IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/44AI/2024/4454124/
Vol. 2024 (22), Article ID 4454124

Data-Driven Optimization of Game Distribution: A Reinforcement Learning Approach*

Diana BRATIĆ, Tvrtko GRABARIĆ and Mirko PALIĆ

University of Zagreb, Faculty of Graphic Arts, Zagreb, Croatia

Correspondence should be addressed to: Diana BRATIĆ, diana.bratic@grf.unizg.hr

* Presented at the 44th IBIMA International Conference, 27-28 November 2024 Granada, Spain

Abstract

This study investigates the application of Reinforcement Learning models, in particular Deep Q-Networks (DQN), for optimizing distribution strategies in the gaming industry. The motivation for this research stems from the growing need for adaptive and efficient distribution methods in dynamic markets and increasingly complex user expectations. Existing approaches rely predominantly on static models that are often unable to adapt to rapid changes in user behavior and market trends, highlighting a critical gap in the literature for flexible and adaptive methods like Reinforcement Learning.
Methodologically, the study uses a DQN model that utilizes relevant data from platforms such as Steam and Epic Games Store, including variables such as game pricing, number of recommendations and reviews, and playtime metrics. The data was pre-processed and normalized to ensure the stability of the model. The evaluation was based on key performance indicators such as RMSE and MAPE.
The results demonstrate the high predictive accuracy of the model (RMSE: 4.31e-3), despite the challenges associated with data distribution (high MAPE). The model successfully identified optimal distribution strategies while reducing costs and enhancing user engagement. This underlines the potential of Reinforcement Learning in adapting distribution strategies in real time. This study not only confirms the applicability of Reinforcement Learning in optimizing game distribution, but also highlights the need to further enhancements in data quality to increase the performance of the model.

Keywords: Reinforcement Learning, Deep Q-Network, distribution optimization, gaming industry.

Introduction

In today's digital landscape, the gaming industry is undergoing rapid transformations in both distribution channels and user behavior. With increasing competition and the emergence of new platforms such as Steam, Epic Games Store and mobile apps, it is crucial to adapt distribution strategies to optimize user reach and engagement. Digital distribution has revolutionized the way games are delivered to users, forcing companies to constantly look for ways to improve their approaches (Hu et al., 2023). This creates a need for new technologies such as artificial intelligence, especially Reinforcement Learning (RL), which can help optimize distribution strategies (Ali and Baizal, 2023).

Cite this Article as: Diana BRATIĆ, Tvrtko GRABARIĆ and Mirko PALIĆ, Vol. 2024 (22) "Data-Driven Optimization of Game Distribution: A Reinforcement Learning Approach " Communications of International Proceedings, Vol. 2024 (22), Article ID 4454124, https://doi.org/10.5171/2024.4454124

In Reinforcement Learning (RL), a branch of machine learning, a model is trained to make optimal decisions in a given environment through iterative processes of rewards and penalities. RL is increasingly used in dynamic industries as it enables real-time strategy adjustments, adaptation to changing conditions and optimization of resources (Alharin, Doan and Sartipi, 2020). In the context of the gaming industry, RL can help companies analyze user behavior feedback and optimize distribution channels based on engagement data, usage frequency and user preferences. In this way, game distribution can become more efficient, targeted and user-centric, directly impacting satisfaction and engagement (Schindler, Uray and Huber, 2023).

Existing distribution strategies in the gaming industry are often based on static models that lack the flexibility to react to rapid changes in the market and user expectations. For example, companies may invest equally in all channels, even though certain channels deliver better results while others have lower user engagement. Such an approach leads to unnecessary costs and lower revenues, and users miss out on a personalized experience that could keep them loyal to a particular platform (Tian, Pan, and Liu, 2024). By introducing RL models, it is possible to develop adaptive distribution strategies based on real-world data and optimize the approach for each user or user group.

The aim of this study is to develop an RL-based model that optimizes distribution strategies in the gaming industry by analyzing user behavior and feedback from different distribution channels. RL enables more precise segmentation and content customization in real time, reducing distribution costs while maximizing user reach, engagement and satisfaction. The development of this model involves several key phases: analyzing distribution channels, creating an RL model tailored to industry specifics, and testing and simulating the model to evaluate its ability to increase reach and optimize investment per channel.

One of the major challenges in this study is the fact that distribution channels in the gaming industry are not static. Each platform and channel has unique audiences, characteristics and recommendation algorithms that influence how games are distributed and promoted. Understanding these channels and developing tailored approaches requires specific data analysis, for which RL is particularly suitable, as it enables automatic learning and adaptation. Using methods such as Q-Learning or Deep Q-Networks, the model can learn from feedback such as download numbers, engagement rates and user feedback to make more accurate predictions about optimal strategies (Yu and He, 2019).

The distribution of games through platforms such as Steam, Epic Games Store, Google Play and App Store often relies on recommendation algorithms that are optimized for user interests and history. However, these platform algorithms are not always fully aligned with the goals of game publishers. An RL model can serve as an internal tool for optimizing channel-related investments and segmenting users based on preferences, even if the platform is outside the publisher's direct control. This gives publishers a powerful tool for predicting user needs and optimizing investments in promotional campaigns that attract and retain users on specific channels.

In addition, RL enables continuous adaptation of the strategy as the market evolves. For example, if a particular channel is gaining popularity with users, an RL model can automatically redirect resources to that channel to optimize engagement. If users are moving away from a particular channel, the model can reduce investment in that channel to minimize losses and maximize distribution efficiency.

Ultimately, exploring the application of RL in optimizing distribution strategies in the gaming industry has the potential to significantly increase channel efficiency, improve user experience and maximize return on investment. The aim of this study is to provide a comprehensive overview and analysis of RL models tailored to the distribution of digital games and to provide concrete recommendations for the further application of this technology in the gaming industry.

Theoretical Background of Modeling

Distribution Channels in the Gaming Industry

Distribution channels in the gaming industry have changed significantly over the last ten years, mainly due to the spread of digital distribution. Traditional methods relying on physical copies have been progressively replaced by digital platforms, which offer advantages such as lower costs, faster content updates and easier accessibility (Huang,

Xie and Xu, 2024). Major digital distribution platforms include Steam and Epic Games Store, which dominate the PC game segment due to their unique operating models.

Steam uses advanced recommendation algorithms that leverage users' preferences, purchase history and gaming behavior. This system facilitates the delivery of highly personalized game recommendations, increasing user retention and satisfaction. In contrast, the Epic Games Store, as a relatively new market entrant, relies on strategic incentives such as exclusive titles and free games to compensate for its less sophisticated personalization mechanisms.

In the console space, platforms such as PlayStation, Xbox and Nintendo have developed distribution channels tailored to the behavior and purchasing habits of console users, who often value high-quality graphics and exclusive game titles. Console platforms integrate interaction mechanisms such as achievement tracking and behavior-based recommendations to optimize user interaction and loyalty.

The mobile gaming sector has also become an important distribution channel, especially via platforms such as the App Store and Google Play. These platforms have advanced ranking and search optimization systems designed to display applications that match users' interests and preferences. The underlying algorithms rely heavily on data such as download frequency, user ratings and engagement metrics, resulting in a highly competitive environment for publishers. The effectiveness of these optimization mechanisms is well documented in the literature, highlighting their role in shaping user acquisition and retention strategies (Lehtonen, Gustafsson and Hassan, 2023).

Traditional Methods for Optimizing Distribution Channels

In the literature, traditional methods for optimizing distribution channels rely predominantly on various statistical analyzes and predictive modeling techniques (Andersen, Goodwin and Granmo, 2024). These methods include analyzing user behavior, download frequency and engagement levels per channel to determine the most effective channels. Traditional approaches often use basic demographic data and historical user behavior to provide information for future strategy planning. For example, studies have shown that the frequency of engagement on certain platforms correlates with users' age groups, their preferences for certain game genres and previous interactions with the brand. Such data provides a basis for decision making regarding investment in specific channels, but lacks the ability to adjust in real time (Souchleris et al., 2023).

However, the limitations of these methods become apparent in dynamic industries such as gaming, where user preferences and trends can change abruptly. Statistical models struggle to account for changes in user behavior over time, resulting in outdated strategies that are not adapted to current market conditions. This challenge is particularly pronounced in the mobile gaming space, where the popularity of games can rise and fall quickly, requiring more flexible optimization methods.

Conventional approaches also have the problem which they are static and therefore cannot react dynamically to new trends or changes in user behavior. Consequently, these methods often lead to inefficiencies as they do not utilize real-time data to adjust strategies, which is becoming increasingly important in the fast-paced gaming industry.

Application of Reinforcement Learning in Distribution Channel Optimization

The use of Reinforcement Learning (RL) in distribution and marketing is becoming increasingly common, not only in the gaming industry, but also in related areas such as e-commerce and digital marketing. RL is a branch of machine learning in which an agent learns through a series of actions evaluated based on feedback (reward or penalty) to achieve an optimal strategy (Amin, 2024). In distribution marketing, RL enables the adaptation of strategies based on actual user behavior, the optimization of resources and the increase of user engagement (Li and Fu, 2021). RL algorithms are particularly suitable for personalizing product recommendations in e-commerce or optimizing content recommendations by tailoring them to users' interests.

In the gaming industry, RL can be used to optimize distribution strategies by adapting to specific channels and user profiles (Gao et al., 2022). The application of RL models, such as Q-Learning and Deep Q-Networks (DQN), allows the model to learn from user behavior feedback and adapt distribution strategies in real time. These algorithms gradually adapt to the results of individual actions, enabling greater precision over time. For example, the model can

learn that users of a certain profile are more likely to download games via mobile platforms, while others prefer consoles, allowing for more precise targeting of preferences (Jayaramireddy et al., 2022).

Reinforcement Learning has the potential to replace traditional optimization methods due to its flexibility, adaptability and real-time responsiveness. RL enables game publishers to segment users and adjust distribution strategies based on real data, resulting in higher engagement and lower distribution costs (Alpaydin, 2014; Safal et al., 2023). Furthermore, as RL learns from feedback, its accuracy in predicting optimal distribution channels improves over time, allowing for additional adjustments without manual intervention.

Methodology

This methodology was developed to apply advanced Reinforcement Learning (RL) techniques to optimize distribution strategies in the gaming industry. The model developed is based on key distribution parameters carefully selected according to industry requirements and available data. The process includes several important steps: data preparation and selection, construction of an RL model tailored to the specific characteristics of digital game distribution, and evaluation of the model using various performance metrics.

Definition of the Reinforcement Learning Model

Reinforcement Learning (RL) is a machine learning technique that enables models to adapt and optimize strategies in real time based on feedback from the environment. RL is particularly suited for complex, dynamic systems such as distribution channels in the gaming industry, where strategies need to be adapted to user behavior and market changes (Gao et al., 2022). RL-based models facilitate the continuous adaptation of distribution strategies with the aim of increasing user engagement and optimizing resources (Raut et al., 2024).

The Deep Q-Network (DQN) algorithm was selected for this study as it is an extension of the classical Q-Learning algorithm. DQN uses neural networks to estimate the Q-function and thus enables learning in complex environments with large state and action spaces. While Q-Learning relies on a simple table to store the rewards for each action in each state, DQN uses neural networks to process large sets of variables (Agostinelli et al., 2018; Vrejoiu, 2019). This makes DQN suitable for this analysis as it can process complex data about user behavior and distribution strategies.

The structure of the RL model includes several key components:

- **State:** represents the current market context for the game and includes various performance metrics. In this model, state includes variables such as the estimated number of owners of the game, the number of peak concurrent users (Peak CCU), the price, positive and negative reviews, the game's rank in the popularity charts (Score Rank), the number of recommendations, the average playtime in the last two weeks, and median playtime forever. These variables help the model to understand the current popularity of the game and user engagement.
- **Actions:** represent the decisions the model can make to optimize the distribution strategy. In the context of game distribution, actions can include changing the price of the game, redistribution of resources to more popular channels or adjusting recommendations to user preferences. Each action can have a positive or negative impact on user engagement and distribution efficiency.
- **Reward:** provides feedback on the effectiveness of a particular action. In this model, rewards can reflect increased engagement, number of downloads, retention rates or reduced distribution costs. Positive rewards encourage the model to maintain successful strategies, while negative rewards help to identify less effective approaches. Rewards are calculated based on a combination of factors, including user ratings, referrals and download numbers.
- **Objective:** the goal of the RL model is to find the optimal distribution strategy that maximizes user engagement and satisfaction while minimizing costs. The model runs through a series of episodes in a simulated environment, testing different strategies until the one that delivers the best long-term results is found. The model adapts during each cycle based on the rewards received for the actions, learning which strategies offer the highest value.

This approach enables the model to continuously learn from real data about distribution channels and user behavior, enabling more precise optimization and adjustment in real time.

Data Preparation and Selection

The data used in this study comes from Kaggle, a popular platform for data science research that offers datasets from various sectors, including the gaming industry. Kaggle collects data from major platforms such as Steam, Epic Games Store, Google Play and App Store, which are important channels for the distribution of digital games. These platforms enable the analysis of distribution strategies in the context of prominent channels and provide deeper insights into user behavior.

The dataset used in this study includes 97,410 games from Kaggle and contains key information such as the estimated number of owners (Estimated owners), peak concurrent users (Peak CCU), the price of the game (Price), positive and negative reviews (Positive, Negative), score rank (Score rank), number of recommendations (Recommendations) and data on average and median playtime (Average playtime two weeks and Median playtime forever).

The following variables were selected for this study due to their direct relevance to the objectives of optimizing distribution strategies:

- **Estimated owners:** the estimated total number of game owners, an important indicator of the popularity of the game
- **Peak CCU (Concurrent Users):** the maximum number of concurrent users during a given time period, which gives an indication of the intensity of engagement
- **Price:** the current price of the game, important for analyzing the impact of pricing on user interest and decisions
- **Positive and Negative:** the number of positive and negative reviews, reflecting the level of user satisfaction
- **Score rank:** the rank of the game based on user reviews and ratings, which serves as an indicator of the overall popularity of the game
- **Recommendations:** the total number of game recommendations, reflecting the level of user satisfaction
- **Average playtime two weeks:** the average playing time over the last two weeks, a important indicator of current user engagement
- **Median playtime forever:** the average total playing time, an indicator of long-term user engagement

To ensure better data quality for modeling and to reduce asymmetry in the distribution of characteristics with large value ranges, a log transformation is applied to the variable Estimated owners. This procedure minimizes the effects of extreme values and improves the stability and efficiency of the modeling. The transformed data is then scaled to fit the input requirements of the RL model.

These variables provide important insights into key aspects of distribution, user engagement, and satisfaction, which are essential for optimizing distribution strategies. Other potentially available data, such as user demographics or game genres, were excluded due to their unclear correlation with the distribution strategies or inconsistencies between platforms.

Construction of the Reinforcement Learning Model

The Deep Q-Network (DQN) algorithm is used for the optimization of distribution strategies as it allows the model to learn from feedback in real time in a complex environment (Embrechts, 2015).
Definition of State S_t: states represent all relevant information about distribution channels and user behavior. The state vector S_t is defined to include the following variables:

$$S_t = \begin{bmatrix} Estimatrd\ owners, Peak\ CCU, Proce, Positive, Negative, Score\ rank, \\ Recommendations, Average\ time\ two\ weeks, Medium\ playtime\ forever \end{bmatrix} \quad (1)$$

Where:

- *estimated owners* – the estimated number of game owners, an indicator of popularity
- *peak CCU* – the maximum number of concurrent users, a measure of engagement
- *price* – the price of the game, important for analyzing the impact of pricing
- *positive and negative* – the number of positive and negative reviews, indicators of user satisfaction
- *score rank* – the rank of the game based on reviews, an indicator of overall popularity
- *recommendations* – the number of game recommendations, a measure of engagement
- *average playtime two weeks* – the average playtime in the last two weeks, an indicator of current engagement
- *median playtime forever* – the median total playtime, representing long-term engagement

Before the features are used as inputs for the RL model, a log transformation is applied to the feature 'Estimated owners' to reduce the impact of extreme values and improve model stability. The log transformation is defined by the following formula:

$$x_{log} = log(1 + x) \tag{2}$$

where *x* represents the original value of the feature. The function *log(1+x)* is used to ensure stability for zero values. This transformation enables a more uniform distribution of the values for the feature 'Estimated owners', which contributes to more effective learning by the RL model.

Definition of Actions A_t: actions represent potential changes in the distribution strategy, including:

- Increase price: raising the price for high-demand games
- Decrease price: lowering the price to encourage downloads
- Redistribute resources: shifting resources toward more popular channels
- Adjust recommendations: optimizing the recommendation algorithm

Reward Function *(R(s,a))*: The reward function evaluates each action based on its impact on user engagement and satisfaction. The mathematical expression for the reward function is defined as:

$$R(s,a) = \alpha \cdot \log(1 + \Delta Positive) + \beta \cdot \frac{\Delta Estimated\ owners}{1+Price} - \gamma \cdot \log(1 + \Delta Negative) \tag{3}$$

Where:

- *α, β, γ* – constants that determine the weight of each component
- *ΔPositive i ΔNegative* – changes in the number of positive and negative reviews
- *ΔEstimated owners* – change in the number of game owners
- *price* – current game price

Q-Function: the model optimizes the value function *Q(s,a)*, which represents the expected cumulative reward for state *s* and action *a*:

$$Q(s,a) = R(s,a) + \gamma \max_{a'} Q(s',a') \tag{4}$$

Where:

- *Q(s,a)* – the value of the current state *s* and action *a*
- *R(s,a)* – the reward for the current action *a* in state *s*
- *γ* – the discount factor that prioritizes long-term rewards
- a'– the optimal action in the new state s'

Mathematical Background of the Model

The model employs several advanced techniques to optimize and ensure the stability of results.

Regularized Loss Function: the regularized loss function is used to reduce overfitting in the model:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(R(s,a) - Q(s,a;\theta)\right)^2 + \lambda\sum_j|\theta_j| \quad (5)$$

Where:

- θ – weights of the neural network
- N – number of examples in the training set
- λ – regularization parameter that reduces overfitting

ϵ-Greedy Approach: The ϵ-greedy approach is used for exploration and exploitation of new strategies, where the agent explores new actions with a probability of ϵ.

Adam Optimizer: the Adam Optimizer is used to adjust weights in the DQN model, facilitating faster convergence and reducing sensitivity to initial conditions. This optimizer automatically adapts the learning rate for each weight in the network, ensuring efficient and stable optimization. Its operation is based on a combination of the first and second moments of gradients, and the mathematical representation of the Adam algorithm is given by the following expressions (Yang, 2024):

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla L(\theta_{t-1}) \quad (6)$$
$$v_t = \beta_2 m v_{t-1} + (1-\beta_2)\left(\nabla L(\theta_{t-1})\right)^2$$
$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\theta_t = \theta_{t-1} - \alpha\frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where:

- m_t – exponential moving average of gradients (first moment)
- v_t – exponential moving average of squared gradients (second moment)
- $\beta 1$, β_2 – hyperparameters controlling the exponential decay rates for the first and second moments
- $\widehat{m}_t, \hat{v}_t$ – bias-corrected estimates of the moments
- $\nabla L(\theta_{t-1})$ – gradient of the loss function L with respect to the weights θ
- α – learning rate
- ϵ – small constant for numerical stability (commonly $\epsilon=10^{-8}$)

Evaluation of Model Performance

Several key metrics are used to evaluate the performance of the RL model, enabling an assessment of accuracy, stability, and long-term effectiveness (Chicco, Warrens, and Jurman, 2021).

Root Mean Squared Error (RMSE): RMSE measures the difference between predicted and actual reward values and is used as an indicator of the model's prediction accuracy:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(Q_{pred} - Q_{true}\right)^2} \quad (7)$$

Where:

- *RMSE* – root Mean Squared Error
- N – number of samples in the test set
- Q_{pred} – predicted reward value
- Q_{true} – actual reward value

Mean Absolute Percentage Error (MAPE): MAPE measures the average absolute error in percentages between the predicted and actual values, enabling the assessment of the model's relative accuracy.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{Q_{true}-Q_{pred}}{Q_{true}}\right| \cdot 100 \quad (8)$$

Where:

- *MAPE* – Mean Absolute Percentage Error
- Q_{true} – actual reward
- Q_{pred} – predicted reward

Cumulative reward: the cumulative reward measures the total long-term performance of the model across all iterations and actions, providing insight into the model's effectiveness in a real-world environment.

$$CR = \sum_{t=1}^{T} R(s_t, a_t) \quad (9)$$

Where:

- *CR* – cumulative reward or the total cumulative reward over time
- *T* – total number of time steps
- $R(s_t,a_t)$ – reward for state s_t and action a_t at time *t*

This metric provides a quantitative measure of how closely the model's predictions align with actual outcomes.
Exploration vs. Exploitation Balance (ϵ-greedy): the model employs the ϵ-greedy method to balance the exploration of new strategies (exploration) and the exploitation of already learned strategies (exploitation). The value of ϵϵ is adjusted over time to gradually enhance the stability of the strategy.

$$a_t = \begin{cases} random\ action & if\ random\ < \epsilon \\ \arg max_a\ Q(s,a) & otherwise \end{cases} \quad (10)$$

Where:

- a_t – the action taken at time *t*
- ϵ – the probability of selecting a random action for exploration
- *Q(s,a)* – the value of the *Q-function* for state *s* and action *a*

The combination of these evaluation metrics enables a thorough assessment of the model's performance, a comparison with other approaches and further adjustments to adapt the model to the requirements of distribution strategies.

This approach ensures a robust and adaptable model that optimizes distribution strategies in the gaming industry and enables real-time strategy adjustments based on user engagement, feedback and changing market conditions.

Implementation and Evaluation of the Model

This chapter contains a detailed description of the implementation and evaluation of the developed DQN model (Deep Q-Network) for the optimization of distribution strategies in the gaming industry. The implementation process includes data preparation, model construction, training and evaluation using various key metrics. The Python programming language is used along with a variety of its popular libraries that facilitate data processing, model development and result analysis.

Python Libraries Used

The development of this model requires the use of several Python libraries, each with a specific role:

- **NumPy**: enables working with multidimensional arrays and performing mathematical operations. It is used for basic mathematical calculations and array manipulations that are essential for data preparation and the implementation of various steps within the DQN model.

- **Pandas**: facilitates the efficient data manipulation required to prepare and process large data sets. In this implementation, Pandas is used for loading, filtering, aggregating and customizing data required for model training. It provides easy access to the data and allows clear organization in a DataFrame format.
- **TensorFlow and Keras**: TensorFlow is a comprehensive platform for machine learning, while Keras provides a simple interface for developing neural networks. By using TensorFlow as a backend and Keras modules, neural network modeling becomes easier and more efficient. In the DQN model, Keras is used to define the model structure, layers and activation functions, while TensorFlow facilitates optimization and rapid model development.

- **Scikit-learn**: a library that provides tools for calculating key assessment metrics such as RMSE and MAPE. Their seamless integration simplifies the evaluation of the accuracy and performance of model predictions.
- **Matplotlib:** used to visualize model results. Matplotlib offers basic functions for graphical representation and customization options for visualizations. Through graphical representations of cumulative rewards, RMSE and MAPE, this library provides insight into model performance and allows tracking of results over time.

Data Preparation

Data preparation involves several key steps, starting with loading and cleaning the data, followed by normalization. The data, which comes from Kaggle, contains information about user distribution and engagement on platforms such as Steam and Epic Games Store. In this phase, the data is normalized to ensure consistency and model stability.

- **Selection of Relevant Variables:** key variables are selected, including the estimated number of owners, peak concurrent users, game price, the number of positive and negative reviews, game rank, the number of recommendations and data on the average playtime. The selection of variables is based on their importance for the optimization of distribution strategies.
- **Data Normalization:** to ensure the consistency of the variable values and the stability of the model, the MinMaxScaler is used to normalize the data into the range [0,1]. Furthermore, an additional data processing step is performed in which the range of the Estimated owners variable is adjusted to its mean value. This transformation allows the neural network to process all variables uniformly, which improves accuracy and convergence speed during training.

The prepared and normalized data is then split into training and testing sets to ensure reliable evaluation of the model (Figure 1).

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

data = pd.read_csv('games.csv')

scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data[[
    'Estimated owners', 'Peak CCU', 'Price', 'Positive', 'Negative',
    'Score rank', 'Recommendations', 'Average playtime two weeks',
    'Median playtime forever'
]])
```

Fig. 1. Python Code for Normalizing and Scaling Dana Using MinMaxScaler

Model Definition

The model is defined with the build_dqn_model function, which uses the Keras API in TensorFlow to build the DQN model. The structure of the model includes the following components:

- **Input Layer**: the input layer receives a vector with all relevant variables (states) representing the current distribution strategy. The input values are normalized so that the neural network can learn efficiently.
- **Hidden Layers**: the model contains two hidden layers, each consisting of 64 neurons and using the ReLU activation function. These hidden layers allow the model to recognize complex patterns in the data and adapt

based on these patterns. The ReLU function mitigates the vanishing gradient problem and enables faster training and better generalization.

- **Output Layer**: the output layer generates values for each possible action and estimates the expected reward for each action. These estimates are used to select the optimal action at a given time.

The defined model has a structure that is complex enough to optimize the distribution strategies and at the same time efficient enough to be executed in a reasonable time frame (Figure 2).

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential([
    Dense(8, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(4, activation='relu'),
    Dense(1, activation='linear')
])

model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mean_squared_error'])
```

Fig. 2. Python Code for Defining a Neural Network Using TensorFlow i Keras

This code trains a DQN model on the provided dataset, using epsilon-greedy policy and calculating RMSE and MAPE for evaluation. Adjust episodes, gamma, epsilon, and epsilon_decay as necessary for optimal performance.

Model Training

The DQN model is trained using the train_dqn_model function, which implements the ϵ-greedy strategy for action exploration and exploitation. The key steps in the training process include:

- **ϵ-Greedy Strategy**: this strategy ensures that the model occasionally explores new actions (exploration) and concentrates on the learned optimal actions (exploitation). In the ϵ-greedy strategy, the exploration factor is gradually reduced over time so that the model can increasingly focus on the best actions.
- **Memory Buffer and 'Experience Replay'**: the memory buffer stores past actions, states and rewards. The 'experience replay' function randomly records these data points during training to break the correlation between successive samples and improve the stability of the model.
- **Model Weight Updates:** at each training step, the model uses stored experience to update its weights via backpropagation. This allows the model to adapt based on feedback from different actions and refine its ability to optimize strategies.

This training method allows the model to learn optimal strategies over a series of episodes, gradually improving the long-term effectiveness of the model (Figure 3).

```python
import numpy as np

def epsilon_greedy_action(q_values, epsilon):
    if np.random.rand() <= epsilon:
        return np.random.randint(action_size)
    return np.argmax(q_values)

def train_dqn_model(data, episodes, gamma, epsilon, epsilon_decay):
    cumulative_rewards = []
    for episode in range(episodes):
        total_reward = 0
        state = np.reshape(data[np.random.choice(data.shape[0])], [1, state_size])
        done = False
        while not done:
            q_values = dqn.model.predict(state)
            action = epsilon_greedy_action(q_values, epsilon)
            next_state = np.reshape(data[np.random.choice(data.shape[0])], [1, state_size])
            reward = calculate_reward(next_state)
            target = q_values.copy()
            target[0][action] = reward + gamma * np.max(dqn.model.predict(next_state)[0])
            dqn.model.train_on_batch(state, target)
            total_reward += reward
            state = next_state
            if np.random.rand() < 0.01:
                done = True
        epsilon *= epsilon_decay
        cumulative_rewards.append(total_reward)
    return cumulative_rewards
```

Fig. 3. Python Code for Training a DQN Model with ϵ-Greedy Strategy Using NumPy, TensorFlow and Keras

Model Evaluation

The evaluation of the model is based on key metrics that assess the performance of the DQN model under real-world conditions:

- **RMSE (Root Mean Squared Error):** measures the average error between the predicted and actual rewards and provides information about the accuracy of the model's predictions. A lower RMSE indicates better accuracy in predicting distribution strategies.
- **MAPE (Mean Absolute Percentage Error):** expresses the average error as a percentage, allowing an assessment of the accuracy of the model in relation to the actual values and a better understanding of the relative precision of the model.
- **Cumulative Reward:** tracks the long-term performance of the model over time. An increase in cumulative reward across episodes indicates successful adaptation of the model to optimal distribution strategies.

The evaluation results confirm the ability of the DQN model to accurately predict key distribution strategies and fit specific patterns in the data. The RMSE shows a high degree of accuracy in the predictions, while the MAPE shows areas for improvement in the relative estimates. The cumulative reward reflects the model's ability to learn and optimize long-term strategies over time, confirming its effectiveness in simulations of real-world conditions (Figure 4).

```python
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
import numpy as np

def calculate_reward(state):
    positive_reviews = state[0][3]
    estimated_owners = state[0][0]
    price = state[0][2]
    negative_reviews = state[0][4]
    reward = np.log(1 + positive_reviews) + estimated_owners / (1 + price) - np.log(1 + negative_reviews)
    return reward

episodes = 100
gamma = 0.95
epsilon = 1.0
epsilon_decay = 0.995
cumulative_rewards = train_dqn_model(scaled_data, episodes, gamma, epsilon, epsilon_decay)

predicted_values = dqn.model.predict(scaled_data)
rmse = np.sqrt(mean_squared_error(scaled_data[:, 0], predicted_values[:, 0]))
mape = mean_absolute_percentage_error(scaled_data[:, 0], predicted_values[:, 0])

print("RMSE:", rmse)
print("MAPE:", mape)
```

Fig. 4. Python Code for Model Evaluation and Calculation of RMSE and MAPE Using NumPy and Scikit-learn

Visualization of Model Performance

Adding the visualization of evaluation metrics with Matplotlib provides a graphical representation of the model's performance over time, offering insights into its evolution and identifying patterns that indicate potential improvements. In this step, graphs of cumulative reward and failure metrics (RMSE and MAPE) across episodes are used. These visualizations help track the progress of the model and provide a clear overview of how the model improves during training (Figure 5).

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

file_path = r'C:\\Users\\User\\Desktop\\IBIMA 2024\\Python\\Games_statistics_Scaled.csv'
data = pd.read_csv(file_path)

data = data.sample(frac=0.01, random_state=42)

y = data['Recommendations']
plt.figure(figsize=(10, 6))
plt.hist(y, bins=50, color='blue', alpha=0.7)
plt.title('Distribution of Recommendations')
plt.xlabel('Number of Recommendations')
plt.ylabel('Frequency')
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

if 'Price' in data.columns:
    X = data.drop(columns=['Recommendations'])
    y_log = np.log1p(y)
    X_log = np.log1p(X)

    plt.figure(figsize=(10, 6))
    plt.scatter(X_log['Price'], y_log, color='red', alpha=0.6, edgecolors='black', s=20)
    plt.title('Price vs Recommendations (Log-Scaled Data)')
    plt.xlabel('Log-Scaled Price')
    plt.ylabel('Log-Scaled Recommendations')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()
else:
    print("Column 'Price' not found in the dataset. Scatter plot cannot be generated.")
```

Fig. 5. Python Code for Data Visualization Using Matplotlib

To call this function, pass the lists or arrays of cumulative_rewards, rmse_values and mape_values together with the corresponding episode list for the *x*-axis. This will create three line graphs, each showing the trend of the respective metric across episodes.

Results and Discussion

The modeling results provide key insights into the effectiveness of the applied DQN model for the optimization of distribution strategies for digital games, which are the focus of this study. The model was evaluated using two standard metrics — RMSE (Root Mean Squared Error) and MAPE (Mean Absolute Percentage Error). These results provide significant insights, but also raise questions related to the specific characteristics of the data used.

Firstly, the achieved RMSE value of 4.31e-3 indicates an exceptionally high model accuracy. The RMSE measures the root mean square error between the actual and predicted values of the target variable, and the result within the scaled data range indicates that the model consistently produces predictions that are very close to the actual values. In the context of this study, the low RMSE value indicates that the model effectively maps the relationships between key variables — such as price, positive and negative reviews, and the number of recommendations — to the target distribution strategies. This model accuracy supports the hypothesis that deep learning, in particular the DQN model, can be successfully used to predict distribution strategies in the gaming industry.

A visual representation of these relationships can be seen in Figure 6, where a scatter plot represents the log-transformed data and shows patterns in the interaction of the key variables. The figure clearly shows how the model uses the relationships between price and recommendations to learn distribution strategies. This visualization provides additional evidence of the high RMSE accuracy as it shows consistent patterns in the log-transformed data.

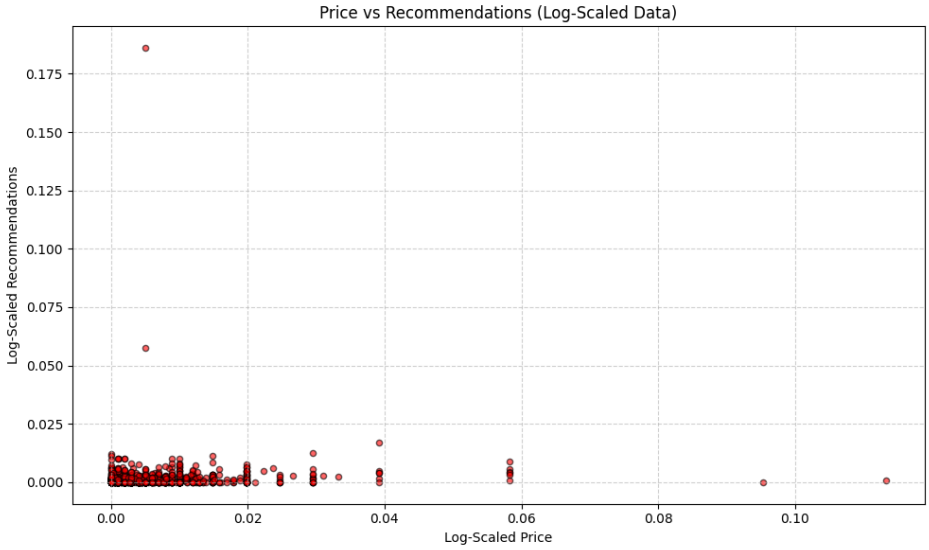

Fig. 6. Scatter Plot of the Relationship Between Price and Recommendations After Log-Transformation

On the other hand, the high MAPE value of 1.10e14% reveals significant limitations of the model. MAPE expresses the average relative error between actual and predicted values as a percentage. The extreme value of this key figure indicates problems in connection with the data distribution. In particular, the analyzed data show a significant number of zeros in the price column (more than 20% of the data) and a potentially skewed distribution of the target variable recommendations, which can distort the MAPE calculation.

MAPE is particularly sensitive to cases where the actual values are close to zero, as even small absolute errors in such situations lead to very high relative percentages. This is illustrated in Figure 7, which clearly shows the distribution of recommendations and highlights the predominant presence of zero values

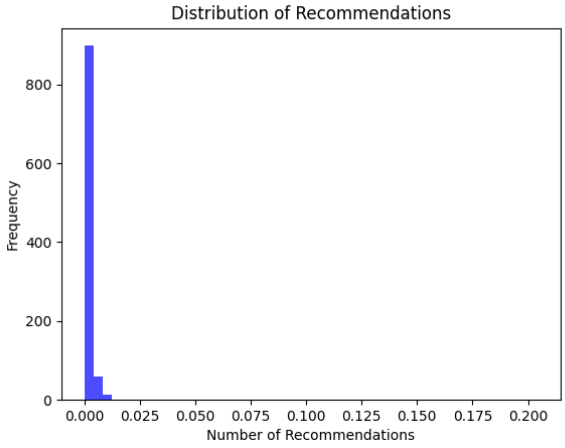

Fig. 7. Histogram of the Distribution of Recommendations in the Dataset

It is important to emphasize that the high MAPE value does not negate the success of the model, but rather provides additional context for the evaluation. For distribution strategies, the absolute accuracy of predictions (expressed by the RMSE) is often more important than percentage errors. The RMSE in this study clearly demonstrates that the model has the ability to generalize and predict key distribution variables within an acceptable error range.

In summary, the modeling results highlight several key implications for this study. Firstly, the exceptionally low RMSE confirms that the proposed DQN model can successfully predict distribution strategies and provide valuable recommendations for optimization in the gaming industry. These results are further strengthened by the fact that the model is capable of analyzing large data sets, making it scalable and applicable to various digital distribution platforms. Secondly, the high MAPE value raises an important discussion about the impact of data quality and structure on model performance. This limitation could potentially be addressed by additional data processing steps, such as reducing the influence of zeros or outliers, which could improve the model's performance in terms of percentage accuracy.

Thus, this study not only contributes to the understanding of the applicability of the DQN model in predicting distribution strategies, but also opens avenues for future research focused on improving data quality and further optimizing the model. The evaluation using RMSE and MAPE provides a solid foundation for discussing the practical implications and limitations of applying artificial intelligence in user engagement analysis and distribution optimization in the digital games industry. These results should serve as a basis for the implementation of the DQN model in real business scenarios, with a particular focus on further adapting the model to the specific needs of distribution platforms and their users.

Conclusion

In conclusion, this study, based on the application of deep learning models to optimize distribution strategies in the games industry, confirms the relevance and effectiveness of Reinforcement Learning (RL) in the distribution of video games. The developed DQN model demonstrated the ability to analyze complex patterns of user behavior and adapt distribution strategies in real time, which is crucial for the dynamic sector of the gaming industry.

The evaluation results show an exceptionally low RMSE value (4.31e-3), which confirms the accuracy of the model in determining optimal distribution strategies. At the same time, the high MAPE value indicates certain limitations in the structure and distribution of the data used, especially for variables with a high proportion of zero values. Nonetheless, the cumulative reward across episodes underscores the model's capacity for long-term learning and adaptation.

By applying RL to distribution optimization, game publishers can identify and target key user segments, increase engagement and reduce distribution costs, which is a significant advantage in a highly competitive environment. The results show not only that the model is scalable, but also that it can be adapted to different distribution platforms, making it widely applicable.

Although the results are promising, the study highlights the need for further data processing, such as mitigating the influence of outliers and reducing the impact of zeros, to further improve the accuracy of the model. Future research could incorporate more complex models or hybrid approaches that combine RL with other machine learning methods to further optimize the efficiency of distribution strategies.

References

- Agostinelli, F. et al. (2018), 'From Reinforcement Learning to Deep Reinforcement Learning: An Overview', *Braverman Readings in Machine Learning: Key Ideas from Inception to Current State*, 11100(218), 298-328.
- Ali, M. A. F. and Baizal, Z. K. A. (2023), 'Video Game Recommender System Using Deep Reinforcement Learning', Proceedings of the International Conference on Advancement in Data Science, E-Learning and Information Systems: Data, Intelligent Systems, and the Applications for Human Life, ICADEIS 2023, ISBN: 979-8-3503-0342-1, 2-3 August 2023, Bali, Indonesia, 152-157.
- Alharin, A., Doan, T. N. and Sartipi, M. (2020), 'Reinforcement Learning Interpretation Methods: A Survey', *IEEE Access*, 8(2020), 171058-171077.
- Alpaydin, E. (2014), Reinforcement Learning, Introduction to Machine Learning, Alpaydin, E. (ed.), MIT Press, Cambridge, Massachusetts, USA.

- Amin, S. (2024), An introduction to reinforcement learning and its application in various domains, Deep Learning, Reinforcement Learning, and the Rise of Intelligent Systems, Uddin, M. and Mashwanu, W. (eds), IGI Global Scientific Publishing, Hershey, Pennsylvania, USA.
- Andersen, P. A., Goodwin, M. and Granmo, O. C. (2024), 'Towards safe and sustainable reinforcement learning for real-time strategy games', *Information Sciences*, 679, 120980.
- Chicco, D., Warrens, M. J. and Jurman, G. (2019), 'The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation', *PeerJ Computer Science*, e623.
- Embrechts, M. J. (2015), Reinforcement Learning, Design of Experiments for Reinforcement Learning, Gatti, C. (ed), Springer-Verlag, Berlin, Germany.
- Gao et al. (2021), 'Cost-Efficient and Quality-of-Experience-Aware Player Request Scheduling and Rendering Server Allocation for Edge-Computing-Assisted Multiplayer Cloud Gaming', IEEE Internet of Things Journal, 9(14), 12029-12040.
- Hu et al. (2023), 'Promoting human-AI interaction makes a better adoption of deep reinforcement learning: a real-world application in game industry', Multimedia Tools and Applications, 83(2023), 6161-6182.
- Huang J. L, Xie, D. X. and Xu, Z. (2024), 'Sequential innovation and contribution distribution: measurement from game live-streaming industry', Humanities and Social Sciences Communications, 11(1), 646.
- Jayaramireddy et al. (2022), 'A Survey of Reinforcement Learning Toolkits for Gaming: Applications, Challenges and Trends', Proceedings of the 7th Future Technologies Conference, FTC 2022, Vol. 1, ISBN: 978-3-031-18460-4, 20-21 October 2022, Vancouver, Canada, 165-184.
- Lehtonen, M. J., Gustafsson, R. and Hassan, L. (2023), 'The multiplex of value creation and capture logics in the video game industry: An integrative review of 20 years of studies and a future research agenda', Technological Forecasting and Social Change, 195(2023), 122756.
- Li, K. and Fu, M. (2021), 'Effectiveness of Machine Learning Methods on Gaming Recommendation and Prediction', Proceedings of the International Conference on Computing and Data Science, CONF - CDS 2021, ISBN: 978-1-4503-8957-0, 28-30 January 2021, Stanford, California, USA, 1-5.
- Raut et al. (2024), 'Unity ML-Agents: Revolutionizing Gaming through Reinforcement Learning', Proceedings of the 2nd World Conference on Communication and Computing, WCONF 2024, ISBN: 979-83-503-9533-4, 12-14 July 2024, Raipur, India, 1-7.
- Safal, A. et al. (2023), 'Artificial Intelligence Involvement in Graphic Game Development', Proceedings of the 2nd International Conference on Augmented Intelligence and Sustainable Systems, ICAISS 2023, ISBN: 9798350325805, 23-25 August 2023, Trichy, India, 82-86.
- Schindler, S., Uray, M., Huber, S. (2023), 'A Mini Review on the utilization of Reinforcement Learning with OPC UA', Proceedings of the 21st International Conference on Industrial Informatics, INDIN 2023, ISBN: 9781665493147, 18-20 July 2023, Lemgo, Germany, 298-303.
- Souchleris, K., Sidiropoulos, G. K., Papakostas, G. A. (2023), 'Reinforcement Learning in Game Industry-Review, Prospects and Challenges', *Applied Sciences Basel*, 13(4), 2443, 1-23.
- Tian, G., Pan, L. and Liu, S. (2024), 'An online cost optimization approach for edge resource provisioning in cloud gaming', *Journal of Network and Computer Applications*, 232(2024), 104008.
- Vrejoiu, M. H. (2019), 'Deep Reinforcement Learning. Case Study: Deep Q-Network', *Romanian Journal of Information Technology and Automatic Control*, 29(3), 65-78.
- Yang, L. (2024), 'Theoretical Analysis of Adam Optimizer in the Presence of Gradient Skewness', *International Journal od Applied Science*, 7(2), 27-42.
- Yu, F. R. and He, Y. Reinforcement Learning and Deep Reinforcement Learning, Deep Reinforcement Learning for Wireless Networks, Yu, F. R. and He, Y. (eds.), Springer Interantional Publishing, Cham, Switzerland.