**Methodology**

**1. Environment Setup:**

Imported the necessary libraries; gym\_super\_mario\_bros for the Super Mario Bros environment and nes\_py for utilizing wrappers. Then we used ‘JoypadSpace’ to simplify controls through SIMPLE\_MOVEMENT and then set up the initial game environment, observation space, and action space. At last, we created a flag and set it to done to track episode completion.

**2. Random Action Loop:**

Iterating through frames in a loop, we took random actions in the environment. Whenever an episode concluded, we reset the environment to commence a new one, rendering each step to visualize our agent's actions.

**3. Wrappers for Preprocessing:**

Importing wrappers for preprocessing, such as GrayScaleObservation to convert frames to grayscale, we utilized ‘DummyVecEnv’ from Stable Baselines to create a vectorized environment for efficient training. Applying frame stacking, passed the stacked frames to the agent using VecFrameStack which provided temporal information to our agent.

**4. Observation Space and State Exploration:**

Checking the modified observation space, we verified that the frames were now grayscaled. Employing the env.step method, we performed actions (e.g., moving right) and observed resulting states, rewards, and other information.

**5. Reinforcement Learning Setup:**

Importing necessary libraries for reinforcement learning, including PPO from Stable Baselines, we defined a custom callback class (TrainAndLoggingCallback) to save the model during training. Directories were set up for model checkpoints and logs.

**6. PPO Model Initialization:**

We then initialized the PPO model with specific parameters, including which neural network architecture to use (CnnPolicy), the learning rate, and number of steps per update. The model was configured to use the provided Super Mario environment (env).

**7. Training the PPO Model:**

Using the model.learn method, we trained the PPO model for a specified number of total timesteps (1,00,000). During training, the custom callback that we defined earlier saved the model every check\_freq timesteps.

**8. Loading and Playing the game with the Trained Model:**

Loading the trained model using PPO.load from a specified checkpoint file. We reset the environment and initiating the game, we used the trained model to predict actions and played the game. Rendering the environment allowed us to visualize our agent's performance.