applied sciences

MDPI

Article

A Staged Curriculum Learning Framework for Solving Minesweeper

Wenhui Ruan [1], Wenbo Wang [1,*] and Chengyou Lei [2,*]

[1] College of Artificial Intelligence and Low-Altitude Technology, South China Agricultural University, Guangzhou 510642, China; ruanwenhui0920@163.com
[2] Wuhan Second Ship Design and Research Institute, Wuhan 430205, China
* Correspondence: wwb@scau.edu.cn (W.W.); leichengyou@126.com (C.L.)

Abstract

Minesweeper is both an intellectual challenge for human players and an effective testbed for intelligent algorithms. For the standard Minesweeper task on a 9 × 9 grid with 10 mines, a convolutional neural network (CNN)-based supervised learning approach, together with optimized network architecture, state encoding, and dynamic learning rate scheduling, achieves a test win rate of 85.16% (SD = 0.36 percentage points), exceeding the baseline Single-Point (SP) algorithm by more than 10 percentage points. The combined use of dropout and data augmentation achieves a higher test win rate than either technique alone under the evaluated settings. Building on this model, we propose an easy-to-hard staged curriculum learning (SCL) strategy to address the increasing task difficulty caused by higher mine density and the difficulty of training models from scratch. By progressively increasing the number of mines and transferring model parameters from the preceding training stage, the model can inherit previously acquired knowledge and gradually adapt to more complex tasks. Representative SCL models achieved test win rates of 55.85%, 20.38%, and 4.00% on the 14-, 18-, and 22-mine tasks, respectively. In the three-seed ablation study on the 18-mine task using validation-based checkpoint selection, SCL achieved the highest mean test win rate of 21.65%, compared with 20.85% for mixed-density adaptation, 17.08% for direct transfer, and 15.59% for training from scratch. SCL achieved numerically higher test win rates than mixed-density adaptation across all three seeds, although the difference was not statistically significant. However, a constraint satisfaction problem (CSP) solver achieves a higher win rate of 47.17% on the same task, highlighting the advantage of explicit constraint-based reasoning in absolute solving performance. Further analysis of clue-value distributions shows that the frequencies of local patterns change with mine density, providing a structural motivation for the easy-to-hard curriculum. These findings indicate that staged curriculum learning can facilitate knowledge transfer across increasingly difficult Minesweeper tasks, although reliable solving of high-density configurations remains challenging.

Keywords: Minesweeper; staged curriculum learning; supervised learning; data augmentation; deep learning; convolutional neural network

Check for updates

Academic Editor: Andrea Prati

Received: 1 October 2026
Revised: 4 November 2026
Accepted: 5 November 2026
Published: 7 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Minesweeper first appeared in the Windows operating system, originally intended to help users become familiar with mouse movement and clicking, and later evolved into a classic logic game with substantial intellectual challenge. During gameplay, a player

must infer the locations of hidden mines from the numerical clues provided by revealed cells. This process requires reasoning and decision making under limited information, making Minesweeper a representative task for studying the reasoning and decision-making capabilities of intelligent algorithms.

In Minesweeper, an agent must infer the distribution of mines among adjacent unrevealed cells from the clues provided by revealed cells and then assess risk and choose among multiple candidate actions. Because the complete minefield state is not directly observable, the agent can only update its understanding of the environment incrementally from limited local observations. Minesweeper therefore exhibits prominent characteristics of incomplete information, randomness, and sequential decision making. As the grid size and number of mines increase, the number of possible minefield states grows exponentially, enabling the task to jointly evaluate spatial feature extraction, logical reasoning, and complex decision-making capabilities [1,2].

From the perspective of computational complexity, Minesweeper presents a challenging combinatorial problem. Previous studies have established that the generalized Minesweeper decision problem is NP-complete [1], reflecting the complexity of reasoning about hidden mine configurations as problem size increases. In the fixed 9×9 environments investigated in this study, increasing mine density also introduces greater decision risk and more challenging learning conditions. These characteristics make Minesweeper a valuable experimental platform for investigating intelligent reasoning and autonomous decision making under uncertainty [2].

Existing approaches to solving Minesweeper mainly include rule-based reasoning, constraint-based search, probability estimation, and machine learning. Early methods primarily relied on hand-crafted logical rules, for example, determining whether adjacent unrevealed cells are mines or safe cells according to numerical clues. Such methods are computationally efficient and interpretable and can solve states containing deterministic local patterns effectively. However, when the current state contains no safe cell or mine that can be determined directly, purely local rules often cannot support further inference, limiting their effectiveness in complex environments [3].

To overcome the limitations of local rule-based methods, researchers have further formulated Minesweeper as a constraint satisfaction problem (CSP), transforming mine-location inference into a mathematical constraint-solving process by establishing constraints between clues and unrevealed cells. Compared with simple rule-based methods, constraint-based search can integrate relationships among multiple local regions and therefore provides stronger reasoning capability [4]. Beyond conventional CSP formulations, Buffet et al. combined Upper Confidence Tree search with domain-specific solvers [5], while Tu et al. explored heuristic probing strategies and their combinations to improve Minesweeper success rates [6]. However, as the minefield size grows, the numbers of variables and constraints increase rapidly, expanding the search space and incurring substantial computational cost. In addition, conventional rule-based and constraint-based approaches typically rely on manually constructed rules or constraint mechanisms and cannot readily learn decision strategies for complex environments from large amounts of experience data [7].

In recent years, advances in machine learning and deep learning have led to increasing use of data-driven methods for constructing Minesweeper agents. Unlike conventional methods, machine learning models can automatically learn mappings between states and decisions from training data, thereby reducing dependence on hand-crafted rules. In particular, convolutional neural networks (CNNs) provide strong spatial feature extraction capabilities: they exploit local receptive fields to extract neighborhood information and hierarchical feature learning to capture spatial dependencies in grid data, offering a new approach for Minesweeper problems with an inherently two-dimensional structure [8].

In previous work, Wang and Lei formulated Minesweeper as a sequential decision-making task and constructed a deep Q-network (DQN) model and a supervised learning (SL) model based on CNNs. By encoding the minefield state, the neural networks directly learned the relationship between minefield states and mine distributions and were evaluated on a 6 × 6 grid with 4 mines. The average win rates of the DQN and SL models reached 93.3% and 91.2%, respectively, demonstrating strong Minesweeper decision-making performance. Supervised learning showed clear advantages in training speed and training stability [9]. These results indicate that CNN-based data-driven methods can effectively learn the spatial features and decision rules of Minesweeper and provide a foundation for training agents in more complex environments. Unlike the previous study, which focused on CNN-based agents in a fixed 6 × 6 environment, the present work investigates model adaptation across different mine densities on a larger 9 × 9 grid. The primary focus is on organizing the training process to facilitate knowledge transfer between tasks of increasing difficulty.

In addition to supervised learning, reinforcement learning (RL) is an important direction for constructing Minesweeper agents. In RL, an agent continuously interacts with the environment and adjusts its policy according to reward feedback, allowing it to learn decision behavior autonomously. Conventional Q-learning requires explicit maintenance of state–action values and is therefore limited by the curse of dimensionality when the state space becomes large [10]. Deep reinforcement learning (DRL) uses deep neural networks to approximate value functions and employs mechanisms such as experience replay and target networks to improve training stability. Previous studies have shown that DQNs can learn high-performance policies in complex game environments [11]. Earlier Minesweeper experiments likewise showed that DQN can achieve a high win rate, but it requires more training steps than supervised learning and exhibits larger fluctuations during training [9]. Therefore, in simulated Minesweeper environments where ground-truth labels can be obtained, supervised learning offers advantages in training efficiency and stability.

However, existing studies have mainly verified the effectiveness of deep neural networks in Minesweeper environments of fixed difficulty. When the grid size or number of mines increases further, the minefield state space expands rapidly, while state uncertainty and decision risk also increase, making model training considerably more difficult. If tasks at different difficulty levels are trained independently from random initialization, the model must repeatedly relearn basic spatial features and local reasoning rules already acquired on easier tasks, while failing to fully exploit knowledge accumulated by existing models. How to transfer knowledge learned from lower-difficulty tasks to higher-difficulty tasks is therefore an important problem for improving the training efficiency and solving performance of complex Minesweeper tasks. An additional question is whether progressively introducing intermediate difficulty levels offers advantages over directly transferring pretrained parameters to a harder task or training on multiple difficulty levels simultaneously.

Curriculum learning (CL) provides a natural approach to this problem. Inspired by the human learning process from easy to difficult tasks, curriculum learning arranges training tasks by difficulty so that the model can progressively acquire more complex knowledge and strategies. Previous studies have shown that curriculum learning can reduce optimization difficulty, improve training efficiency, and enhance generalization performance on complex tasks [12–14]. Further research has indicated that, in reinforcement learning and complex decision-making tasks, an appropriate progressive task-difficulty schedule can reduce the search space faced by a model early in training and improve learning stability [15,16].

Minesweeper itself exhibits a natural progression of difficulty that is closely related to the number of mines, mine density, and state complexity. Easier environments generally contain more readily recognizable local patterns, which facilitate learning of basic spatial features and reasoning rules. As the number of mines increases, the proportion of safe

cells decreases, while state uncertainty and decision risk increase, requiring the model to integrate more spatial information. Introducing intermediate difficulty levels provides a stepwise learning path, allowing the model to adapt to progressively changing task distributions rather than moving directly from an easier task to a substantially harder one. Thus, an easy-to-hard sequence of training stages can be constructed according to the number of mines, with model parameters from each stage transferred to the next so that the model progressively adapts to more complex Minesweeper environments on the basis of previously acquired knowledge.

Based on this idea and our previous work on Minesweeper agents [9], this study further investigates model training for high-difficulty Minesweeper tasks and proposes a Minesweeper-solving framework based on staged curriculum learning (SCL). First, a CNN-based supervised learning model is constructed using multi-channel state encoding to represent spatial information in the minefield, together with dynamic learning rate scheduling and data augmentation to improve training stability and generalization performance. The number of mines is then used as the primary variable controlling task difficulty. Training is organized into a fixed easy-to-hard sequence of 10 → 14 → 18 → 22 mines, with model parameters transferred between consecutive stages. Through this progressive training process, the CNN can build upon spatial representations and decision-making patterns acquired from easier tasks, gradually adapting its learned knowledge to increasingly complex minefield environments. To further investigate the effectiveness of this staged training strategy, we compare SCL with training from scratch, direct transfer, and mixed-density adaptation, examining whether the intermediate learning stages and curriculum organization contribute to improved performance on higher-density tasks.

The main contributions of this work are as follows: (a) We extend our previous CNN-based supervised learning framework to a 9 × 9 Minesweeper environment and investigate the effects of cosine annealing, dropout, and geometric data augmentation on model training and test performance. (b) We introduce a staged curriculum learning strategy that progressively increases mine density while transferring CNN parameters between consecutive training stages. (c) We evaluate the proposed strategy across different mine densities and conduct an ablation study comparing SCL with training from scratch, direct transfer, and mixed-density adaptation. In addition, we compare the learning-based agent with both Single-Point and constraint-based solvers to clarify its relative performance and limitations.

2. Algorithmic Framework and Model Architecture

2.1. Minefield State Encoding

The raw minefield state is represented as a 9 × 9 grid containing 81 cells. Each cell takes a value from −1 to 8: −1 denotes an unrevealed cell, whereas 0–8 denote revealed values, with 0 indicating a blank cell. The one-hot encoding scheme used in this study is shown in Figure 1. Channels 1–9 correspond to one-hot encodings of clue values 0–8. For example, if a cell has a value of 3, its value in channel 4 is 1 and its values in all other channels are 0. Channel 10 encodes the value −1: if a cell is unrevealed, its value in channel 10 is 1; otherwise, it is 0.

After one-hot encoding, the minefield state has dimensions 10 × 9 × 9. Each channel corresponds to one observable cell state and contains only 0 or 1. By separating different states into independent channels, this representation provides a clear and consistent input structure for the CNN. Its effectiveness was evaluated in our previous study [9], which compared one-hot encoding with alternative state representations and supported its selection for CNN-based Minesweeper agents. Accordingly, the present study adopts

the same encoding strategy, while focusing on staged curriculum learning across different mine densities.

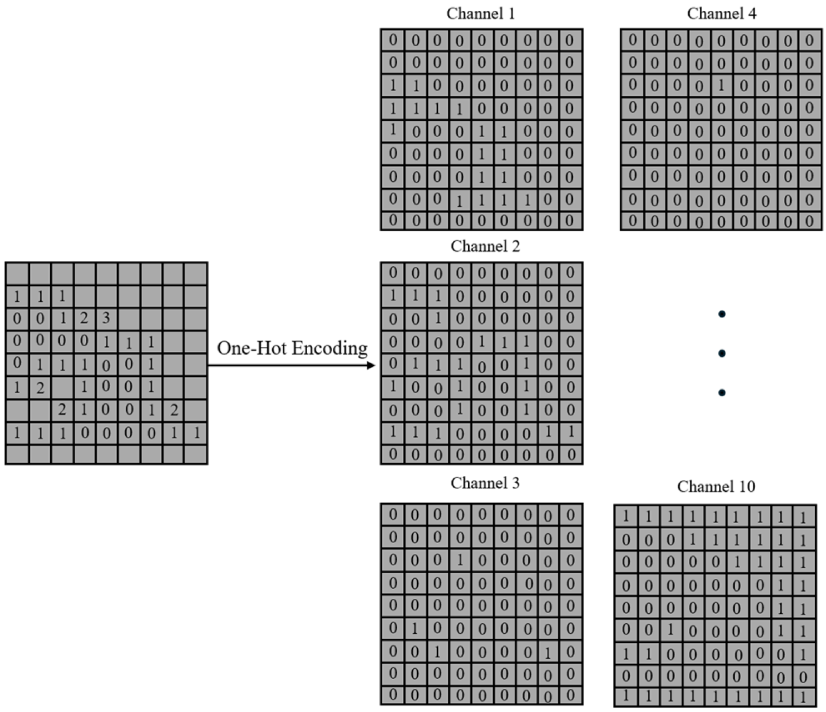


Figure 1. One-hot encoding representation of the minefield state.

2.2. Convolutional Neural Network Architecture

This study adopts a convolutional neural network architecture adapted from our previous work [9], consisting of three convolutional layers followed by three fully connected layers, as illustrated in Figure 2. For the 9 × 9 Minesweeper task, the model input and output dimensions are adjusted to match the problem size. To handle the increased modeling complexity caused by the substantially larger number of minefield states, the network is expanded relative to the previous model, and two dropout layers are introduced in the fully connected part to improve generalization performance and mitigate overfitting [17].

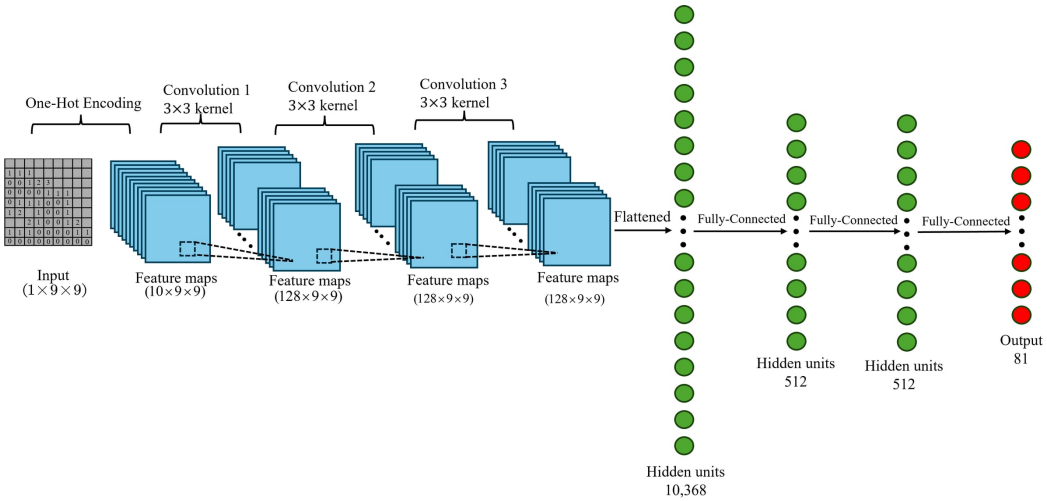


Figure 2. Neural network model of the Minesweeper agent.

- Input Layer: a tensor of shape (10, 9, 9), obtained by one-hot encoding the minefield state.
- Convolutional layer 1: 10 input channels, 128 output channels, with a 3 × 3 kernel.
- Convolutional layer 2: 128 input channels, 128 output channels, with a 3 × 3 kernel.
- Convolutional layer 3: 128 input channels, 128 output channels, with a 3 × 3 kernel.
- Fully connected layer 1: input size 10,368 (the convolutional output has shape (128, 9, 9), which flattens to 128 × 9 × 9 = 10,368), output size 512, with ReLU activation and dropout at a rate of 0.3.
- Fully connected layer 2: input size 512, output size 512, with ReLU activation and dropout at a rate of 0.3.
- Fully connected layer 3: input size 512 and output size 81, producing one raw mine logit for each board cell. The corresponding mine probabilities can be obtained by applying the sigmoid function.

The two dropout layers are enabled or disabled according to the experimental configuration. When enabled, the dropout rate is set to 0.3.

2.3. Dynamic Learning Rate Scheduling

As grid size and mine density increase, the state complexity of Minesweeper grows substantially, creating a challenge for a fixed learning rate, which is difficult to reconcile with optimization requirements throughout the full training cycle. Early in training, the primary objective is exploration: a relatively large learning rate promotes exploration of the parameter space and helps identify promising optimization directions, enabling the win rate to increase rapidly. A learning rate that is too small at this stage may result in very slow learning or convergence to a local optimum. Later in training, the emphasis shifts to refinement. A smaller learning rate enables fine-grained parameter updates and allows the model to approach its performance limit, whereas an excessively large learning rate can cause repeated oscillation around suboptimal solutions and prevent stable convergence.

We employ cosine annealing to adjust the learning rate during training [18]. The learning rate follows a cosine trajectory determined by the number of scheduler updates. It decreases during the first half-cycle and may increase again if scheduler updates continue beyond $T_{\max}$. The learning-rate schedule is expressed as follows:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left[1 + \cos\left(\frac{t}{T_{\max}}\pi\right)\right] \quad (1)$$

Equation (1) shows that, during training, the learning rate decays smoothly along a cosine curve from the initial learning rate $\eta_{\max}$ to the specified minimum learning rate $\eta_{\min}$ and increases again according to the cosine trajectory rather than remaining fixed at the minimum value. Here, t denotes the current scheduler update, and $T_{\max}$ specifies the cosine half-period measured in scheduler updates. This dynamic scheduling mechanism is conceptually consistent with a coarse-to-fine optimization strategy.

Implementation details of the learning-rate schedule: The Adam optimizer and CosineAnnealingLR scheduler were used for the cosine-annealing experiments, with $\eta_{\min} = 1 \times 10^{-7}$. For the initial 10-mine training, the initial learning rate was set to 1×10^{-4}, with $T_{\max} = 5000$. For the 18-mine from-scratch ablation experiments, the same initial learning rate was used, while $T_{\max}$ was increased to 7813 to align the cosine-annealing schedule with the approximately 8.0 million environment interactions. For the 10-to-14-mine and 14-to-18-mine transfer stages, as well as the corresponding direct-transfer and mixed-density adaptation experiments, a smaller initial learning rate of 5×10^{-6} and $T_{\max}$= 2500 were used. The scheduler advances once after a complete training-buffer optimization, rather than after each game action, epoch, or gradient update. Each full buffer of 1024 samples is optimized

for 10 epochs with a batch size of 32, resulting in 320 optimizer updates per scheduler step. Consequently, T_{max}= 5000 corresponds to 5,120,000 environment interactions and 1,600,000 optimizer updates, whereas T_{max}= 2500 corresponds to 2,560,000 environment interactions and 800,000 optimizer updates. For the from-scratch setting, T_{max} = 7813 corresponds to approximately 8.0 million environment interactions, given that the scheduler advances once per 1024 training samples. Throughout this study, training steps refer to environment interactions. T_{max} defines the cosine half-period in scheduler updates rather than the training termination point. The optimizer and scheduler are reinitialized at each SCL stage transition, while the corresponding reset settings for the other transfer-based strategies are described in Section 3.4.

As shown in Figure 3, compared with a fixed learning rate 1×10^{-4}, cosine annealing yields greater training stability and better convergence behavior on the 9 × 9 Minesweeper task with 10 mines. As shown in Figure 3a, with MSE loss, the peak win rate increases from 63% to 83%; with BCE loss in Figure 3b, it increases from 70% to 87%. The training curves also show that extending the fixed-learning-rate runs does not produce a comparable late-stage improvement. These observations support the use of cosine annealing in the present experiments. However, alternative learning-rate schedules were not evaluated, so the results do not establish that cosine annealing is universally superior to other scheduling strategies.

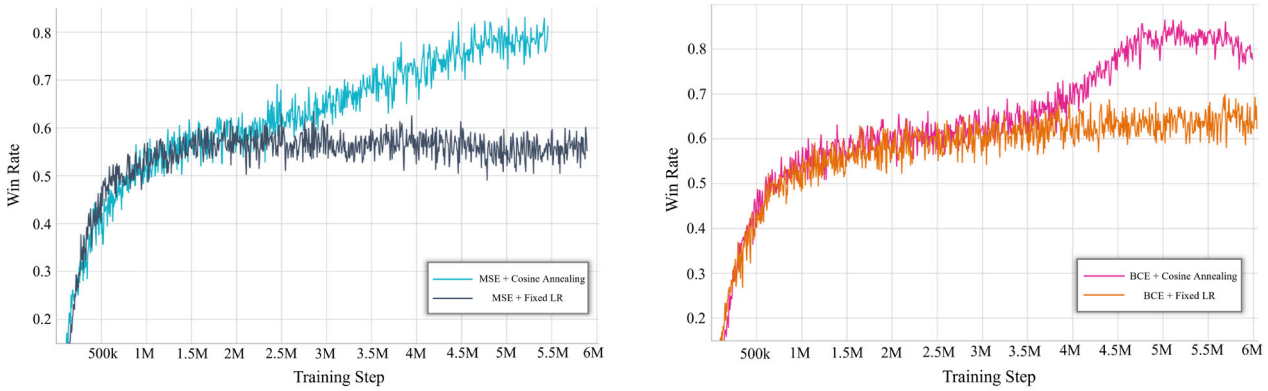


Figure 3. Comparison of training curves under fixed and cosine annealing learning rate schedules: (**a**) MSE loss; (**b**) BCE loss.

2.4. Data Augmentation

As grid size and mine density increase, the growing variety of minefield configurations poses additional challenges for model training and adaptation. To expose the CNN to a wider range of spatial configurations, we apply random transformations to both the input feature tensor and the corresponding label tensor [19], as illustrated in Figure 4. Specifically, horizontal and vertical reflections are applied independently with a probability of 0.5, followed by a rotation randomly selected from 0°, 90°, 180°, and 270° with equal probability. These transformations preserve the spatial relationships underlying Minesweeper and increase the diversity of configurations encountered during training. Their effects on test performance are evaluated in Section 3.2.

These transformations preserve the logical structure of Minesweeper on the fixed 9 × 9 square grid. Horizontal and vertical reflections, as well as rotations by multiples of 90°, preserve the adjacency relationships between cells, including those at the board boundaries. When the observable state and the corresponding mine-location label are transformed identically, each revealed numerical clue remains consistent with the transformed mine distribution. In our implementation, a single random combination of reflections and rotation

is sampled for each full training buffer and applied identically to all 1024 input samples and their corresponding mine-location labels. The transformed buffer is then reused for ten optimization epochs. A new transformation combination is sampled when the next buffer is processed. Thus, augmentation is performed at the buffer level rather than independently for each sample, mini-batch, or training epoch.

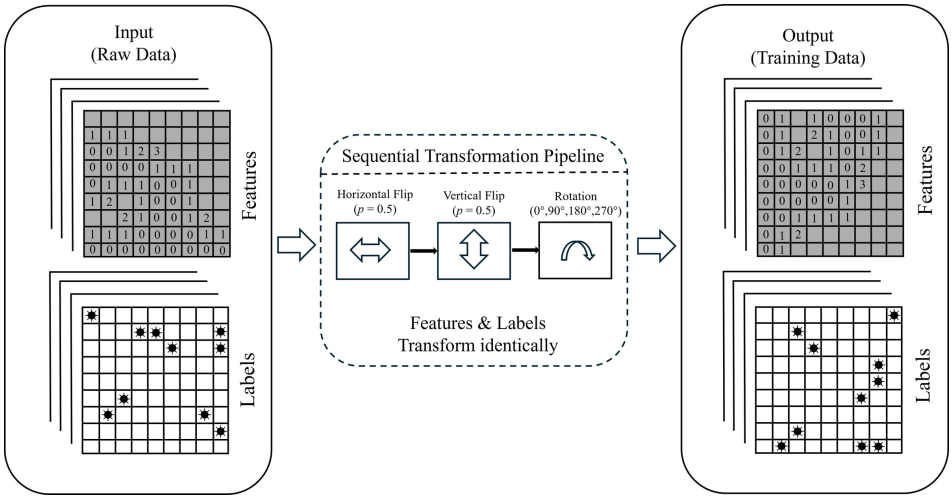


Figure 4. Data augmentation procedure for Minesweeper.

2.5. Improved Supervised Learning Algorithm

Our previous work systematically described the basic principles of training a Minesweeper agent with supervised learning, established an initial algorithmic framework, and verified its correctness and effectiveness in a simplified 6 × 6 Minesweeper environment with 4 mines [9]. However, extending the task to a 9 × 9 grid substantially increases the number of possible minefield configurations and poses greater challenges for model training and adaptation.

To investigate this larger 9 × 9 setting, the existing CNN-based supervised agent [9] is trained with cosine learning-rate scheduling and geometric data augmentation. The data-collection and optimization procedure actually used by the agent is detailed below and summarized in Algorithm 1.

Training states are collected online through actual interaction between the CNN agent and a randomly generated minefield. A new board is created with the configured mine count, and the first left-click is made safe: if the initially selected cell is a mine, the entire minefield is regenerated until that cell is safe; a zero-valued opening is not guaranteed. At every action, the model outputs 81 raw mine logits; revealed cells are excluded from the candidate set, and the unrevealed cell with the smallest logit is selected. Because the sigmoid is strictly increasing, this is equivalent to selecting the smallest predicted mine probability. The interaction policy does not use explicit epsilon-greedy exploration, a symbolic inference engine, or mine flags. During training-data collection, dropout is enabled only in configurations that use dropout and may introduce stochasticity into action selection.

After each action, the updated observable board state is paired with the complete underlying mine map supplied by the simulator. The mine map serves only as the supervised training label and is not available to the action-selection policy. Each recorded environment interaction contributes one state–label pair, including interactions that terminate an episode. The initial all-unrevealed state is not independently stored at game reset.

Training samples are collected into a buffer of 1024 entries. Once the buffer is full, a geometric transformation is applied to the input–label pairs if data augmentation is enabled. The resulting buffer is then optimized for ten epochs with a batch size of 32.

This produces 320 optimizer updates per complete buffer, after which the learning-rate scheduler advances once. The buffer is then cleared and replaced with samples collected from subsequent interactions, rather than being maintained as a cumulative replay dataset.

The standard optimization objective is unweighted BCEWithLogitsLoss averaged over all 81 board cells. No visibility mask is supplied to the loss function: both revealed and unrevealed positions contribute to supervision, including terminal-state samples. This full-board supervision is distinct from the action mask, which excludes already revealed positions only when choosing the next move. In the reported training curves, one step denotes one environment action rather than one gradient update.

Algorithm 1 Training a Minesweeper agent based on supervised learning with cosine annealing learning rate scheduling and data augmentation

Input: Encoded minesweeper grid state
Output: Deep neural network parameters θ

Initialize a training buffer D with capacity $N = 1024$
Randomly initialize the neural network θ or load pretrained weights
Set learning rate parameters $\eta_{\max}$, $\eta_{\min}$ and $T_{\max}$
repeat
    Retrieve the current grid state s
    Select $a = \arg\min_{i \in U(s)} f(s, \theta)_i$
    Execute action a and obtain the updated state s'
    Retrieve the ground-truth label y
    Store $\langle s', y \rangle$ in D
    if s' is a terminal state **then**
        Reset the game
    end if
    if the size of D reaches *sample_size* **then**
        Data augmentation: transform D to D_{aug}
        Optimize the neural network
        for each training epoch epoch = 1 to *num_epochs* **do**
            Shuffle D_{aug} and divide it into batches of size batch_size
            $D_{aug} = \{B_1, B_2, \ldots, B_m\}$
            for each batch B_i in D_{aug} **do**
                for each sample $\langle s_j, y_j \rangle$ in B_i **do**
                    Predict the mine probability $p_{j,k}$ for each grid cell using the model $p_{j,k} = \sigma(f_\theta(s_j)_k), k = 1, 2, \ldots, MN$
                    Compute the sample loss using binary cross-entropy (BCE)
                    $L(y_j, p_j) = -\frac{1}{MN} \sum_{k=1}^{MN} \left[y_{j,k} \log(p_{j,k}) + (1 - y_{j,k}) \log(1 - p_{j,k}) \right]$
                end for
                Compute the average loss for the batch
                $L(\theta) = \frac{1}{batch_size} \sum_{j=1}^{batch_size} L(y_j, p_j)$
                Update the model parameters θ using the Adam optimizer based on $L(\theta)$
            end for
        end for
            Update the learning rate using a cosine annealing schedule
        Clear training-buffer D
    end if
Until the termination criterion is met

2.6. Staged Curriculum Learning Algorithm

2.6.1. Problem Description and Motivation

The staged curriculum learning (SCL) protocol transfers a CNN across a manually specified sequence of increasingly dense 9 × 9 Minesweeper environments. The same network architecture and board dimensions are retained while the number of mines increases. Unlike an adaptive curriculum controller, the implemented protocol consists of separate training runs; progression to the next difficulty is performed by loading the checkpoint selected from the preceding run. This design evaluates whether features learned at lower mine densities support adaptation to harder tasks.

2.6.2. Stage Schedule

The training process is divided into K stages and represented by the sequence

$$S = \{m_1, m_2 \ldots m_K\} \tag{2}$$

where m_k denotes the number of mines at stage k. The number of mines increases monotonically across stages, i.e.,

$$m_1 < m_2 < \ldots < m_K \tag{3}$$

The investigated stage sequence is 10 → 14 → 18 → 22 mines. Each stage uses the same 9 × 9 input/output dimensions, enabling direct transfer of the CNN weights. Unlike adaptive self-paced approaches that adjust the training-task distribution during learning [20], the curriculum ordering investigated in this study is manually specified according to mine count.

2.6.3. Parameter Transfer

The network parameters at stage k are initialized using the training result from the previous stage rather than random initialization:

$$\theta_k^{init} = \theta_{k-1}^{*} \tag{4}$$

where θ_{k-1}^{*} denotes the network parameters obtained after completing stage $k-1$. In this way, the model reuses parameters learned at the preceding stage and continues training on a more difficult task. This parameter-transfer procedure is intended to facilitate adaptation across increasing mine densities [21]. The complete procedure is given in Algorithm 2.

In practice, the curriculum stages were trained sequentially, with model parameters transferred between consecutive difficulty levels. The representative cross-density experiments reported in Section 3.3 used checkpoints selected according to the rolling win rate over the latest 500 completed training games. These earlier results are retained as representative evaluations of the staged training framework.

For the three-seed ablation study in Section 3.4, a different checkpoint-selection protocol was adopted. The 14-mine stage was completed using its predefined budget of 1.0 million environment interactions, and the model parameters at the end of this stage were directly transferred to the 18-mine stage. The final 18-mine checkpoint was then selected using a separate, fixed 18-mine validation set rather than online training win rates. At each stage transition, the Adam optimizer, cosine learning-rate scheduler, and training buffer were reinitialized while retaining the transferred model parameters. No automatic win-rate threshold was used to trigger stage transitions.

Algorithm 2 Training a Minesweeper agent using staged curriculum learning

Input: Stage Schedule $S = \{m_1, m_2 \ldots m_K\}$ (e.g., mine counts)
Input: Initial Neural Network Architecture $f(s; \theta)$
Output: Final Optimized Parameters θ_k^*

Initialize network parameters θ randomly
for each stage $k = 1$ to K **do**
    Set current task difficulty $D_{task} \leftarrow m_k$
    if $k > 1$ **then**
        $\theta \leftarrow \theta_{k-1}^*$ (Load the selected weights from the previous stage)
    else
        Keep θ as random initialization (for the first stage)
    end if
    Initialize empty Training Buffer D_k, Adam optimizer, and learning rate scheduler
        Initialize $P_{best} \leftarrow -\infty$
        repeat (Start training for stage k)
        Execute Supervised Training Procedure with
            - Current difficulty D_{task}
            - Current parameters θ
        Monitor online training performance
        if checkpoint selection is required, **then**
            Evaluate the current model according to the checkpoint-selection criterion of the experiment.
            Save the checkpoint if the selection metric improves.
        end if
    until the current training run is terminated
    Transfer the stage-end or selected model parameters according to the experimental protocol.
end for
return θ_k^*

3. Results

All model training and algorithm evaluations were performed locally. The experimental platform used Windows 11, Python 3.12, PyTorch 2.7.1, and CUDA 11.8. The hardware consisted of an NVIDIA GeForce RTX 4060 Laptop GPU with 8 GB of video memory, an Intel Core i9-14900HX CPU, and 16 GB of system memory.

For supervised learning, training samples were collected online into a buffer containing 1024 state–label pairs. Each full buffer was optimized for 10 epochs with a batch size of 32, resulting in 320 optimizer updates. The training interaction count refers to environment actions rather than gradient updates. During independent testing, all CNN models were evaluated with dropout disabled.

3.1. Comparison and Selection of Loss Functions

We evaluate three loss functions—binary cross-entropy (BCE), weighted binary cross-entropy (WBCE), and focal loss (FL)—on the standard Minesweeper environment with a 9×9 grid and 10 mines [22]. All three loss-function configurations were trained without dropout or data augmentation. Their mathematical formulations are given below:

$$L_{BCE}(y, p) = -[y \log(p) + (1 - y) \log(1 - p)] \tag{5}$$

$$L_{WBCE}(y, p) = -[\alpha y \log(p) + (1 - \alpha)(1 - y) \log(1 - p)] \tag{6}$$

$$L_{FL}(y, p) = -\left[\alpha (1 - p)^{\gamma} y \log(p) + (1 - \alpha) p^{\gamma} (1 - y) \log(1 - p)\right] \tag{7}$$

Here, $y \in \{0, 1\}$ is the class label, where 1 denotes a mine and 0 denotes a non-mine cell, and p is the predicted probability that the cell contains a mine. The coefficient α weights classes inversely according to class frequency and is used to address class imbalance. For a

9 × 9 Minesweeper problem with 10 mines, the global ratio of mine cells to non-mine cells is approximately 1:7, and α is therefore set to 0.875. The focusing parameter γ assigns greater weight to hard examples to address differences in prediction difficulty and is typically set to $\gamma = 2$. WBCE generalizes BCE, while FL further extends WBCE [22].

The model training curves and test win rates obtained with the three loss functions are shown in Figure 5 and Table 1, respectively. Under the same evaluation protocol, the implemented Single-Point (SP) baseline achieves a win rate of approximately 70–71%, whereas all three CNN configurations achieve higher test win rates. Specifically, BCE, weighted BCE (WBCE), and focal loss (FL) achieve test win rates of 80.14%, 78.65%, and 79.88%, respectively. Although standard BCE achieves the highest test win rate among the three configurations, the observed differences are relatively small and do not establish its statistical superiority. A noteworthy observation is that WBCE does not outperform standard BCE despite the global class imbalance in Minesweeper, suggesting that class weighting does not necessarily improve game-level performance. Similarly, FL does not provide an improvement under the investigated settings, indicating that its emphasis on difficult training examples does not necessarily translate into a higher game win rate. The possible relationship between class imbalance and loss-function performance is further discussed in Section 4.1. Given its competitive performance and the absence of additional class-weighting or focusing hyperparameters, standard BCE is adopted in subsequent experiments.

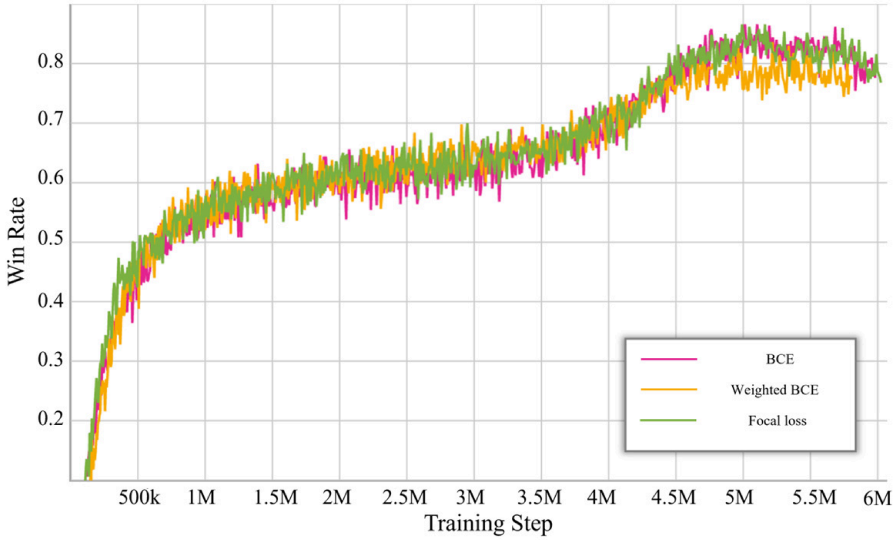


Figure 5. Comparison of model training curves under different loss functions.

Table 1. Comparison of model test win rates under different loss functions.

Loss Function	BCE	Weighted BCE	Focal Loss
Peak Training Win Rate	87%	86%	87%
Test Win Rate	80.14% ± 0.21 pp	78.65% ± 0.52 pp	79.88% ± 0.87 pp

The training win rates shown in Figure 5 reflect online performance during model optimization, whereas the test win rates in Table 1 are obtained from separate evaluations of trained models. Therefore, differences between peak training win rates and average test win rates should not be directly interpreted as evidence of overfitting or generalization gaps. Nevertheless, improving model generalization to unseen minefield configurations remains an important objective. The following section investigates whether dropout and data augmentation can further improve test performance.

3.2. Effects of Dropout and Data Augmentation on Test Performance

We investigate two approaches to improving model generalization performance: adding dropout layers with a dropout rate of 0.3 after the fully connected layers and applying data augmentation based on the spatial symmetry of Minesweeper. Figure 6 presents representative training curves, while Table 2 compares the test win rates of four configurations: the BCE baseline without dropout or data augmentation, dropout alone, data augmentation alone, and their combination.

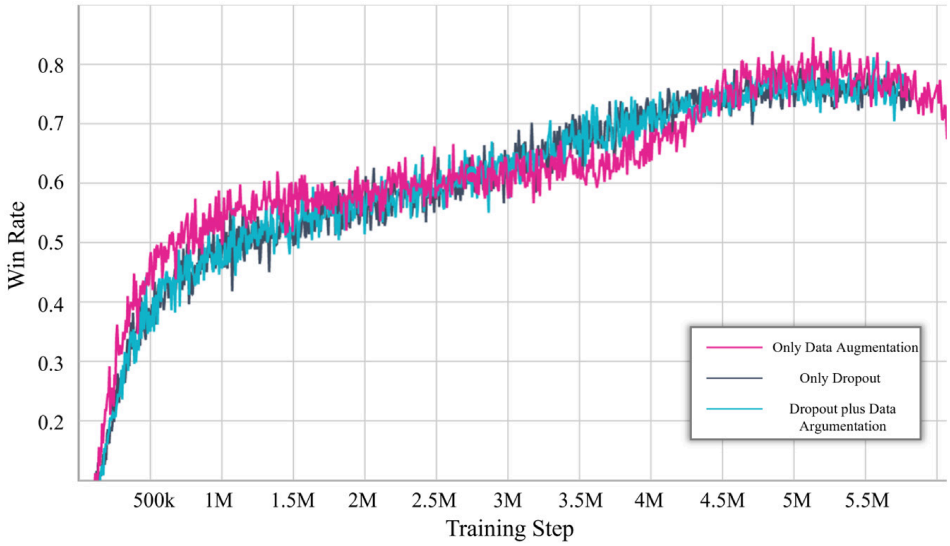


Figure 6. Comparison of model training curves under different configurations.

Table 2. Ablation study of dropout and data augmentation across three independent training runs.

Configuration.	Seed 161803	Seed 271828	Seed 314159	Mean ± SD
BCE Baseline	80.14%	79.30%	77.13%	78.86% ± 1.55 pp
Dropout Only	78.00%	80.04%	78.10%	78.71% ± 1.15 pp
Data Augmentation Only	76.90%	76.91%	77.79%	77.20% ± 0.51 pp
Dropout + Data Augmentation	85.44%	85.30%	84.75%	85.16% ± 0.36 pp

As shown in Table 2, the combined use of dropout and data augmentation achieves the highest mean test win rate of 85.16% ± 0.36 pp across three independent training runs. In comparison, the BCE baseline achieves 78.86% ± 1.55 pp, dropout alone achieves 78.71% ± 1.15 pp, and data augmentation alone achieves 77.20% ± 0.51 pp. The combined configuration outperforms the BCE baseline by 6.31 percentage points, indicating that using both techniques provides greater benefits than applying either technique individually under the evaluated settings. Two-sided Welch's t-tests based on the three independent training runs per configuration yielded p-values of 0.01563, 0.00622, and 0.000055 for the combined configuration versus the BCE baseline, dropout alone, and data augmentation alone, respectively. The corresponding Holm-adjusted p-values were 0.01563, 0.01245, and 0.000166, indicating statistically significant differences under the evaluated settings.

Dropout may reduce excessive dependence on particular learned features, whereas geometric data augmentation exposes the network to spatially transformed minefield configurations while preserving their underlying logical constraints. Neither dropout nor data augmentation alone consistently improves the mean test win rate over the BCE baseline under the present training conditions. In contrast, their combined use achieves the highest test win rate, indicating that the two techniques may complement each other in improving

the model's performance on unseen minefield configurations. However, the underlying mechanism cannot be determined from win-rate comparisons alone and is discussed further in Section 4.2.

It is important to distinguish online training performance from offline test performance. The training curves in Figure 6 reflect the performance of a model that continues to change during optimization, whereas the test results in Table 2 are obtained from separately evaluated checkpoints. Therefore, differences between peak online training win rates and offline test win rates cannot be interpreted directly as generalization gaps or quantitative measures of overfitting.

For Table 1, each loss-function configuration was trained once and evaluated in three separate test runs of 10,000 games each. For Table 2, each configuration was evaluated using three independently trained models, and the reported mean and standard deviation were calculated across these training runs. Thus, the standard deviations in Table 2 reflect training-seed variability rather than repeated evaluations of a single checkpoint.

3.3. Performance of Staged Curriculum Learning Across Mine Densities

To investigate the performance of staged curriculum learning (SCL) across different task difficulties, experiments were conducted on fixed 9 × 9 Minesweeper boards with 10, 14, 18, and 22 mines. These configurations correspond to mine densities of approximately 12.35%, 17.28%, 22.22%, and 27.16%, respectively. By keeping the board dimensions and network architecture unchanged, the number of mines serves as the variable controlling the training sequence. The investigated curriculum follows the order 10 → 14 → 18 → 22 mines, with the CNN parameters learned at each stage transferred to the subsequent stage. These configurations represent different mine densities on the same board size and are not assumed to be equivalent to the conventional Intermediate or Expert Minesweeper configurations.

The training cost of SCL includes all preceding stages rather than only the interactions performed at the target difficulty. The nominal experimental schedule allocates approximately 6.0 million environment interactions to the initial 10-mine training stage, followed by 1.0 million, 0.85 million, and 0.60 million interactions for adaptation to 14, 18, and 22 mines, respectively. Consequently, the nominal cumulative training budgets are approximately 7.00 million, 7.85 million, and 8.45 million environment interactions for the three target difficulties. These values describe the configured training budgets; the precise interaction count associated with an individual selected checkpoint may be smaller. The from-scratch models for the 14- and 22-mine tasks were trained for 7.00 million and 8.5 million environment interactions, respectively.

Table 3 summarizes the test results for SCL, training from scratch, and the Single-Point (SP) baseline. Each method was evaluated over 10,000 complete games at each mine density. The first click was fixed at the top-left cell and guaranteed to be safe, although a zero-valued opening was not guaranteed. Wilson 95% confidence intervals are reported in parentheses. The CNN results in Table 3 correspond to representative training runs, whereas the three-seed comparison on the 18-mine task is presented separately in Table 4. The 18-mine CNN results in Table 3 were obtained from earlier representative training runs using the original checkpoint-selection procedure. They are therefore distinct from the independently retrained, validation-selected models reported in Table 4.

Table 3. Comparison of test win rates and Wilson 95% confidence intervals across different mine densities.

Method	From Scratch	SCL	SP
14 mines	45.35% (44.38–46.33%)	55.85% (54.87–56.82%)	31.50% (30.60–32.42%)
18 mines	10.19% (9.61–10.80%)	20.38% (19.60–21.18%)	7.06% (6.57–7.58%)
22 mines	≈0.00% (0.00–0.04%)	4.00% (3.63–4.40%)	0.80% (0.64–0.99%)

Table 4. Ablation study of training strategies on the 9 × 9 Minesweeper task with 18 mines.

Strategy	Seed 161803	Seed 271828	Seed 314159	Mean ± SD
SCL	22.60%	22.20%	20.15%	21.65% ± 1.31 pp
Mixed Adaptation	21.09%	21.40%	20.07%	20.85% ± 0.70 pp
Direct Transfer	17.10%	16.73%	17.42%	17.08% ± 0.35 pp
From Scratch	15.76%	16.57%	14.45%	15.59% ± 1.07 pp

For the 14-mine configuration, SCL achieves a test win rate of 55.85%, outperforming training from scratch by 10.50 percentage points and the Single-Point (SP) baseline by 24.35 percentage points. This result suggests that knowledge acquired from the 10-mine task can provide a useful initialization for learning in a moderately denser environment.

As the number of mines increases to 18, the test win rates of all three methods decrease substantially, reflecting the increased challenge of the task. Nevertheless, SCL maintains a win rate of 20.38%, compared with 10.19% for training from scratch and 7.06% for the SP baseline. The SCL model therefore achieves a 10.19-percentage-point improvement over the evaluated from-scratch model. These results indicate that progressive adaptation can be beneficial when transferring a CNN-based Minesweeper agent to higher mine densities.

For the 22-mine configuration, SCL achieves a win rate of 4.00%, while the SP baseline achieves 0.80% and the evaluated from-scratch model records no successful games. Although SCL retains a relative advantage in the reported experiments, its absolute win rate remains low, demonstrating that reliable decision-making in highly dense minefields is still challenging. Thus, the result should be interpreted as evidence of limited successful adaptation rather than complete mastery of the task.

Overall, SCL achieves higher reported test win rates than the two baseline methods across the investigated mine densities. However, the curriculum requires training on preceding tasks, and its cumulative training cost must be considered when assessing computational efficiency. To further examine whether the performance advantage is associated with the staged organization of training, Section 3.4 compares SCL with direct transfer and mixed-density adaptation under the 18-mine configuration.

3.4. Ablation Study of Curriculum Organization

To further investigate the effectiveness of the proposed staged curriculum learning (SCL) strategy, we conducted an ablation study on the 9 × 9 Minesweeper task with 18 mines. Four training strategies were considered: training from scratch, direct transfer, mixed-density adaptation, and SCL. Training from scratch initializes the CNN randomly and trains it directly on the 18-mine task. Direct transfer initializes the model using pretrained weights from the 10-mine task and then performs adaptation directly on 18 mines, without an intermediate stage. Mixed-density adaptation also starts from the 10-mine pretrained model but interleaves training episodes from the 14-mine and 18-mine tasks. In contrast,

SCL follows the sequential training schedule 10 → 14 → 18, allowing the model to complete the intermediate stage before adapting to the target difficulty.

As a zero-shot reference, the three pretrained 10-mine models were evaluated directly on the 18-mine task without additional training. They achieved a mean test win rate of 5.12% ± 0.58 pp, indicating that subsequent adaptation training is important for improving performance at higher mine densities.

For the transfer-based strategies, the nominal adaptation budget was 1.85 million environment interactions after the initial 10-mine pretraining stage. In SCL, 1.00 million interactions were allocated to the 14-mine stage, followed by 0.85 million interactions on the 18-mine task. Mixed-density adaptation used the same difficulty-specific interaction allocations but interleaved the two tasks during training instead of separating them into consecutive stages. Direct transfer allocated the entire 1.85 million adaptation interactions to the 18-mine task. The three strategies therefore shared the same nominal cumulative budget of approximately 7.85 million interactions, including the initial 10-mine pretraining. The from-scratch model was trained directly on the 18-mine task for approximately 8.0 million environment interactions. Its nominal total interaction budget was therefore broadly comparable to the approximately 7.85 million cumulative interactions allocated to the three transfer-based strategies. For the three transfer-based strategies, the adaptation stages used an initial learning rate of 5×10^{-6} with $T_{max} = 2500$. To ensure comparable learning-rate schedules, Adam and the scheduler were reinitialized after 1.00 million adaptation interactions in all three strategies. Training from scratch retained the initial learning rate of 10^{-4} and $T_{max} = 7813$, with the latter adjusted to match the approximately 8.0-million-interaction training budget.

For the ablation study, checkpoint selection was based on a separate validation set consisting of 1000 fixed 9 × 9 Minesweeper boards with 18 mines. The same validation boards were used across all four strategies and three training seeds. During validation, dropout was disabled, and no gradient updates or training-buffer modifications were performed. Validation was performed every 30,000 environment interactions, with an additional evaluation at the end of each training budget. The checkpoint with the highest validation win rate was retained. In the event of a tie, the earlier checkpoint was selected.

For SCL, the model parameters at the end of the 1.00-million-interaction 14-mine stage were directly transferred to the 18-mine stage. For SCL, direct transfer, and mixed-density adaptation, final checkpoint selection was restricted to the last 0.85 million adaptation interactions, using the same 18-mine validation task and candidate-selection window. For training from scratch, checkpoints were evaluated throughout the 8.0-million-interaction training budget.

After checkpoint selection, each model was evaluated on a separate, untouched test set of 10,000 fixed 18-mine boards. The validation and test sets were generated independently, and the same test boards were used across strategies and training seeds. The first click was fixed at the top-left cell and guaranteed to be safe. The test set was not used for training, hyperparameter adjustment, or checkpoint selection.

Each strategy was trained with three independent random seeds, and the selected checkpoint from each run was evaluated on 10,000 independent 18-mine games. Table 4 reports the mean test win rate and standard deviation across the three training seeds.

As shown in Table 4 and Figure 7, SCL achieves the highest mean test win rate of 21.65% ± 1.31 pp across three independent training seeds. Mixed-density adaptation achieves 20.85% ± 0.70 pp, followed by direct transfer at 17.08% ± 0.35 pp and training from scratch at 15.59% ± 1.07 pp. SCL outperforms training from scratch by 6.06 percentage points and direct transfer by 4.57 percentage points, while its advantage over mixed-density

adaptation is smaller, at 0.80 percentage points. SCL achieves a higher test win rate than mixed-density adaptation in all three training seeds.

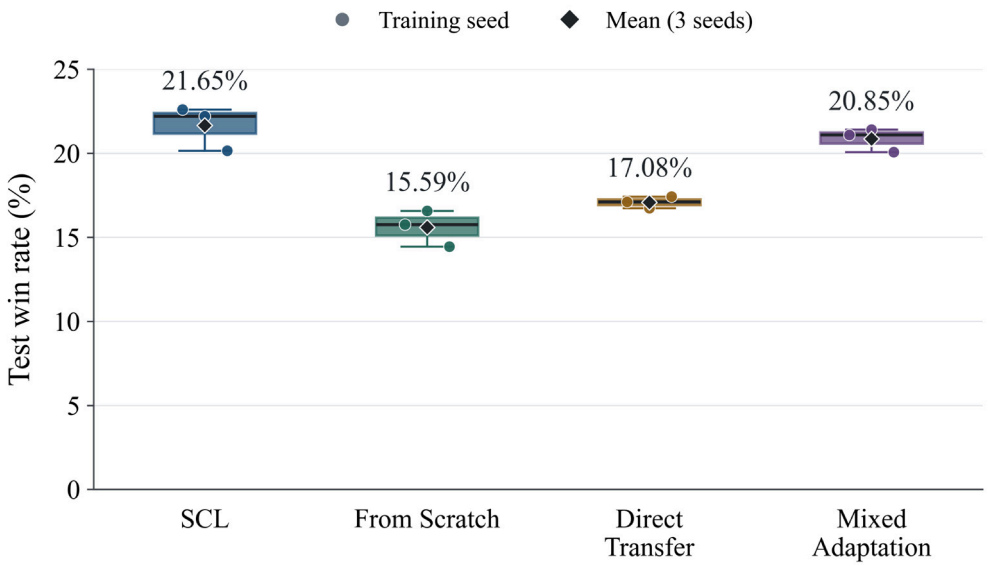


Figure 7. Comparison of test win rates among different training strategies.

Two-sided Welch's *t*-tests based on the three independently trained models per strategy yielded *p*-values of 0.00395, 0.02081, and 0.42112 for SCL versus training from scratch, direct transfer, and mixed-density adaptation, respectively. The corresponding Holm-adjusted *p*-values were 0.01186, 0.04162, and 0.42112. Thus, SCL achieves statistically significant improvements over training from scratch and direct transfer under the evaluated settings, whereas its difference from mixed-density adaptation is not statistically significant. These results support the effectiveness of the evaluated staged training procedure and demonstrate its competitive performance relative to mixed-density adaptation.

3.5. Comparison with a Constraint-Based Solver

To further evaluate the proposed SCL framework against more advanced traditional Minesweeper-solving methods, we introduce a constraint satisfaction problem (CSP) solver as an additional baseline. Unlike the Single-Point (SP) algorithm, which primarily relies on elementary local deduction rules, the CSP solver combines deterministic constraint propagation, enumeration of feasible mine configurations, and probability-based action selection. When no safe move can be identified through logical deduction, the solver estimates mine probabilities using configurations consistent with the revealed numerical clues and the total number of mines, and selects the unrevealed cell with the lowest estimated risk. To control computational complexity, a maximum of 20,000 search iterations is allowed per decision, with a heuristic fallback strategy applied when the enumeration budget is exceeded.

The CSP solver was evaluated on 9 × 9 Minesweeper boards containing 10, 14, 18, and 22 mines, with 10,000 games conducted for each configuration. The evaluation followed the same game rules and first-click safety condition used in the CNN experiments, with the first click fixed at the top-left cell and guaranteed to be safe. Table 5 presents the comparative results for the 10-, 14-, 18-, and 22-mine tasks, for which corresponding SCL, SP and CSP results are available.

Table 5. Comparison of test win rates and Wilson 95% confidence intervals among SCL, SP, and CSP across different mine densities.

Mines	SP Win Rate	SCL Win Rate	CSP Win Rate
10	70.69% (69.79–71.57%)	85.44% (84.74–86.12%)	90.04% (89.44–90.61%)
14	31.50% (30.60–32.42%)	55.85% (54.87–56.82%)	74.22% (73.35–75.07%)
18	7.06% (6.57–7.58%)	20.38% (19.60–21.18%)	47.17% (46.19–48.15%)
22	0.80% (0.64–0.99%)	4.00% (3.63–4.40%)	20.73% (19.95–21.54%)

Values are test win rates, with Wilson 95% confidence intervals shown in parentheses. Each result is based on one evaluation of 10,000 independent test games. The 10-mine SCL result of 85.44% corresponds to the seed-161803 result for the combined dropout and data augmentation configuration in Table 2, rather than the three-seed mean of 85.16%. The intervals reflect evaluation uncertainty rather than variability across independent training seeds. The 18-mine SCL result corresponds to the earlier representative evaluation in Table 3, rather than the validation-selected three-seed ablation results in Table 4.

As shown in Table 5, the evaluated CNN models achieve higher test win rates than the SP baseline across all four mine densities, whereas the CSP solver consistently achieves the highest win rates. However, the CSP solver achieves substantially higher win rates than SCL. For the 14-mine task, CSP reaches 74.22%, compared with 55.85% for SCL. At 18 mines, CSP achieves 47.17%, exceeding the SCL result of 20.38% by 26.79 percentage points. For the more challenging 22-mine task, the win rates decrease to 20.73% and 4.00%, respectively, indicating that reliable decision making remains difficult for the learning-based agent in highly dense minefields.

The curriculum ablation study in Section 3.4 demonstrates that SCL is a competitive training strategy among the evaluated CNN-based approaches. Nevertheless, its improved performance does not eliminate the substantial gap between learning-based and constraint-based methods. The CSP solver explicitly incorporates numerical clues and global mine-count constraints to identify logically safe moves and estimate the risks of uncertain actions. In contrast, the CNN learns to predict mine locations from observable board states without explicitly enforcing these constraints.

Across the 40,000 CSP test games, the search limit was reached 61 times in 20 games. In these cases, the solver used the heuristic fallback strategy to select the next move. Overall, the results indicate that SCL improves the adaptation of CNN-based Minesweeper agents to higher mine densities through progressive training and parameter transfer, while explicit constraint-based reasoning retains a substantial advantage in absolute solving performance. Combining staged curriculum learning with logical constraint inference may therefore offer a promising direction for future research.

4. Discussion

4.1. Is Minesweeper a Class-Imbalanced Task?

For a 9 × 9 Minesweeper board with 10 mines, the complete mine map contains exactly 10 positive labels and 71 negative labels, corresponding to a mine proportion of approximately 12.35%. Therefore, from the perspective of full-board mine classification, Minesweeper is clearly a class-imbalanced task. In our implementation, the BCE loss is computed over all 81 positions, including both revealed and unrevealed cells. Consequently, this global class imbalance is directly relevant to the supervised learning objective.

However, the decision-making process in Minesweeper differs from full-board classification. When a player reveals a cell with value 0, the game automatically reveals the connected region of zero-valued cells and the numbered cells along its boundary. This

mechanism is known as "flood fill". As the game progresses, the revealed numerical clues form a dynamically changing frontier. Unrevealed cells adjacent to this frontier constitute the frontier region, whereas unrevealed cells farther away belong to the non-frontier region. The frontier provides local constraints that can be used to assess the locations of hidden mines, although not every cell in this region can necessarily be determined to be safe or mined.

The distribution of mines and safe cells within the frontier region may differ substantially from that of the complete minefield. Figure 8 illustrates a particular configuration in which the selected frontier region contains 10 unrevealed cells, including five mines and five safe cells, giving a local mine-to-safe ratio of 1:1. By comparison, the global ratio for the complete 9 × 9 minefield is 10:71. This example demonstrates that local and global class distributions can differ considerably, but it should not be interpreted as statistically representative of all game states. Furthermore, numerical clues from 1 to 8 do not occur with equal probability, as further discussed in Section 4.3. Therefore, the local mine-to-safe ratio cannot be reliably derived from an equal-probability assumption and generally varies with the minefield configuration and the progression of the game.

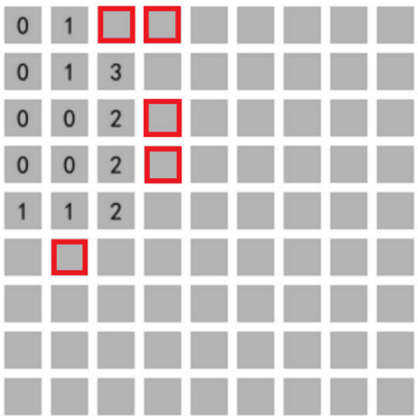


Figure 8. Example minefield state on a 9 × 9 grid with 10 mines; red boxes indicate mine cells.

This distinction between global supervision and local decision-making provides a possible perspective for interpreting the loss-function results. Although WBCE assigns greater importance to the minority class during training, the agent ultimately selects actions by comparing predicted mine risks among unrevealed cells rather than by maximizing full-board classification accuracy. Consequently, improving a class-weighted classification objective does not necessarily lead to a higher game win rate. This may help explain why WBCE does not outperform standard BCE in the present experiments. However, the current results do not establish that differences between global and local class distributions are the direct cause of the observed performance difference.

More generally, Minesweeper solving involves a sequence of decisions based on progressively updated observations. Each successful action may reveal additional numerical clues and change the frontier region. When flood fill occurs, multiple safe cells can be revealed automatically, providing additional information without requiring separate actions for each cell. The agent then evaluates the updated board state and selects its next action. Thus, the complete game can be viewed as a sequence of decisions under evolving local constraints. Although these constraints provide information for mine-risk prediction, the present CNN does not explicitly perform symbolic logical deduction. Instead, it learns to estimate mine locations from observable board states and uses these predictions to guide successive decisions.

4.2. A Recognition–Reasoning Perspective on the Joint Effect of Dropout and Data Augmentation

Minesweeper has a finite but extremely large state space. Nevertheless, the rules governing the game are inherently local: each revealed numbered cell specifies the number of mines among its adjacent cells. Complex minefield configurations can therefore be understood in terms of overlapping local patterns and the relationships among them. As the game progresses, newly revealed clues provide additional information, allowing the agent to update its assessment of the remaining unrevealed cells. This structure provides a useful basis for understanding Minesweeper solving from the complementary perspectives of pattern recognition and logical reasoning.

Within this conceptual framework, *recognition* refers to identifying relevant local patterns and spatial relationships in the observable minefield state, whereas *reasoning* refers to using the constraints associated with these patterns to infer mine locations, identify safe cells, or assess the risks of candidate actions. Recognition and reasoning are closely related: recognizing a familiar configuration helps identify the constraints relevant to a decision, while combining multiple constraints may reveal relationships that are not apparent from individual patterns. These two aspects provide a common perspective for analyzing different Minesweeper solvers, although their actual implementations differ considerably.

The Single-Point (SP), constraint satisfaction problem (CSP), and CNN-based approaches illustrate different ways of processing spatial information and numerical constraints. The SP algorithm applies predefined logical rules within small neighborhoods around revealed numbered cells, but typically does not jointly solve constraints from multiple overlapping neighborhoods. CSP-based solvers, in contrast, integrate constraints from multiple revealed clues and may incorporate the total number of remaining mines. By propagating constraints and examining feasible mine configurations, they can perform more extensive logical inference, although computational costs may increase with problem complexity. This distinction is reflected in the results in Section 3.5, where CSP achieves higher absolute win rates than the CNN-based agent.

The CNN-based supervised learning approach processes spatial information in a different manner. Convolutional kernels extract features from local neighborhoods, while successive layers integrate information across increasingly large receptive fields. In this sense, the CNN can learn representations involving relationships among multiple local configurations rather than relying exclusively on predefined rules applied to individual clue neighborhoods. Unlike SP and CSP, however, the CNN does not explicitly encode or execute Minesweeper deduction rules. Instead, it learns statistical relationships between observable minefield states and mine locations through supervised training. Its predictions may reflect regularities associated with numerical constraints, even though the model is not explicitly programmed to perform constraint propagation or combinatorial search. Thus, the CNN can be viewed as an alternative approach to learning and using spatial relationships, but its representational capacity should not be equated with verified logical reasoning ability.

This framework also provides a possible interpretation of the combined effects of dropout and data augmentation. Geometric data augmentation reflects and rotates minefield configurations while preserving adjacency relationships and numerical constraints. As a result, equivalent local patterns can appear in different spatial orientations during training. Exposure to these variations may help the CNN learn representations that are less dependent on the orientation of particular configurations. In terms of the proposed framework, data augmentation may therefore contribute primarily to the robustness of local pattern recognition.

Dropout, in contrast, randomly deactivates neural units during training and may discourage excessive dependence on particular activations or feature combinations. This regularization may help the network develop representations that are more robust when

information from multiple local patterns must be integrated to estimate mine risks. From the recognition–reasoning perspective, such robustness could support the learned use of relationships among local configurations. However, this interpretation does not imply that dropout directly improves logical reasoning ability or causes the CNN to acquire explicit deduction rules. Its effects may be explained by conventional regularization without invoking a distinct reasoning mechanism.

When dropout and data augmentation are combined, the network is exposed to spatially diverse configurations while being discouraged from relying excessively on particular feature combinations. These effects may complement each other, potentially improving the robustness of learned spatial relationships and their use in mine-risk prediction. This interpretation is consistent with the experimental results in Section 3.2, where the combined configuration achieves a mean test win rate of 85.16% across three independent training seeds, compared with 78.86% for the BCE baseline, 78.71% for dropout alone, and 77.20% for data augmentation alone. Nevertheless, these observations do not establish a statistically confirmed interaction between the two techniques or demonstrate that recognition and reasoning form a mutually reinforcing internal mechanism within the CNN.

The present experiments do not directly measure local-pattern recognition accuracy, multi-step logical deduction, or the internal reasoning processes of the CNN. Alternative explanations for the combined performance improvement, including complementary regularization effects and sensitivity to the selected training settings, remain possible. Future work could investigate these issues through controlled Minesweeper patterns, frontier-specific prediction tests, intermediate-feature analysis, and comparisons between learned predictions and explicit constraint-based deductions. Such investigations would help determine whether the proposed conceptual framework corresponds to identifiable computational mechanisms in CNN-based Minesweeper agents.

4.3. Clue-Value Distributions and Curriculum Design

To further investigate the rationale for staged curriculum learning, we analyze how mine density affects the distribution of numerical clues on a fixed 9 × 9 grid. Increasing the number of mines changes the local configurations encountered by the model without altering the input dimensions or the underlying rules of Minesweeper. Figure 9 illustrates representative minefields containing 10, 14, 18, and 22 mines, corresponding to mine densities of approximately 12.35%, 17.28%, 22.22%, and 27.16%, respectively.

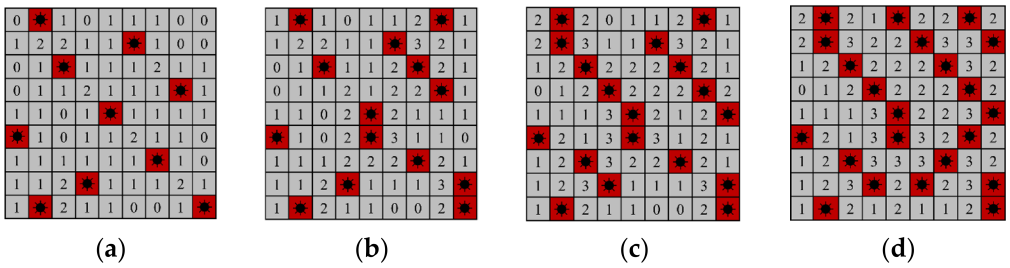


Figure 9. Example 9 × 9 Minesweeper grids under different mine densities: (**a**) 10 mines; (**b**) 14 mines; (**c**) 18 mines; (**d**) 22 mines.

To quantify these changes, a Monte Carlo simulation was conducted using 10,000 randomly generated minefields for each mine count, yielding a total of 40,000 boards. The NumPy random seed was fixed at 2026. For each board, the specified number of mines was placed at distinct randomly selected positions, and the numerical clue of every non-mine cell was calculated according to the number of adjacent mines. The simulation analyzed complete minefields without executing any game actions. For each mine count, the occurrences of

clue values from 0 to 8 were accumulated over all generated boards and divided by the total number of grid cells (10,000 × 81). Mine cells were included in the denominator but excluded from the clue counts. Accordingly, the sum of the reported clue proportions equals the proportion of non-mine cells rather than 100%. For visualization, clue values from 5 to 8 were grouped into a single category (≥5).

As shown in Figure 10, increasing mine density produces a progressive shift in the clue-value distribution. At lower mine densities, zero-valued cells are relatively common, whereas higher mine densities are associated with a greater occurrence of larger numerical clues. This change reflects the increasing number of mines surrounding safe cells and indicates that the frequencies of local clue configurations vary across tasks.

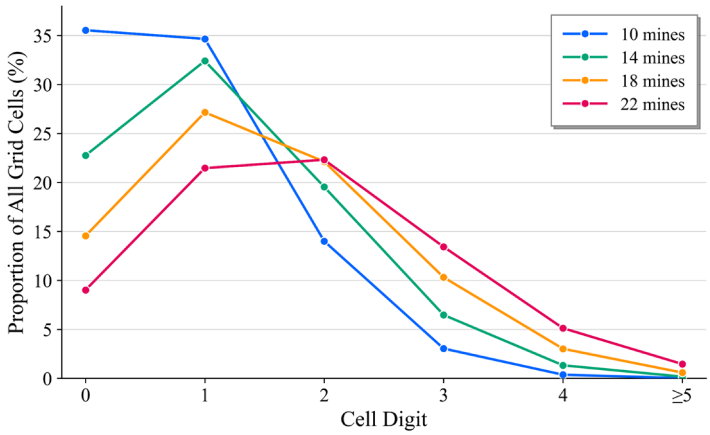


Figure 10. Clue-value distributions on 9 × 9 Minesweeper grids with different numbers of mines based on Monte Carlo simulation.

These observations provide a basis for organizing training tasks according to mine density. Although the underlying Minesweeper rules remain unchanged, the model encounters different distributions of local patterns as the number of mines increases. The sequence 10 → 14 → 18 → 22 therefore provides a structured progression through increasingly difficult tasks, allowing the model to adapt to successive changes in the training distribution while retaining parameters learned in preceding stages. However, changes in clue-value distributions alone do not directly measure logical reasoning difficulty or establish the superiority of a particular curriculum organization.

An important distinction between SCL and mixed-density adaptation lies in the organization of intermediate learning tasks. SCL follows a stepwise progression in which the 14-mine task serves as an intermediate stage between the 10-mine source task and the 18-mine target task. This arrangement allows the model to focus on intermediate-density configurations before adapting to the more difficult target task. In contrast, mixed-density adaptation interleaves 14- and 18-mine tasks during training, exposing the model to both difficulty levels without separating them into successive stages. Thus, SCL and mixed-density adaptation represent two different ways of organizing knowledge transfer: sequential progression through intermediate difficulty levels and interleaved learning across multiple difficulty levels.

The ablation results demonstrate that SCL is an effective and competitive training strategy. It achieves statistically significant improvements over training from scratch and direct transfer, and numerically outperforms mixed-density adaptation across all three training seeds. Although the difference between SCL and mixed-density adaptation is not statistically significant, the results support stepwise curriculum learning as a viable approach to transferring knowledge across increasingly difficult Minesweeper tasks. Further investigations involving additional training seeds, different intermediate difficulty levels, and

alternative stage allocations would help clarify when sequential staging is preferable to mixed-density adaptation. Combining staged progression with mixed-difficulty training may also provide a promising direction for future research.

5. Conclusions

To address the expansion of the state space, increasing complexity of local reasoning, and difficulty of training from scratch in high-difficulty Minesweeper tasks, this study proposes a Minesweeper-solving framework based on staged curriculum learning. We first construct a CNN-based supervised learning model and combine one-hot state encoding, cosine annealing learning rate scheduling, dropout, and data augmentation to improve training stability and generalization performance. In the standard environment with a 9×9 grid and 10 mines, the combined configuration achieves a test win rate of $85.16\% \pm 0.36$ pp, clearly outperforming the SP baseline. The joint use of dropout and data augmentation yields better test performance than either technique alone under the evaluated settings.

Building on this model, we construct an easy-to-hard task sequence according to the number of mines and employ stage-to-stage parameter transfer to progressively adapt the CNN to tasks of increasing difficulty. The intermediate difficulty levels provide a structured, stepwise path for knowledge transfer between related Minesweeper tasks. Earlier representative evaluations on the 14-, 18-, and 22-mine tasks yielded test win rates of 55.85%, 20.38%, and 4.00%, respectively. In the three-seed ablation study on the 18-mine task, SCL achieved the highest mean test win rate of 21.65%, significantly outperforming training from scratch (15.59%) and direct transfer (17.08%). It also achieved numerically higher win rates than mixed-density adaptation (20.85%) across all three seeds, although the difference was not statistically significant. These findings demonstrate the effectiveness of the evaluated staged curriculum for knowledge transfer across increasing mine densities and establish SCL as a competitive alternative to mixed-density adaptation.

Nevertheless, the CSP solver achieves a higher win rate of 47.17% in the representative 18-mine comparison, indicating that explicit constraint-based reasoning remains more effective in absolute solving performance. The relatively low win rate on the 22-mine task also highlights the challenges of reliably solving high-density configurations. Thus, the principal contribution of SCL lies in investigating structured training and knowledge transfer across progressively more difficult tasks rather than outperforming specialized constraint-solving algorithms.

The underlying principle of SCL is not specific to Minesweeper. By organizing related tasks into progressively increasing difficulty levels and transferring learned parameters between consecutive stages, SCL provides a straightforward framework for exploiting shared knowledge across tasks. The findings of this study demonstrate the feasibility of this approach for CNN-based Minesweeper agents, while its generalizability to other learning problems requires further investigation. Future work will explore adaptive task-difficulty partitioning and stage-transition mechanisms, as well as hybrid schedules that combine staged progression with mixed-difficulty adaptation. In addition, integrating learned models with logical reasoning, probabilistic reasoning, and reinforcement learning may provide opportunities for further improving decision-making performance on challenging tasks.

Author Contributions: Conceptualization, W.W.; methodology, W.R.; funding acquisition, C.L.; writing—original draft, W.R.; writing—review and editing, C.L.; supervision, W.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Department of Education of Guangdong Province through the Young Innovative Talents Project in Guangdong Province, grant number 2019KQNCX013.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code and implementation materials associated with this study are available in a GitHub repository at https://github.com/Ruan-WH/minesweeper-cnn-code (accessed on 21 October 2026). The experimental results supporting the findings of this study are presented in the article. Further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kaye, R. Minesweeper is NP-complete. *Math. Intell.* **2000**, *22*, 9–15. [CrossRef]
2. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 4th ed.; Pearson: Hoboken, NJ, USA, 2021.
3. Li, Y.; Wang, H.; Zhang, C. Assessing logical puzzle solving in large language models: Insights from a Minesweeper case study. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2024), Mexico City, Mexico, 16–21 June 2024; pp. 59–81.
4. Studholme, C. Minesweeper as a Constraint Satisfaction Problem. 2000. Available online: https://www.cs.toronto.edu/cvs/minesweeper/minesweeper.pdf (accessed on 15 September 2026).
5. Buffet, O.; Lee, C.-S.; Lin, W.-T.; Teytuad, O. Optimistic Heuristics for MineSweeper. In *Advances in Intelligent Systems and Applications—Volume 1*; Chang, R.-S., Jain, L.C., Peng, S.-L., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 199–207.
6. Tu, J.; Li, T.; Chen, S.; Zu, C.; Gu, Z. Exploring Efficient Strategies for Minesweeper. In Proceedings of the AAAI-17 Workshop on What's Next for AI in Games? (WS-17-15), San Francisco, CA, USA, 4 February 2017; pp. 999–1005.
7. Liu, C.; Huang, S.; Gao, N.; Khalid, M.N.A.; Iida, H. A solver of single-agent stochastic puzzle: A case study with Minesweeper. *Knowl.-Based Syst.* **2022**, *246*, 108630. [CrossRef]
8. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
9. Wang, W.; Lei, C. Training a Minesweeper agent using a convolutional neural network. *Appl. Sci.* **2025**, *15*, 2490. [CrossRef]
10. Watkins, C.J.C.H.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
11. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
12. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009), Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.
13. Soviany, P.; Ionescu, R.T.; Rota, P.; Sebe, N. Curriculum learning: A survey. *Int. J. Comput. Vis.* **2022**, *130*, 1526–1565. [CrossRef]
14. Wang, X.; Chen, Y.; Zhu, W. A survey on curriculum learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 4555–4572. [CrossRef] [PubMed]
15. Narvekar, S.; Peng, B.; Leonetti, M.; Sinapov, J.; Taylor, M.E.; Stone, P. Curriculum learning for reinforcement learning domains: A framework and survey. *J. Mach. Learn. Res.* **2020**, *21*, 1–50. [CrossRef]
16. Jiang, M.; Grefenstette, E.; Rocktäschel, T. Prioritized level replay. In Proceedings of the 38th International Conference on Machine Learning (ICML 2021), Virtual, 18–24 July 2021; Volume 139, pp. 4940–4950.
17. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
18. Loshchilov, I.; Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In Proceedings of the International Conference on Learning Representations (ICLR 2017), Toulon, France, 24–26 April 2017.
19. Alomar, K.; Aysel, H.I.; Cai, X. Data augmentation in classification and segmentation: A survey and new strategies. *J. Imaging* **2023**, *9*, 46. [CrossRef] [PubMed]
20. Klink, P.; D'Eramo, C.; Peters, J.; Pajarinen, J. Self-paced deep reinforcement learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020), Virtual, 6–12 December 2020*; Curran Associates, Inc.: Red Hook, NY, USA, 2020; pp. 9216–9227.
21. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A comprehensive survey on transfer learning. *Proc. IEEE* **2021**, *109*, 43–76. [CrossRef]
22. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2999–3007.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.