International Conference AUTOMATICS AND INFORMATICS'2025, October 09 - 11, 2025, Varna, Bulgaria (ICAI'25)

Integrating Deep Learning and Reinforcement Learning: A Case Study on the Chrome Dino Game

Krish Anish Shah
Department of Business
Univ. of Europe for Applied Science
14469 Potsdam, Germany
krishanish.shah@ue-germany.de

Loubna Ali
Department of Computer Science and Informatics
BSBI
Berlin, Germany
loubna.ali@berlinsbi.com

Shan Faiz
Department of Business
Univ. of Europe for Applied Science
14469 Potsdam, Germany.
shan.faiz@ue-germany.de

Meerah Karunanithi
Department of Business
Univ. of Europe for Applied Science
14469 Potsdam, Germany
meerah.karunanithi@ue-germany.de

Debopriya Das
Department of Business
Univ. of Europe for Applied Science
14469 Potsdam, Germany
debopriya.das@ue-germany.de

***Abstract*—The intersection of deep learning and reinforcement learning has opened new avenues in real-time decision-making for interactive systems like games. In this context, the Chrome Dino game serves as a compelling platform to evaluate AI models' ability to perceive and act dynamically. Despite significant progress in AI gaming applications, there remains a gap in integrated approaches that simultaneously leverage visual recognition and adaptive decision-making in fast-paced environments. This study addresses the challenge by combining deep learning for action detection with reinforcement learning for policy optimization. A convolutional neural network (CNN) was trained to classify player actions—jump, duck, and idle—based on game images. The trained model was then integrated with a Deep Q-Network (DQN) reinforcement learning framework to enable adaptive gameplay based on trial-and-error interaction. The combined model showed strong performance, with the CNN achieving high accuracy and low loss during classification, and the DQN demonstrating significant improvement in gameplay metrics such as episode length and cumulative rewards. Training graphs revealed a consistent reduction in loss and enhanced policy convergence over 1,000 episodes. Additionally, a custom simulation environment was developed to facilitate training and evaluation in real-time. This study demonstrates a successful integration of vision-based deep learning with decision-making reinforcement learning, offering a viable blueprint for AI agents in dynamic and unpredictable settings.**

***Index Terms*—deep learning, reinforcement learning, Chrome Dino, action detection**

I. Introduction

The advancement of artificial intelligence [1], particularly in areas like computer vision [2] and autonomous systems [3], has led to significant innovations. Among these is the application of AI in gaming, where models learn to interact with and navigate complex environments [4]. The Chrome Dino game presents a simple yet challenging scenario for testing the capabilities of deep learning [5] and reinforcement learning [6]. By training a model to perform actions such as ducking, jumping, and remaining idle based on visual cues, we aim to illustrate the effectiveness of these AI techniques in a real-time setting. AI has become a cornerstone of technological advancement [7], revolutionizing diverse industries such as healthcare, finance, manufacturing, and entertainment [8]. Among its subfields, Machine Learning (ML) [9] and Deep Learning (DL) [10] have gained substantial attention for their ability to extract meaningful patterns from large datasets and enable autonomous decision-making. These techniques power everything from personalized recommendations and fraud detection to language translation and medical diagnostics. Reinforcement Learning (RL), a specialized branch of ML, focuses on training agents to make sequences of decisions through interaction with an environment [11], [12]. By learning from rewards and penalties, RL enables agents to adapt and optimize their behavior in dynamic and uncertain settings. As these AI capabilities evolve, they increasingly support the development of intelligent systems capable of real-time learning, planning, and execution [13].

The synergy between deep learning, which excels at feature extraction [14], and reinforcement learning, which optimizes action selection, creates a powerful combination for training AI agents [15]. In gaming, where unpredictability is common, this combination allows for the development of sophisticated models capable of making complex decisions. The Chrome Dino game, with its straightforward mechanics but high-speed action, serves as an ideal testbed for these methodologies [16]. The gaming industry has emerged as a highly effective testbed for evaluating and advancing AI methodologies, particularly reinforcement learning and deep learning [17]. Games provide controlled yet complex environments where agents must perceive, reason, and act in response to rapidly changing conditions—an ideal setting for testing adaptive algorithms. Applications of AI in gaming range from non-player character (NPC) behavior modeling to real-time strategy optimization and autonomous gameplay. Deep Learning enhances these ap-

979-8-3315-8611-9/25/$31.00 ©2025 IEEE

plications by enabling agents to interpret raw visual data, while RL empowers them to make contextually appropriate decisions based on experience. This synergy has been demonstrated in landmark achievements such as AlphaGo and OpenAI Five, showcasing AI's potential to outperform humans in strategic domains. Beyond entertainment, such advancements are transferable to real-world scenarios like robotics, autonomous driving, and industrial automation. As a result, the fusion of AI and gaming not only elevates gameplay experiences but also drives progress in intelligent system development.

A. Deep Learning Session

Deep learning [18] is employed to identify and classify the key actions in the game. The model is trained on a dataset consisting of images captured from various stages of gameplay, with each image labeled according to the action being performed. The primary challenge is to enable the model to recognize these actions accurately and swiftly, even in dynamic environments.

B. Reinforcement Learning Session

Once the deep learning model is trained, reinforcement learning is applied to refine the model's performance. This stage focuses on teaching the model to maximize rewards through trial-and-error interaction with the game environment. By combining deep learning with reinforcement learning, the model not only recognizes actions but also optimizes them to improve gameplay performance over time.

II. Methodology

The methodological framework of this study integrates deep learning–based perception with reinforcement learning–based decision making to enable autonomous gameplay in the Chrome Dino environment. The pipeline consisted of four major components: dataset acquisition, preprocessing, model architecture, and reinforcement learning setup.

A. Dataset and Preprocessing

The dataset is composed of thousands of images from the Chrome Dino game, categorized into three primary actions: duck, idle, and jump. Each image is standardized to 83x100 pixels, a size chosen to balance between image resolution and computational efficiency.

The preprocessing steps involve resizing, normalizing, and augmenting the images to improve the model's robustness. After preprocessing, the images are fed into the deep learning model, which is trained to predict the correct action based on visual input.

The overall workflow followed in the study is shown in Figure 4.

B. Deep Learning Model Architecture

The deep learning model utilizes a convolutional neural network (CNN) to process the input images. The architecture includes several convolutional layers, followed by pooling layers and fully connected layers, which work together to classify the images into one of the three action categories.

TABLE I
Network Configuration for Deep Learning Model

Network Configuration	
Epochs	50
Learning rate	0.0001
Mini batch size	64
Optimizer	Adam
Loss Function	Categorical Cross-Entropy
Training samples	3000
Validation samples	1000

The CNN architecture is chosen for its effectiveness in image classification tasks. Convolutional layers capture spatial features, while pooling layers reduce dimensionality. The fully connected layers then map these features to the final output, predicting the action being performed in each image.

As shown in Figure 3, the training process resulted in high accuracy and low loss, indicating the model's effectiveness in recognizing actions from the game images.

C. Dataset Acquisition

Images were systematically collected from multiple stages of the Chrome Dino game to ensure that the dataset captured the variability of in-game conditions. The collection covered a range of obstacle types and speeds, thereby reflecting different levels of gameplay difficulty. Each image was annotated according to the action required by the Dino character—jump, duck, or idle—resulting in a dataset suitable for supervised training. By incorporating diverse gameplay scenarios, the dataset was designed to enhance the generalizability of the trained model.

D. Preprocessing

To prepare the dataset for model training, several preprocessing steps were applied. All images were converted to grayscale to reduce computational complexity while retaining key visual features. The frames were then resized to 83 × 100 pixels, a resolution chosen to balance visual fidelity with efficiency. Normalization of pixel intensities was performed to standardize input values, thereby stabilizing model training. Data augmentation, including random flips and intensity shifts, was employed to artificially expand the dataset and improve robustness against variations in obstacle placement and lighting.

E. Model Architecture

The core perception module was built around a convolutional neural network (CNN), a well-established architecture for visual classification tasks. The CNN comprised three convolutional layers, each followed by pooling layers to progressively capture spatial hierarchies while reducing dimensionality. These layers extracted features such as obstacle edges, shapes, and positions relative to the Dino character. The final fully connected layers mapped these extracted features to the three possible action classes. The design rationale was to allow convolutional layers to act as automatic feature extractors,

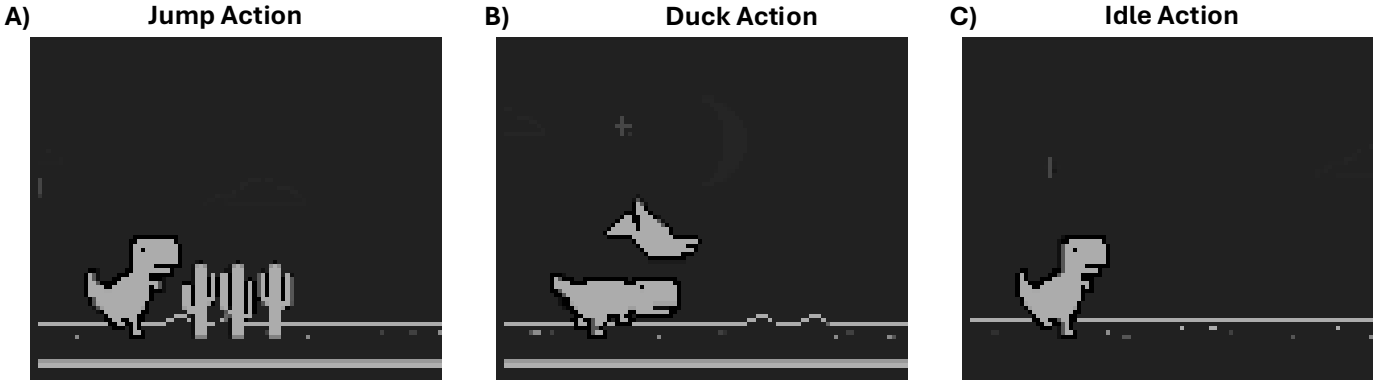

Fig. 1. Examples from the dataset, illustrating different actions (duck, idle, jump) executed by the Dino character in the game.

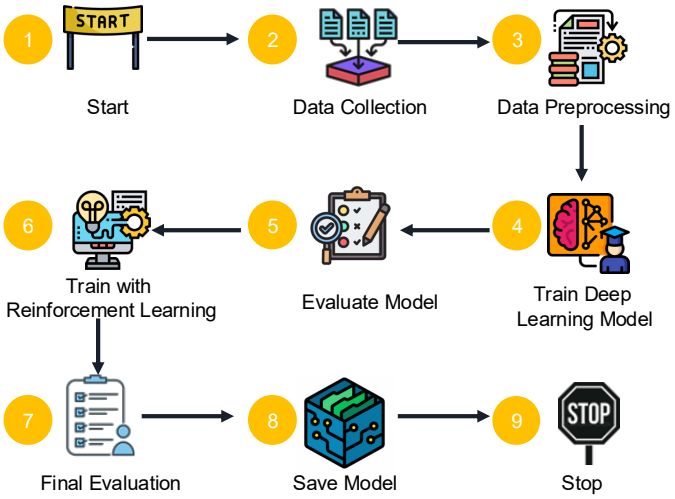

Fig. 2. Workflow Diagram illustrating the deep learning and reinforcement learning integration process.

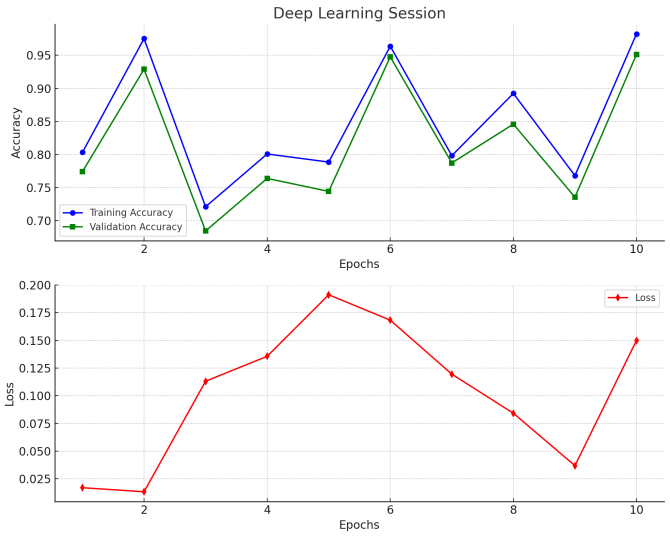

Fig. 3. Graph depicting the training and validation accuracy, as well as the loss over epochs for the deep learning model.

Fig. 4. Results for Reinforcement Learning: The steps from data preprocessing to model training and performance evaluation.

while pooling layers mitigated overfitting by discarding redundant information. The fully connected layers subsequently translated high-level visual features into action predictions.

F. Reinforcement Learning Setup

Once the perception model was trained, its outputs were integrated into a reinforcement learning framework to enable dynamic gameplay. A custom simulation environment was defined, where the Dino agent interacted with the game by executing discrete actions: jump, duck, or idle. The reinforcement learning algorithm received states from the CNN, executed an action, and obtained feedback in the form of rewards. Rewards were granted for successful avoidance of obstacles and penalized for collisions, ensuring that the agent learned to maximize survival time. An adaptive exploration–exploitation strategy was adopted: during the early training stages, higher exploration encouraged the agent to test diverse behaviors, while over time, the policy shifted toward exploiting the most effective strategies identified.

III. Process 2: Reinforcement Learning for Dynamic Adaptation

A. Reinforcement Learning Overview

Reinforcement learning (RL) is used to enhance the model's adaptability by allowing it to learn optimal strategies through interaction with the game environment. The deep Q-network (DQN) algorithm is implemented to enable the model to make decisions that maximize cumulative rewards.

B. Reinforcement Learning Workflow

The reinforcement learning process begins with the deep learning model providing the initial policy. The DQN model then interacts with the game, receiving feedback in the form of rewards, which it uses to refine its policy.

TABLE II
REINFORCEMENT LEARNING EXPERIMENTAL SETUP

Reinforcement Learning Configuration	
Episodes	1000
Discount factor (gamma)	0.99
Learning rate	2.5e-4
Exploration rate	0.1
Replay buffer size	10000
Batch size	64
Target update frequency	1000

C. Experimental Settings

The experimental setup for reinforcement learning is designed to optimize the model's performance in the game environment. The key hyperparameters used in the training process are listed in Table II. These parameters were carefully chosen based on extensive trials to ensure the model's stability and effectiveness.

The model was trained over 1000 episodes, with a discount factor set at 0.99 to prioritize long-term rewards. The learning rate was set to 2.5×10^{-4} to balance between convergence speed and stability. The exploration rate was initially set to 0.1, allowing the model to explore different actions before settling on an optimal strategy.

D. Results and Evaluation

The reinforcement learning phase led to notable improvements in the model's performance. The metrics, such as the mean episode length and cumulative rewards, indicated a strong learning curve. These metrics are visualized in Figure 5, which shows the progression of the model over time.

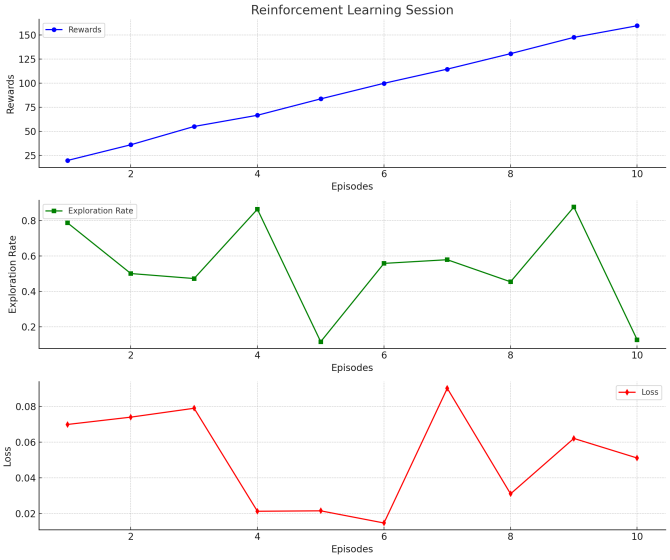

Fig. 5. Graph depicting the cumulative rewards, exploration rate, and loss over episodes for the reinforcement learning model.

The model's ability to adapt to the game environment improved significantly, as evidenced by the increasing episode lengths and rewards [19]. The reduction in loss over time further highlights the model's effectiveness in optimizing its strategy [20].

E. Overall Evaluation

Positive Outcomes:

- **Improvement in Episode Length and Reward:** The model demonstrated a clear improvement in gameplay, with longer episodes and higher rewards as training progressed.
- **Reduction in Loss:** The consistent decrease in loss reflects the model's successful learning and optimization of its actions.

Challenges and Future Directions:

- **Stability of Training:** While the model showed improvement, some instability in training was observed, particularly in the exploration vs. exploitation balance. Future work could focus on fine-tuning the hyperparameters and improving the regularization techniques to stabilize the training process.
- **Real-Time Adaptability:** Enhancing the model's ability to adapt in real-time to more complex and unpredictable game scenarios is an area for further research.

IV. IMPLEMENTATION DETAILS AND CODE INTEGRATION

The foundation of this research is deeply rooted in the implementation and integration of advanced deep learning and reinforcement learning methodologies. These techniques were meticulously selected and tailored to address the challenges presented by the dynamic environment of the Chrome Dino game. To realize the objectives and achieve the results discussed in this paper, a carefully structured Python codebase was developed [21]. This code plays a pivotal role in the entire process, seamlessly combining the tasks of image preprocessing, the creation of a custom environment for simulation, and the rigorous training phase using reinforcement learning algorithms [22]. The following Python code encapsulates the essence of our approach, providing a detailed glimpse into the underlying mechanisms that drive the learning and decision-making processes of the AI model [23].

The implementation of the proposed framework was carefully designed to integrate deep learning–based perception with reinforcement learning–based decision making in the Chrome Dino game [24]. Instead of presenting raw code, this section describes the major components and workflow that together enabled autonomous gameplay [25].

A. Development Environment

The system was developed using Python, leveraging well-established machine learning and reinforcement learning libraries. For deep learning tasks, TensorFlow/Keras was employed due to its robust support for convolutional neural networks and ease of model customization. For reinforcement learning, the Stable Baselines3 library was used, which provides reliable implementations of state-of-the-art algorithms, including the Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN). The Gymnasium toolkit was adopted to

define a custom environment that mimics the dynamics of the Chrome Dino game. Training and evaluation were carried out on a workstation equipped with a GPU, ensuring efficient model convergence.

B. Dataset Handling and Preprocessing

A dataset of gameplay images was organized into three categories: jump, duck, and idle. Each image was converted to grayscale and resized to 83×100 pixels, striking a balance between computational efficiency and visual detail. Normalization was applied to scale pixel values, while data augmentation techniques such as random flips and intensity shifts were introduced to improve generalization. This standardized dataset enabled the convolutional neural network (CNN) to learn robust action recognition across different gameplay scenarios.

C. Custom Simulation Environment

A custom simulation environment was implemented to replicate the interaction loop between the agent and the Dino game. This environment provided the model with states (image frames), accepted discrete actions (jump, duck, idle), and returned appropriate rewards or penalties based on survival. The reward function was crafted to incentivize correct actions when facing obstacles and penalize incorrect decisions, thereby aligning agent learning with successful gameplay. The design of this environment followed the agent–environment–reward cycle of reinforcement learning and was validated to ensure compliance with Gymnasium standards.

D. Reinforcement Learning Framework

The reinforcement learning component was initialized using the trained CNN as a perceptual backbone. The reinforcement agent then interacted with the environment over multiple episodes, refining its decision-making policy. For stability and efficiency, a Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) were tested, with PPO showing superior performance in long-episode survival tasks. Key hyperparameters, such as the learning rate, discount factor, and batch size, were optimized through extensive experimentation (as summarized in Table II). Exploration–exploitation balance was maintained by introducing an adaptive exploration rate, ensuring that the agent initially explored diverse strategies before converging to an optimal policy.

E. Training Procedure and Model Saving

Training proceeded in two major phases. First, the CNN was trained on the action-recognition dataset, producing a reliable feature extractor with high classification accuracy. Second, the reinforcement learning agent employed this perceptual model to interact with the custom Dino environment, gradually improving its gameplay performance through trial and error. Over 1,000 episodes, the agent demonstrated consistent increases in cumulative reward and mean episode length, accompanied by a steady reduction in loss values. The final trained model was serialized and stored, allowing reproducibility and facilitating further experimentation.

F. Summary of Implementation

The modular structure of the implementation—consisting of dataset preprocessing, custom environment creation, deep learning for perception, and reinforcement learning for decision optimization—ensured a systematic integration of both paradigms. This design not only provided strong empirical results in the Chrome Dino testbed but also established a blueprint for extending the approach to more complex and visually dynamic environments.

Description of the Code: The code above illustrates the complete workflow for training a deep learning model to classify actions (duck, jump, idle) in the Chrome Dino game and then applying reinforcement learning (RL) using the Proximal Policy Optimization (PPO) algorithm to optimize gameplay.

1) Dataset Loading and Preprocessing: The script starts by loading images from the dataset directory. These images are preprocessed into arrays and labeled according to the action they represent (jump, duck, idle). The image dimensions are set to 83x100 pixels to maintain a balance between performance and computational load.
2) Custom Environment Setup: A custom environment (DinoGameEnv) is created using the Gymnasium library. This environment is essential for RL, as it allows the model to interact with the game by providing actions and receiving states, rewards, and done signals in return. The environment's observation space is set to handle the preprocessed images, while the action space is discrete, representing the three possible actions.
3) Reinforcement Learning Model Configuration: The code configures the PPO algorithm using Stable Baselines3, a popular library for RL in Python. The model is trained with a Convolutional Neural Network (CNN) policy, and the neural network architecture is customized to include three hidden layers with 256 units each (net_arch=[256, 256, 256]). Key hyperparameters such as learning rate (learning_rate=2.5e-4), discount factor (gamma=0.99), and batch size (batch_size=64) are set to optimize training.
4) Training and Evaluation: The model undergoes training for a total of 50,000 timesteps. Throughout this process, the model continuously interacts with the game environment, updating its policy to maximize cumulative rewards. The trained model is saved for later use or further refinement.
5) Model Saving: After the training process is complete, the model is saved in a file (dino_game_model.zip). This allows for easy retrieval and deployment of the trained model in different environments or for further testing.

V. Conclusion

This study successfully demonstrates the integration of deep learning and reinforcement learning techniques to create an intelligent agent capable of playing the Chrome Dino game autonomously. By training a Convolutional Neural Network

(CNN) to classify in-game actions and combining it with a Deep Q-Network (DQN) for real-time decision-making, the agent was able to adapt and improve its gameplay over time. The results indicate that the CNN achieved high accuracy in action recognition, while the DQN agent consistently improved its performance through interaction with the environment. The custom simulation environment further contributed to efficient training and evaluation. Overall, this project highlights the potential of combining visual perception and reinforcement-based learning for developing adaptive AI agents in interactive settings. The approach can be extended to more complex games and real-world scenarios requiring both perception and decision-making. Future work may involve optimizing model efficiency and applying similar techniques to 3D environments or robotic control tasks.

REFERENCES

[1] S. Abbas, A. Maaz, R. H. Ali, T. A. Khan, and I. Ahmed, "Automatic timetable generation using neural networks trained by genetic algorithms," in *2024 18th International Conference on Open Source Systems and Technologies (ICOSST)*. IEEE, 2024, pp. 1–6.

[2] M. Ahmed, R. H. Ali, and N. Ali, "Traffic sign recognition system," in *2024 International Conference on IT and Industrial Technologies (ICIT)*. IEEE, 2024, pp. 1–6.

[3] R. Ali, "Determining the relationship between gene class network graph and gene age," Ph.D. dissertation, Chalmers University of Technology, 2009.

[4] D. Javed, U. Arshad, S. Peerzada, M. R. Saud, N. Ali, and R. H. Ali, "Vigilantai: Real-time detection of anomalous activity from a video stream using deep learning," in *2024 International Conference on IT and Industrial Technologies (ICIT)*. IEEE, 2024, pp. 1–6.

[5] T. A. Khan, R. H. Ali *et al.*, "Crafting a player impact metric through analysis of football match event data," *Journal of Computational Mathematics and Data Science*, p. 100115, 2025.

[6] R. M. Mian, S. Khan, and R. H. Ali, "Bank note authentication using deep learning," in *2024 International Conference on IT and Industrial Technologies (ICIT)*. IEEE, 2024, pp. 1–6.

[7] A. Ramzan, R. H. Ali, N. Ali, and A. Khan, "Enhancing fake news detection using bert: A comparative analysis of logistic regression, rfc, lstm and bert," in *2024 International Conference on IT and Industrial Technologies (ICIT)*. IEEE, 2024, pp. 1–6.

[8] M. H. Shah, M. A. Bakar, R. H. Ali, Z. U. Abideen, U. Arshad, A. Z. Ijaz, N. Ali, M. Imad, and S. Nabi, "Investigating novel machine learning based intrusion detection models for nsl-kdd data sets," in *2023 International Conference on IT and Industrial Technologies (ICIT)*. IEEE, 2023, pp. 1–6.

[9] I. Ul Hassan, R. H. Ali, Z. u. Abideen, A. Z. Ijaz, and T. A. Khan, "Towards effective emotion detection: A comprehensive machine learning approach on eeg signals," *BioMedInformatics*, vol. 3, no. 4, pp. 1083–1100, 2023.

[10] C. Shackleton, R. H. Ali, and T. A. Khan, "Enhancing rangeland weed detection through convolutional neural networks and transfer learning," *Crop Design*, vol. 3, no. 3, p. 100060, 2024.

[11] H. Anjum, U. Arshad, R. H. Ali, Z. U. Abideen, M. H. Shah, T. A. Khan, A. Z. Ijaz, A. B. Siddique, and M. Imad, "Robust and reliable liveness detection models for facial recognition systems," in *2023 International Conference on Frontiers of Information Technology (FIT)*. IEEE, 2023, pp. 292–297.

[12] W. Babar, R. H. Ali, A. Faheem, and S. A. Mansoor, "Using convolutional neural networks for enhanced pneumonia detection via chest x-rays," in *2024 International Conference on IT and Industrial Technologies (ICIT)*. IEEE, 2024, pp. 1–6.

[13] A. W. Paracha, U. Arshad, R. H. Ali, Z. U. Abideen, M. H. Shah, T. A. Khan, A. Z. Ijaz, N. Ali, and A. B. Siddique, "Leveraging ai and nlp in chatbot development: An experimental study," in *2023 International Conference on Frontiers of Information Technology (FIT)*. IEEE, 2023, pp. 172–177.

[14] A. Maaz, S. Abbas, R. H. Ali, I. Ahmed, and T. A. Khan, "Vgg models in image captioning: Which architecture delivers better descriptions?" in *2024 18th International Conference on Open Source Systems and Technologies (ICOSST)*. IEEE, 2024, pp. 1–6.

[15] A. Khan, R. H. Ali, U. Akmal, and A. Ramazan, "Asl recognition using deep learning algorithms," in *2024 International Conference on IT and Industrial Technologies (ICIT)*. IEEE, 2024, pp. 1–6.

[16] P. Adomako, T. A. Khan, R. H. Ali, and R. Koutaly, "Comparison of leading ai models an analytical study of chatgpt google bard and microsoft bing," *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 14, pp. e31 857–e31 857, 2025.

[17] N. Ali, K. Ali, F. Ali, A. Ali, N. Ali, and R. H. Ali, "Streamlining attendance with voice recognition via gaussian mixture model," in *2024 International Conference on IT and Industrial Technologies (ICIT)*. IEEE, 2024, pp. 1–6.

[18] W. Alam, R. H. Ald, N. Ali, M. Imad, Z. U. Abideen, and M. H. Shah, "Machine learning based fraudulent detection system for financial transactions," in *2024 International Conference on IT and Industrial Technologies (ICIT)*. IEEE, 2024, pp. 1–6.

[19] A. B. Siddique, H. B. Khalid, and R. H. Ali, "Diving into brain complexity: Exploring functional and effective connectivity networks," in *2023 18th International Conference on Emerging Technologies (ICET)*. IEEE, 2023, pp. 321–325.

[20] D. Mahmood, U. Arshad, R. H. Ali, Z. U. Abideen, M. H. Shah, T. A. Khan, A. Z. Ijaz, N. Ali, and A. B. Siddique, "Exploiting partial observability and optimized simple state representations in deep q-learning," in *2023 International Conference on Frontiers of Information Technology (FIT)*. IEEE, 2023, pp. 25–30.

[21] M. Karunanithi, P. Chatasawapreeda, and T. A. Khan, "A predictive analytics approach for forecasting bike rental demand," *Decision Analytics Journal*, vol. 11, p. 100482, 2024.

[22] M. Karunanithi, H. Mouchrik, A. A. Rizvi, T. A. Khan, R. Kouatly, and I. Ahmed, "An improved particle swarm optimization algorithm," in *2023 IEEE 64th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*. IEEE, 2023, pp. 1–6.

[23] S. Bhardwaj, S. M. Salim, T. A. Khan, and S. JavadiMasoudian, "Automated music generation using deep learning," in *2022 International Conference Automatics and Informatics (ICAI)*. IEEE, 2022, pp. 193–198.

[24] T. A. Khan, S. H. Ling, and A. S. Mohan, "Advanced gravitational search algorithm with modified exploitation strategy," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 1056–1061.

[25] T. A. Taj, T. A. Khan, and I. Ijaz, "An enhanced harmony search (enhs) algorithm for solving optimization problems," in *Proceedings of the 2014 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference*. IEEE, 2014, pp. 84–88.