Maintenance and User Manual

June 6, 2022

Chapter 1

User Manual

This document describes the usage and basic functioning of the Proximal Policy Optimization of the neural network AI for the Battle for Wesnoth computer game

1.1 General Operating Considerations

The PPO AI for the BFW was designed and tested in the Windows Services for Linux environment with Ubuntu Linux distribution. The software should also function in a standalone Linux computer. The software can be launched from the Linux prompt and from the Visual Studio Code Integrated Development Environment.

1.2 Operation from the Linux Command Prompt

- In Linux prompt, change the current directory to BFW PPO project directory, for example:

  ```
  cd ~/bfd-ppo
  ```

- Command to start training:

  ```
  python PPO/train.py
  ```

- Display usage and optional training command arguments help screen:

  ```
  python PPO/train.py -h
  ```

- Usage and optional argument listing help screen:

```
usage: train.py [-h] [--exp-name EXP_NAME] [--gym-id GYM_ID]
       [--seed SEED] [--cuda CUDA] [--showgui SHOWGUI]
       [--variations VARIATIONS] [--rotation ROTATION]
       [--mutations MUTATIONS]
       [--torch-deterministic TORCH_DETERMINISTIC]
       [--total-timesteps TOTAL_TIMESTEPS]
       [--batch-size BATCH_SIZE]
       [--mini-batch-size MINI_BATCH_SIZE]
       [--learning-rate LEARNING_RATE] [--gamma GAMMA]
       [--gae-lambda GAE_LAMBDA] [--update-epochs UPDATE_EPOCHS]
       [--clip-coef CLIP_COEF] [--ent-coef ENT_COEF]
       [--vf-coef VF_COEF] [--epsilon EPSILON]

optional arguments:
-h, --help                  show this help message and exit
--exp-name EXP_NAME         The name of this experiment
--gym-id GYM_ID             The id of the gym environment
--seed SEED                 Seed of the experiment random number
                            generator
--cuda CUDA                 Toggles cuda GPU acceleration
--showgui SHOWGUI           Toggles the environment GUI
--variations VARIATIONS     Sets the game map variations, how often
                            the map will be updated or mutated
--rotation ROTATION         Sets the game map rotation, how many maps
                            will be included in map rotation, range 0-5
--mutations MUTATIONS       How many times to mutate the game map
--torch-deterministic TORCH_DETERMINISTIC
                            Toggles ‘torch.backends.cudnn.deterministic‘
--total-timesteps TOTAL_TIMESTEPS
                            Total timesteps of the experiments
--batch-size BATCH_SIZE     The size of the batch, the number of steps
                            to run in each environment per policy rollout
--mini-batch-size MINI_BATCH_SIZE
                            The size of the mini batch, the number
                            of steps per training iteration
--learning-rate LEARNING_RATE
                            The learning rate of the optimizer
--gamma GAMMA               The discount factor gamma for the general
                            advantage estimation
--gae-lambda GAE_LAMBDA     The lambda for the general advantage estimation
--update-epochs UPDATE_EPOCHS
                            The number of epochs to update the policy
--clip-coef CLIP_COEF       The surrogate clipping coefficient
--ent-coef ENT_COEF         Coefficient of the entropy, c2
--vf-coef VF_COEF           Coefficient of the value function, c1
--epsilon EPSILON           Optimizer epsilon
```

- Command to start AI game play on the screen after successful training

  ```
  python PPO/play.py --showgui True
  ```

1.3 Operation from VS Code IDE

- In WSL Linux prompt, change the current directory to BFW PPO project directory, for example:

  ```
  cd ~/bfd-ppo
  ```

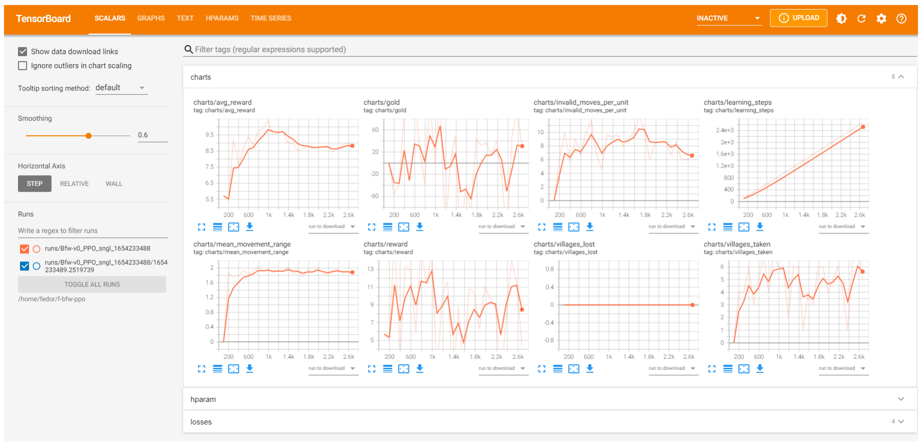
- Start VS Code IDE:

  ```
  code .
  ```

- Open Run Debug panel - type *Ctrl-Shift-D* or click *Run & Debug button.*
- To start training, select *PPO train single player* from the drop-down menu and click the green arrow next to it.
- The same arguments that are listed in the previous section can be added or modified by editing the *launch.json* file in the project *.vscode* folder. The PPO training section of the launch file is named *PPO train single player.* Example launch configuration:

  ```
  "name": "PPO train single player",
  "type": "python",
  "request": "launch",
  "program": "${workspaceFolder}/PPO/train.py",
  "args": [
      "--exp-name","PPO_sngl",
      "--learning-rate","0.0003",
      "--total-timesteps", "2000000",
      "--cuda","False",
      "--clip-coef","0.1",
      "--ent-coef","0.0",
      "--batch-size","25",
      "--mini-batch-size","5",
      "--vf-coef","0.5",
      "--variations","2000000",
      "--mutations","0",
      "--update-epochs","4",
      "--epsilon","1e-6",
      "--seed","1"
  ]
  ```

- To start AI game play on the screen after successful training, select *PPO play single player* from the drop-down menu and click the green arrow next to it..



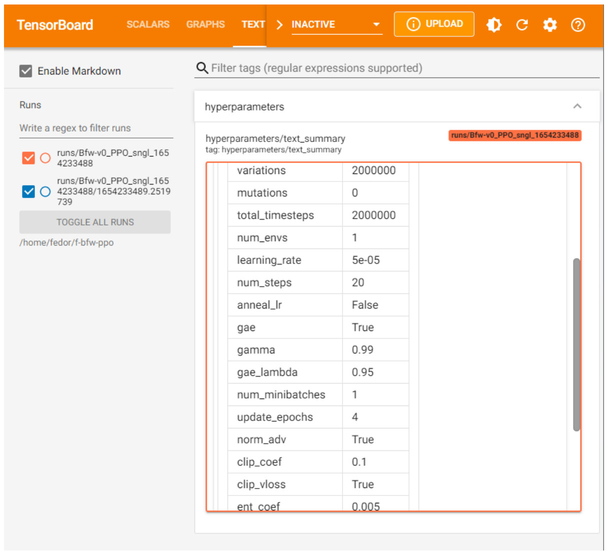
☑ Show data download links
☐ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis: STEP
Runs
☑ runs/Bfw-v0_PPO_sngl_1654233488
☑ runs/Bfw-v0_PPO_sngl_1654233488/1654233489.2519739
/home/fedor/f-bfw-ppo

Figure 1.1: Monitoring training progress via Tensorboard server. The vital stats of the ongoing AI training are charted under *SCALARS* tab.

1.4 Training Progress Monitor

The PPO training session for BFW can take many hours. Therefore, it is critical to monitor the state of the training, so one should be able to stop the training and adjust the hyperparameters if the results are undesirable. The train.py program continuously logs the stats of the ongoing training to the Tensorboard server. The training progress can be observed in web browser at the url *http://localhost:6006*

Figure 1.1 shows a screenshot of the ongoing training session. The progress can be evaluated by observing multiple charts including *avg reward*, *villages taken*, *illegal moves*, and a few others. The train.py program also logs the hyperparameters associated with the particular run as shown on Figure 1.2



runs/Bfw-v0_PPO_sngl_1654233488

runs/Bfw-v0_PPO_sngl_1654233488/1654233489.2519739

/home/fedor/f-bfw-ppo

hyperparameters/text_summary

tag: hyperparameters/text_summary

runs/Bfw-v0_PPO_sngl_1654233488

variations	2000000
mutations	0
total_timesteps	2000000
num_envs	1
learning_rate	5e-05
num_steps	20
anneal_lr	False
gae	True
gamma	0.99
gae_lambda	0.95
num_minibatches	1
update_epochs	4
norm_adv	True
clip_coef	0.1
clip_vloss	True
ent_coef	0.005

Figure 1.2: Hyperparameters log. The hyperparameters associated with a particular training run can be observed and analyzed under either *TEXT* or *HPARAMS* tabs.

Chapter 2

Maintenance Manual

This document provides instructions for setting up the environment required to run PPO training of the AI for BFW

2.1 General Maintenance Considerations

The PPO AI for the BFW was developed on a Windows PC using Windows Services for Linux environment with the default Ubuntu Linux distribution. This manual describers all of the steps for Windows installation with a virtual Linux environment. For the stand-alone Linux computer one can skip to *B.3 Linux Environment Setup* section.

2.2 Development and Testing Environment Setup under Windows

- Install Windows Services for Linux from Windows command prompt with elevated privileges:

```
wsl --install
```

- Start Ubuntu app from Windows start menu, create a user account. After logging into Linux, clone BFW-PPO project:

```
git clone https://github.com/ivanbalak/BFW-PPO.git
```

- Start Visual Studio Code in Linux environment from the project folder, then continue with section *B.3 Linux Environment Setup*:

```
cd bff-ppo
code .
```

2.3 Linux Environment Setup

- Install Battle for Wesnoth:

  ```
  sudo apt-get install wesnoth
  ```

- If the project is not yet cloned in previous steps, clone BFW-PPO project:

  ```
  git clone https://github.com/ivanbalak/BFW-PPO.git
  ```

- Copy Bfw-gym add-on to wesnoth add-on folder:

  ```
  cp PythonAddon ~/.config/wesnoth-1.14/data/add-ons
  ```

- Create a folder for input files for use by BFW gym:

  ```
  mkdir ~/.config/wesnoth-1.14/data/input
  ```

- Configure Python environment and project dependencies. The project was tested with Python 3.8, but it can be compatible with later versions of Python. Below is the command sequence specific to Python 3.8:

  ```
  python3.8 -m venv env
  source env/bin/activate
  pip install -U pip wheel setuptools
  pip3 install -e ./gym-bfw
  pip3 install -r requirements.txt
  ```

- Train the BFW AI and test it with the game as outlined in the User Manual(Appendix A).