# Instructions on how my project was trained

First, please take into account the requirements listed below to create an appropriate Python environment.

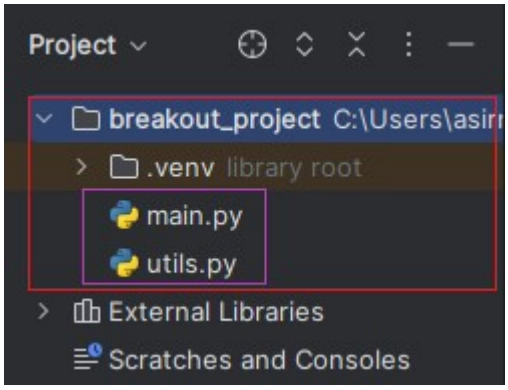
Python Version: 3.10.11

Required Libraries

- pip install gym==0.22.0
- pip install gym[atari]==0.22.0
- pip install gym[accept-rom-license]
- pip install opencv-python
- pip install psutil
- pip install tensorflow==2.13.0
- pip install matplotlib==3.8.2

# How to train from scratch:

From Source Code folder, copy “main.py” and “utils.py” to working directory so that the structure would be like following:




My working directory name is “breakout_project”

breakout_project\main.py

breakout_project\utils.py

After that, open “main.py” and find the following lines of code at the end of the file.

Simply, set the order as follows if you want to train model from scratch.

```
388
389 ▷ if __name__ == "__main__":
390         atariBreakout(train=True, load=False, plot=False, render=False)
391
```

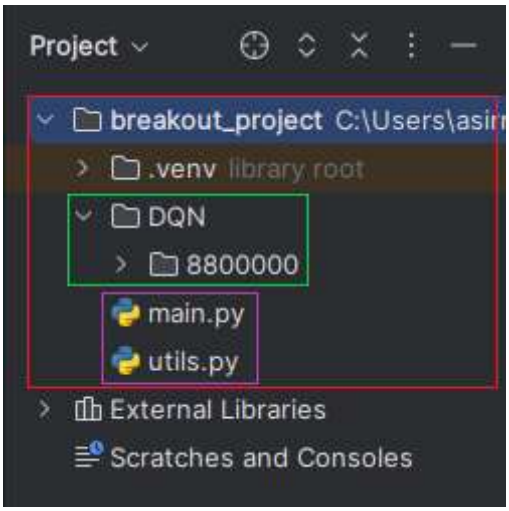
Now you can wait till it reaches 10 million frames and can see the progress.

```
New Best Score!
Best Score: 0.29, Score: 0.29, Frame Number 10000, Epsilon 0.996, Time 42.91Sec, Total Time 0.72Min, Memory Usage 3803.21MB
Best Score: 0.29, Score: 0.28, Frame Number 20000, Epsilon 0.992, Time 41.24Sec, Total Time 1.40Min, Memory Usage 6194.33MB
Best Score: 0.29, Score: 0.25, Frame Number 30000, Epsilon 0.988, Time 40.99Sec, Total Time 2.09Min, Memory Usage 6362.02MB
Best Score: 0.29, Score: 0.29, Frame Number 40000, Epsilon 0.984, Time 40.71Sec, Total Time 2.76Min, Memory Usage 6502.05MB
Best Score: 0.29, Score: 0.27, Frame Number 50000, Epsilon 0.980, Time 41.09Sec, Total Time 3.45Min, Memory Usage 6644.61MB
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
INFO:tensorflow:Assets written to: DQN/50000/models/breakout_model_final/assets
INFO:tensorflow:Assets written to: DQN/50000/models/breakout_model_final/assets
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
INFO:tensorflow:Assets written to: DQN/50000/target_models/breakout_model_final/assets
INFO:tensorflow:Assets written to: DQN/50000/target_models/breakout_model_final/assets
```

In case of kernel crash or any interruption, go to next part “How to train using pre-trained model”

# How to train using pre-trained model:

From Source Code folder, copy “main.py” and “utils.py” to working directory. Also, from Model folder, copy DQN folder to working directory so that the structure would be like following:




My working directory name is “breakout_project”

breakout_project\DQN

breakout_project\DQN\8800000

breakout_project\main.py

breakout_project\utils.py

After that, open “main.py” and find the following lines of code at the end of the file.

Simply, set the order as follows if you want to train model using pre-trained model.

```
388
389 ▷   if __name__ == "__main__":
390         atariBreakout(train=True, load=True, plot=False, render=False)
391
```

Run the “main.py” file after you set these parameters. The interpreter asks you to enter input, in this case you need to enter 8800000 as “frame number”.

```
A.L.E: Arcade Learning Environment (version 0.7.5+db37282)
[Powered by Stella]
2024-02-02 01:04:26.802170: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Enter frame number8800000
```

Check the DQN folder to get that number. The “frame number” can be changed after the training. Before loading a model, be careful regarding the “frame number”. Now you can wait till it reaches 10 million frames and can see the progress.

```
Enter frame number 8800000
Model, Target Model and States are loaded successfully.
New Best Score!
Best Score: 21.61, Score: 21.61, Frame Number 8810000, Epsilon 0.010, Time 64.77Sec, Total Time 916.63Min, Memory Usage 6925.77MB
Best Score: 21.61, Score: 20.92, Frame Number 8820000, Epsilon 0.010, Time 72.20Sec, Total Time 917.83Min, Memory Usage 7496.40MB
```

# Detailed Information Regarding the Training

I implemented this project by the help of the following papers

- Playing Atari with Deep Reinforcement Learning (the 2013 vanilla-DQN paper)
- Human-level control through deep reinforcement learning (the 2015 Nature paper)

### A brief explanation about the training.

Deep Q-Learning algorithm was used to train this Agent. This process involves the agent perceiving its environment through situation observations and choosing an action by evaluating these situations. The agent accumulates experience by interacting with its environment and calculates target Q-values using this experience. It then updates the network by calculating the error between the actual and predicted Q-values. This process allows the agent to learn and optimize complex tasks in its environment. With hyper parameter tuning and iterations, the agent's performance improves over time and it becomes better able to make better decisions on a given task.

In addition, I used epsilon-greedy strategy. This strategy helps the agent randomly selects an action with a certain epsilon probability at each step, allowing it to better understand and explore the environment. However, over time, the epsilon value decreases, directing the agent to choose the best-predicted action more frequently as it gains more confidence. In my project, starting epsilon value is 1.0 and minimum epsilon value is 0.01. In each ten thousand frames, epsilon value decreases by 0.0000004.

### Neural Network

I used convolutional neural networks to work with image input data. My neural network consists of a sequence of layers, including three convolutional layers, a flattening layer, and two fully connected layers. The first convolutional layer has 32 filters, an 8x8 kernel size, a stride of 4, and uses the ReLU activation function. The second convolutional layer has 64 filters, a 4x4 kernel size, a stride of 2, and also uses ReLU activation. The third convolutional layer has 64 filters, a 3x3 kernel size, a stride of 1, and uses ReLU activation. The flattening layer transforms the output of the convolutional layers into a vector. After that, a fully connected layer with 512 neurons and ReLU activation is added, followed by an output layer with a number of neurons corresponding to the specified number of actions, in breakout atari game it is 4, using a linear activation function.

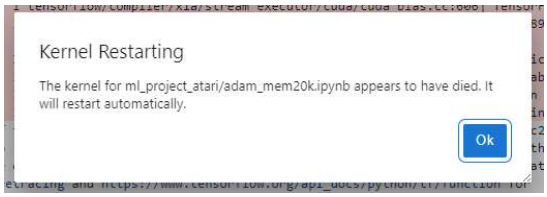
### Preparation of environment

As mentioned in the 2015 paper, I used several wrappers to achieve better results. These enhancements include introducing no-operation resets with a maximum of 30 steps, skips frames for faster training, terminates episodes on life loss, triggers a "fire" action at the start, preprocesses frames to grayscale and resizes them, scales pixel values, clips rewards, and creates a buffer stacking the last 4 frames for temporal information capture.

# Results and Information regarding the training

It's been trained on the Hof University Jupyter server with NVIDIA A100 80GB PCIe MIG 3g.40 GB, Compute Performance 8.0 graphics cards.

When it was training, kernel was always crashed for each 6 hours. To address this problem, I've created a function to save the game data in every 50 thousand frames into saving directory and also added one more function that will remove older game data from the saving directory after 100 thousand frames so there are always two game data.




Training took approximately 17 hours for 10 million frames and I reached maximum 21.13 clipped score at 8800000$^{th}$ frame.

21.13 clipped score refers to approximately 400 score in real game