**Capstone Project**

The primary goal of this project is to understand and implement the process of training an AI agent to autonomously play the Flappy Bird game using computer vision and reinforcement learning. This task involves utilizing image-based inputs, such as frames from the game, and making decisions using reinforcement learning techniques. The project offers two distinct paths: the conceptual path, which focuses on understanding the theoretical framework behind the AI’s decision-making process, and the coding path, which requires practical implementation. In this report, we will cover the conceptual path and provide a thorough analysis of the steps involved in setting up an environment, implementing AI algorithms, and training a reinforcement learning agent to successfully play Flappy Bird.

This project is relevant because it applies concepts from artificial intelligence (AI) and machine learning to a real-time decision-making scenario. The task requires the agent to process visual data from the game, which can be compared to real-world applications where AI must navigate environments based on visual input. This is particularly valuable in fields such as robotics, autonomous vehicles, and gaming, where agents need to learn to make decisions in dynamic, uncertain environments. The core skills developed in this project—image processing, reinforcement learning, and model fine-tuning—are fundamental to more complex AI systems in real-world applications.

The Flappy Bird game consists of a simple environment where the player controls a bird that must avoid colliding with obstacles (pipes) by flapping to stay airborne. The bird is constantly subject to gravity, which pulls it downward, and the player must flap at the right times to avoid hitting the pipes or the ground. The game’s objective is to score points by passing the bird through pairs of pipes, and the longer the player survives, the higher the score. The game’s scoring system is relatively simple: the player earns one point for each pair of pipes successfully navigated.

In terms of setting up the environment for an AI agent to interact with, the first step is to represent the state of the game. In this context, the state consists of all the relevant information the agent needs to make a decision. A game state in Flappy Bird can be broken down into several key elements, such as the bird’s vertical position, its velocity, and the positions of the pipes. The agent needs to process the state to understand where the bird is in relation to the obstacles. This involves capturing the visual information in each frame of the game and converting it into a structured format that the agent can use.

In reinforcement learning (RL), the action space refers to the actions the agent can choose from. In the case of Flappy Bird, the action space is binary: the agent can either choose to flap the bird upward or allow it to fall due to gravity. The agent’s decision-making process is based on the current state of the game, such as how far the bird is from the ground and the proximity of the next set of pipes. The goal is to maximize survival time and accumulate points.

The reward system is another essential part of the environment setup. In RL, the agent learns by receiving feedback in the form of rewards or penalties based on its actions. The reward system should encourage the agent to keep the bird alive for as long as possible while passing through pipes. For example, the agent could receive a positive reward each time it successfully passes through a pair of pipes. Conversely, a negative reward or penalty could be assigned when the bird collides with a pipe or falls to the ground. The agent’s challenge is to learn how to navigate the pipes by flapping at the right moments and avoiding obstacles.

To facilitate the interaction between the AI agent and the game environment, tools like PyGame and PyGame Learning Environment (PLE) are commonly used. PyGame is a library for creating games, while PLE is an extension specifically designed for training AI agents in games like Flappy Bird. It simplifies the setup by providing a framework to interact with the game, retrieve the game state, take actions, and receive rewards in a standardized way. PLE works seamlessly with OpenAI Gym, a toolkit used for developing and comparing reinforcement learning algorithms. By integrating PyGame with Gym, we create an environment where the agent can observe the game state, perform actions, and learn from the results.

Because the raw visual inputs from the game are high-dimensional (i.e., the pixel values of each frame), preprocessing is crucial to make the data more manageable and computationally feasible. The preprocessing steps typically include grayscale conversion to eliminate the color information that is not necessary for decision-making, resizing the image to a smaller resolution (e.g., 84x84 pixels), and normalizing the pixel values so that they fall within a consistent range (e.g., 0 to 1). These steps reduce the computational burden while retaining the critical visual information needed by the agent to navigate the environment effectively.

One of the most powerful techniques in modern AI is transfer learning, which involves taking a pre-trained model—usually trained on a large dataset for a specific task—and adapting it to a new, but similar, task. In the case of the Flappy Bird AI, transfer learning helps to leverage the knowledge already learned by a pre-trained model, such as a Convolutional Neural Network (CNN), which has been trained on a large image dataset like ImageNet.

For Flappy Bird, we can use a pre-trained model like MobileNetV2, which is known for its efficiency and speed, making it well-suited for real-time applications. MobileNetV2 is a lightweight CNN that was originally trained on a large dataset, ImageNet, to recognize objects in images. By using it for our task, we benefit from the model’s ability to recognize general visual features, such as edges, textures, and patterns, which can then be applied to the game’s frames. The idea behind using MobileNetV2 is not to classify the objects in the game but rather to use it as a feature extractor—a model that processes game frames and outputs a set of features that can be used by another part of the system (such as the Q-network in reinforcement learning).

To modify MobileNetV2 for this task, we remove the final layers used for classification and repurpose the convolutional layers for extracting features from the game images. These features can then be passed into a reinforcement learning model, like a Q-network, to make decisions about whether to flap or not. This modification allows us to take advantage of MobileNetV2’s ability to capture spatial hierarchies and transform them into useful features for reinforcement learning.

One challenge when adapting a pre-trained model for Flappy Bird is the difference in the visual content between the original training images (e.g., everyday objects like cars, animals, and buildings) and the Flappy Bird environment. The model was trained on images that are fundamentally different from the game’s frames, which may make it harder for the model to recognize relevant features. To overcome this challenge, the pre-trained model can be fine-tuned, where certain layers are adjusted during training to better adapt the features to the task of playing Flappy Bird. Fine-tuning involves updating the weights of the pre-trained model to allow it to better detect features like the bird, pipes, and gaps.

At the heart of training an AI agent to play Flappy Bird is the concept of reinforcement learning. In RL, an agent learns to take actions in an environment to maximize cumulative rewards. This learning process is driven by feedback from the environment, which is provided in the form of rewards or penalties. The agent’s task is to learn a policy—a mapping from states to actions—that maximizes its long-term expected reward.

In the context of Flappy Bird, we can use the Q-learning algorithm, which is a foundational RL algorithm. Q-learning estimates the Q-values, which represent the expected future rewards for taking a specific action in a given state. In the case of Flappy Bird, the state consists of the features extracted from the game frame, and the actions correspond to the binary choices: flap or not flap. The goal of Q-learning is to find the best policy by updating the Q-values using the Bellman equation, which expresses the relationship between current and future rewards.

To implement this, we use a Deep Q-Network (DQN), which is a deep neural network that approximates the Q-value function. A DQN takes the state as input (the preprocessed game frame) and outputs Q-values for each action (flap or no flap). The agent will then choose the action corresponding to the highest Q-value. The training process involves updating the Q-values based on the reward received after each action. The epsilon-greedy policy is used to balance exploration (trying new actions) and exploitation (choosing the best-known action). Initially, the agent explores more by selecting random actions, but over time, it becomes more exploitative by choosing the action with the highest Q-value.

The concept of experience replay is crucial for stable learning. In experience replay, the agent stores its experiences (state, action, reward, next state) in a memory buffer. During training, random samples of experiences are taken from the buffer to train the model, which helps break the correlation between consecutive experiences and leads to better generalization. This technique improves the stability and efficiency of training.

Additionally, a target network is often used in DQNs to improve the stability of training. The target network is a copy of the Q-network that is updated less frequently. This allows the agent to target more stable Q-values during training, preventing the model from fluctuating too quickly due to frequent updates.

Training the AI agent to play Flappy Bird is an iterative process that involves running multiple episodes of the game. Each episode starts with the bird at the beginning of the game, and the agent takes actions to interact with the environment. The agent’s actions are based on the current state, and after each action, the environment provides feedback in the form of rewards or penalties. Over time, the agent updates its Q-values using these experiences and refines its policy.

Key hyperparameters for training include the learning rate, which controls how much the Q-values are updated during training, the discount factor (gamma), which determines how much future rewards are valued relative to immediate rewards, and the epsilon value, which controls the exploration-exploitation trade-off. These hyperparameters need to be carefully tuned to balance the speed and stability of training.

A potential challenge during training is catastrophic forgetting, where the agent forgets previously learned behaviors because it overfits to more recent experiences. To mitigate this, the agent can store experiences in a replay buffer, allowing it to sample a diverse set of experiences from throughout the training process.

After the agent has been trained, it is essential to evaluate its performance. This involves running the trained agent in the game environment and measuring its ability to survive and score points. The key performance metrics are the average score, which indicates how many pipes the agent successfully navigates, and the average survival time, which measures how long the agent survives before crashing into an obstacle.

Testing should be conducted across multiple episodes to understand how well the agent generalizes to different scenarios. Visualization techniques can be used to gain insights into the agent’s decision-making process by tracking the actions it takes in various game situations. For example, you can plot the bird’s movements and observe how it reacts to different pipe arrangements and obstacles.

The results can be analyzed to identify strengths and weaknesses in the agent’s behavior. If the agent performs poorly, adjustments to the reward system, training parameters, or model architecture may be needed to improve its performance.

The Flappy Bird AI project serves as an excellent introduction to reinforcement learning and the challenges of training AI agents to interact with dynamic environments. By leveraging pre-trained models, fine-tuning them for the task, and implementing a Q-learning-based reinforcement learning algorithm, this project highlights the importance of model design, data preprocessing, and training techniques in building effective AI systems. The insights gained from this project are valuable for understanding how AI systems can be applied to a wide range of real-time, interactive scenarios, from gaming to robotics and beyond.

**Bibliography**:

1. Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533. https://doi.org/10.1038/nature14236

This paper introduces the Deep Q-Network (DQN) algorithm, which combines deep learning with Q-learning to train agents to play Atari games. The concepts from this paper are foundational to the reinforcement learning approach used in this project.

1. Szegedy, C., Liu, W., Jia, Y., et al. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 1-9).

This paper introduces GoogLeNet and discusses the use of deep convolutional neural networks (CNNs) in image classification tasks. While not directly related to Flappy Bird, it provides important insights into CNN architectures like MobileNet, which was used for feature extraction in this project.

1. Howard, A. G., Zhu, M., Chen, B., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

This paper presents MobileNets, a lightweight and efficient CNN architecture that balances speed and accuracy, making it suitable for real-time applications like playing Flappy Bird. The use of MobileNetV2 in this project follows the principles outlined in this paper.

1. Silver, D., Huang, A., Maddison, C. J., et al. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489. https://doi.org/10.1038/nature16961

This groundbreaking paper describes how AlphaGo uses deep reinforcement learning to master the game of Go. The reinforcement learning principles discussed in this paper were key to understanding the theoretical framework for training an agent in Flappy Bird.

1. Bellman, R. (1957). Dynamic programming. Princeton University Press.

The Bellman equation is a cornerstone of reinforcement learning, and this book by Richard Bellman provides the theoretical foundation for dynamic programming, which is central to algorithms like Q-learning.

1. OpenAI. (2021). OpenAI Gym. https://gym.openai.com

OpenAI Gym is a widely used toolkit for developing and comparing reinforcement learning algorithms. This resource was integral in understanding how reinforcement learning environments can be structured for projects like Flappy Bird.

1. Zhang, X., Zhao, Z., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in Neural Information Processing Systems (NeurIPS) (Vol. 28).

Although primarily focused on text, this paper's discussion of convolutional layers for feature extraction is relevant to understanding how CNNs can process image data, a key part of preprocessing game frames for Flappy Bird.

1. DeepMind. (2013). Playing Atari with deep reinforcement learning. https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

This technical report by DeepMind provides an early look into the application of deep Q-learning for playing Atari games, which is conceptually similar to the task of training an AI agent for Flappy Bird.

1. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

This textbook provides a comprehensive introduction to reinforcement learning, covering key algorithms and concepts such as Q-learning, policy gradients, and the exploration-exploitation trade-off. It is a critical resource for understanding the reinforcement learning techniques used in this project.

1. Brockman, G., Cheung, V., Pettersson, L., et al. (2016). OpenAI Gym. In Proceedings of the 2016 Conference on Artificial Intelligence (AI).

OpenAI Gym, referenced in the context of setting up the Flappy Bird environment, is an essential toolkit for creating and evaluating reinforcement learning environments and agents. This paper outlines the Gym framework’s functionality and its applications.