT (91.07%) trained on 80% data. This indicates that TiCRL conducts an adaptive re-finetuning process that reorganizes data utilization based on model feedback, rather than a standard fine-tuning phase. These results confirm that TiCRL generalizes across learning paradigms, from CNNs trained from scratch to large-scale pretrained transformers.

Qualitative analysis (CNN-based models). To analyze how the agent maximizes rewards and stabilizes its policy during training, we first examined the complete training log based on Algorithm 1. This analysis focused on the final agent obtained at the end of 10 curriculum cycles (Episode 743), from which we identified the *best agent* that achieved the lowest validation loss during training (Episode 379). All subsequent inference and performance evaluations in this study were conducted using this best agent. By focusing on this generalization-optimal point, we aim to provide a more precise analysis of inference performance and policy stability.

Figure 2 summarizes the training dynamics across 10 cycles. For each metric, both raw values (dotted line) and smoothed trends (solid line) are shown, with alternating white and gray backgrounds delineating cycle boundaries. The reward exhibited high variability during early episodes of each cycle but gradually stabilized, indicating policy improvement. The Q-value increased gradually within each cycle, suggesting that the critic increasingly assigned higher values to effective state–action pairs. The actor loss initially decreased sharply and then stabilized, reflecting policy convergence. Notably, after cycle 7, the actor loss began to rise slightly, potentially indicating policy overfitting. The threshold was flexibly adjusted rather than monotonically increasing, demonstrating adaptive difficulty control based on classifier state.

Most cycles terminated via early stopping when validation loss stagnated for five consecutive evaluations. The best agent emerged around cycle 6 when

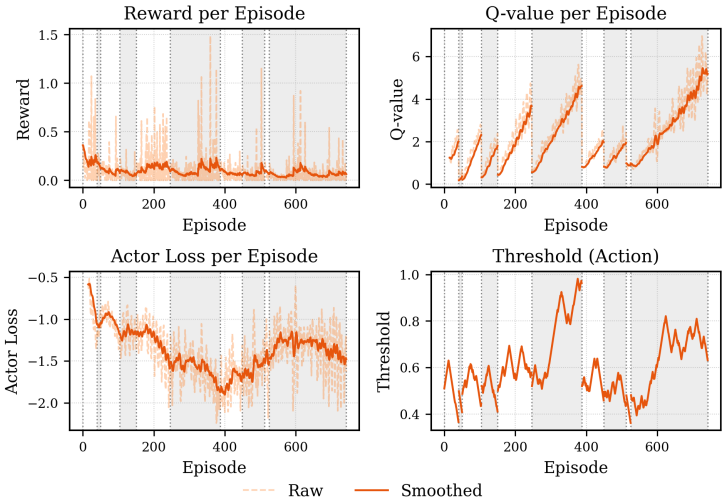

Fig. 2: Learning dynamics of the TiCRL agent over 10 cycles. Smoothed curves are computed via an exponential moving average (EMA) with a smoothing factor of 0.9.

validation accuracy peaked and actor loss minimized, underscoring the value of tracking policy-level signals for agent selection.

Additionally, to analyze TiCRL's performance across data difficulty levels, we defined HARD (advertisement, news article, scientific publication) and EASY (file folder, budget, letter) groups based on average difficulty scores, representing the top and bottom 3 classes respectively (detailed distributions in Suppl. Sec. 6). Figure 3 shows boxplots comparing the difficulty score distributions of samples that TiCRL predicted correctly (True) and incorrectly (False) for each class. In the HARD group, the median difficulty score of correctly predicted samples is relatively high, indicating that TiCRL successfully handles difficult samples with high accuracy. This result suggests that TiCRL does not simply repeat training on easy samples, but also achieves generalization on high-difficulty examples. In contrast, in the EASY group, the difficulty scores are generally low, and TiCRL's predictions tend to be concentrated on easier samples.

We also visualized class-wise accuracy for each model as a heatmap, separately for the HARD and EASY groups, as shown in Figure 4. In the HARD group, TiCRL achieves competitive accuracy across all classes (84.6%, 83.7%, 88.6% for advertisement, news article, and scientific publication, respectively), with an average of 85.6% that surpasses pre-defined CL (69.1% avg.) and automated CL (77.1% avg.) (per-method breakdown in Figure 4). While ResNet-34 baseline achieves comparable accuracy on individual hard classes using 100% of training data, TiCRL demonstrates superior overall performance with only 82% of the data. This is particularly remarkable given that TiCRL uses only 82% of training data and selects predominantly easier samples during training. On the other hand, in the EASY group, most models achieve high accuracy with

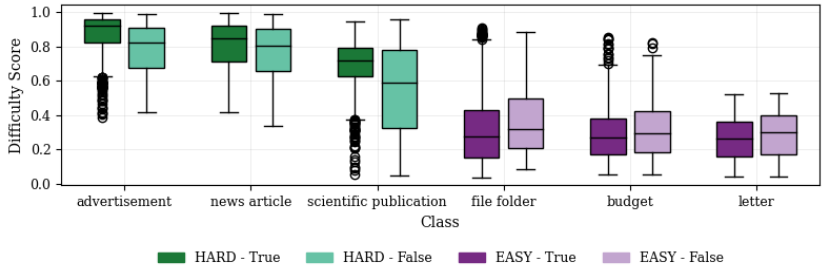

Fig. 3: Distribution of difficulty scores for True/False samples predicted by TiCRL per class.

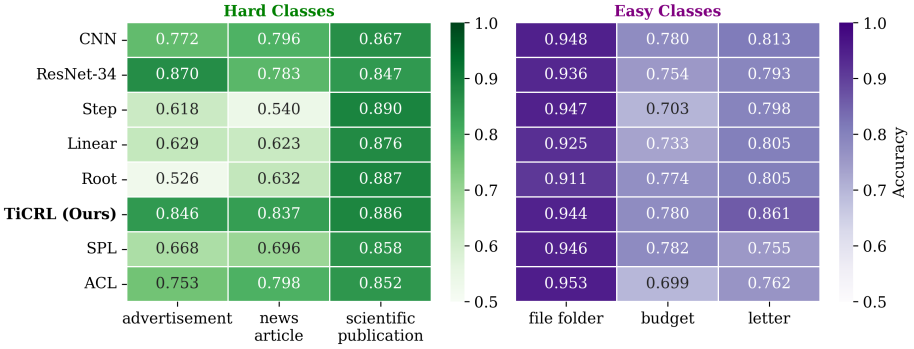

Fig. 4: Heatmap of class-wise accuracy for each model on HARD and EASY groups.

small performance gaps. This suggests that the advantages of RL-based strategies are relatively less pronounced for easy samples, and various curriculum strategies can secure similar levels of generalization performance. These findings demonstrate that TiCRL forms an effective learning sequence based on visual-textual feature-based difficulty measures, and particularly improves performance on more challenging samples through adaptive curriculum scheduling.

Performance of best and final agents. To assess the practicality and re-usability of our approach, we evaluate agent performance on unseen datasets with varying sizes by comparing two checkpoints. The final agent was saved at the last training step, whereas the best agent was chosen by its validation score. Table 4 shows that the best agent surpassed the final agent on every dataset size. On the smallest split (10K samples) the margin exceeded 18%p. The best agent also kept its accuracy stable across random draws, which suggests strong tolerance to data variation.

To analyze the underlying causes of this performance gap, we examined the raw actions of both agents. The best agent initially explored a relatively wider action range and gradually converged to a stable policy, whereas the final agent operated within a narrow range from the beginning, leading to limited gener-

Table 4: Performance comparison between the final and best agent across various unseen test sets. For the 10K dataset, results are averaged over 5 different random seeds.

Dataset	Acc. (Final Agent)	Acc. (Best Agent)
100K	83.12	85.61
50K	78.87	84.89
10K	61.99	80.48

alization (Suppl. Sec. 7). These behavioral differences align with the training dynamics observed in our training logs, where policy-level indicators such as reward, Q-value, and actor loss suggested that the agent had already reached an optimal policy before the final training stage. These results highlight the importance of policy-level early stopping based on RL-specific indicators such as reward, Q-value, and actor loss, rather than classifier validation loss alone. By identifying the point at which exploration diminishes and policy converges, practitioners can preserve a more transferable and robust curriculum strategy.

Cross-dataset transfer. To assess policy transferability, we applied the RVL-CDIP-trained DDPG agent to Tobacco-3482 [19] without retraining. Tobacco-3482 differs substantially from RVL-CDIP in both scale (3,482 vs. 320K images) and label space (10 vs. 16 classes), posing a meaningful domain shift. Despite these differences, TiCRL achieved 86.88±1.10% accuracy versus 85.30±1.48% for the baseline trained for 100 epochs, demonstrating that the learned curriculum policy generalizes effectively across datasets with different class structures and sizes without any retraining.

5 Conclusion

This study demonstrated that RL-based curriculum learning is an effective and efficient strategy for textual image classification. On lightweight CNN models trained from scratch (1.35M parameters), TiCRL achieved 83.67% accuracy using only 82% of training data, substantially outperforming all curriculum learning baselines by up to 13.8%p. On pretrained transformers, TiCRL reached 93.02% accuracy with DiT using only 58% of training data, surpassing all transformer baselines trained on 80% of data. The hybrid difficulty measurer captures structural layout complexity without OCR, and the RL-based scheduler dynamically adapts to the learner's state, jointly enabling robust generalization to hard-to-classify categories. The learned policy further transferred to unseen datasets and architectures without retraining, validating TiCRL as a broadly applicable curriculum framework. Future work can extend TiCRL to other vision tasks, explore policy transferability across broader domains, and investigate confidence-based agent selection mechanisms incorporating policy-level diagnostics (e.g., actor/critic stability, Q-value variance, action entropy) validated on a held-out test fold.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00455158 and RS-2025-16068665). This was also received from the Ministry of Education of the Republic of Korea, the National Research Foundation of Korea (2022S1A3A2A02089685)

References

1. Afzal, M.Z., Capobianco, S., Malik, M.I., Marinai, S., Breuel, T.M., Dengel, A., Liwicki, M.: DeepDocClassifier: Document classification with deep convolutional neural network. In: ICDAR. pp. 1111–1115 (2015)
2. Asim, M.N., Khan, M.U.G., Malik, M.I., Razzaque, K., Dengel, A., Ahmed, S.: Two stream deep network for document image classification. In: ICDAR. pp. 1410–1416 (2019)
3. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: ICML. pp. 41–48 (2009)
4. Dengel, A., Dubiel, F.: Clustering and classification of document structure—A machine learning approach. In: ICDAR. pp. 587–591 (1995)
5. El-Bouri, R., Eyre, D., Watkinson, P., Zhu, T., Clifton, D.: Student-teacher curriculum learning via reinforcement learning: Predicting hospital inpatient admission location. In: ICML. pp. 2848–2857 (2020)
6. Fan, Y., Tian, F., Qin, T., Li, X.Y., Liu, T.Y.: Learning to teach. In: ICLR. vol. 34, pp. 1–16 (2018)
7. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: ICML. pp. 1587–1596 (2018)
8. Guo, S., Huang, W., Zhang, H., Zhuang, C., Dong, D., Scott, M.R., Huang, D.: CurriculumNet: Weakly supervised learning from large-scale web images. In: ECCV. pp. 135–150 (2018)
9. Hacohen, G., Weinshall, D.: On the power of curriculum learning in training deep networks. In: ICML. pp. 2535–2544 (2019)
10. Harley, A.W., Ufkes, A., Derpanis, K.G.: Evaluation of deep convolutional nets for document image classification and retrieval. In: ICDAR. pp. 991–995 (2015)
11. Huang, Y., Lv, T., Cui, L., Lu, Y., Wei, F.: LayoutLMv3: Pre-training for document ai with unified text and image masking. In: ACM MM. pp. 4083–4091 (2022)
12. Jain, R., Wigington, C.: Multimodal document image classification. In: ICDAR. pp. 71–77 (2019)
13. Jiang, L., Meng, D., Mitamura, T., Hauptmann, A.G.: Easy samples first: Self-paced reranking for zero-example multimedia search. In: ACM MM. pp. 547–556 (2014)
14. Jiang, L., Zhou, Z., Leung, T., Li, L.J., Fei-Fei, L.: MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In: ICML. pp. 2304–2313 (2018)
15. Jobin, K.V., Mondal, A., Jawahar, C.V.: Document image analysis using deep multi-modular features. SN Comput. Sci. **4**(1), 5 (2022)
16. Kang, L., Kumar, J., Ye, P., Li, Y., Doermann, D.: Convolutional neural networks for document image classification. In: ICPR. pp. 3168–3172 (2014)
17. Kong, Y., Liu, L., Wang, J., Tao, D.: Adaptive curriculum learning. In: ICCV. pp. 5067–5076 (2021)

18. Kumar, G., Foster, G., Cherry, C., Krikun, M.: Reinforcement learning based curriculum optimization for neural machine translation. In: NAACL-HLT. pp. 2054–2061 (2019)
19. Kumar, J., Ye, P., Doermann, D.: Structural similarity for document image classification and retrieval. Pattern Recognition Letters **43**, 119–126 (2014)
20. Kumar, M., Packer, B., Koller, D.: Self-paced learning for latent variable models. In: NeurIPS (2010)
21. Li, J., Xu, Y., Lv, T., Cui, L., Zhang, C., Wei, F.: DiT: Self-supervised pre-training for document image transformer. In: ACM MM. pp. 3530–3539 (2022)
22. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N.M.O., Erez, T., Tassa, Y., Silver, D., Wierstra, D.P.: Continuous control with deep reinforcement learning. In: ICLR (2016)
23. Liu, L., Wang, Z., Qiu, T., Chen, Q., Lu, Y., Suen, C.Y.: Document image classification: Progress over two decades. Neurocomputing **453**, 223–240 (2021)
24. Matiisen, T., Oliver, A., Cohen, T., Schulman, J.: Teacher–student curriculum learning. IEEE Trans. Neural Netw. Learn. Syst. **31**(9), 3732–3740 (2020)
25. Mirzasoleiman, B., Bilmes, J., Leskovec, J.: Coresets for data-efficient training of machine learning models. In: ICML. pp. 6950–6960 (2020)
26. Paul, M., Ganguli, S., Dziugaite, G.K.: Deep learning on a data diet: Finding important examples early in training. In: NeurIPS. pp. 20596–20607 (2021)
27. Pentina, A., Sharmanska, V., Lampert, C.H.: Curriculum learning of multiple tasks. In: CVPR. pp. 5492–5500 (2015)
28. Platanios, E.A., Stretcu, O., Neubig, G., Poczos, B., Mitchell, T.M.: Competence-based curriculum learning for neural machine translation. In: NAACL-HLT. pp. 1162–1172 (2019)
29. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
30. Seedat, N., Crabbé, J., Bica, I., van der Schaar, M.: Data-IQ: Characterizing subgroups with heterogeneous outcomes in tabular data. In: NeurIPS. pp. 23660–23674 (2022)
31. Spitkovsky, V.I., Alshawi, H., Jurafsky, D.: From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In: NAACL-HLT. pp. 751–759 (2010)
32. Tang, Z., Yang, Z., Wang, G., Fang, Y., Liu, Y., Zhu, C., Zeng, M., Zhang, C., Bansal, M.: Unifying vision, text, and layout for universal document processing. In: CVPR. pp. 19254–19264 (2023)
33. Uhlenbeck, G.E., Ornstein, L.S.: On the theory of the brownian motion. Phys. Rev. **36**(5), 823 (1930)
34. Wang, X., Chen, Y., Zhu, W.: A survey on curriculum learning. IEEE TPAMI **44**(9), 4555–4576 (2021)
35. Wei, Y., Liang, X., Chen, Y., Shen, X., Cheng, M.M., Feng, J., Zhao, Y., Yan, S.: STC: A simple to complex framework for weakly-supervised semantic segmentation. IEEE TPAMI **39**(11), 2314–2320 (2016)
36. Zhao, M., Wu, H., Niu, D., Wang, X.: Reinforced curriculum learning on pre-trained neural machine translation models. In: AAAI. pp. 9652–9659 (2020)