Bipedal Walker Deep Reinforcement Learning

Vidya Giri

Video Demos

- video of model with minimal training
- video of optimized model

Github Repository

- vidgi/bipedal-walker

Project Overview

In this project, I will be working on using RL models to teach a bipedal walker to walk! For this project, I used the Open Gym AI bipedalwalker-v2 environment and stable baselines to train it with deep reinforcement learning. I calculated the mean reward over episodes in order to evaluate the RL agent and used Proximal Policy Optimization in order to define, train, and evaluate the model. To compare and contrast evaluation timesteps, I first tried the procedure with 50,000 steps and then tried it with 500,000 steps for the normal bipedal walker environment. Then, looking at Tensorboard, I was able to see that the better results were after training it ~150,000 steps, so I restarted and got a much better improvement in the mean reward with the bipedal walker having much better results.

Approach

Installing system wide packages

Here we will install the linux server packages needed for the project using `apt-get` and python packages using `pip`

```
!apt-get install swig cmake libopenmpi-dev zlib1g-dev xvfb x11-utils
ffmpeg -qq #remove -qq for full output
!pip install stable-baselines[mpi] box2d box2d-kengz pyvirtualdisplay
pyglet==1.3.1 --quiet #remove --quiet for full output
%tensorflow_version 1.x # Stable Baselines only supports tensorflow
1.x for now
```

```
Selecting previously unselected package libxxf86dga1:amd64.
(Reading database ... 155222 files and directories currently
installed.)
Preparing to unpack .../libxxf86dga1_2%3a1.1.4-1_amd64.deb ...
Unpacking libxxf86dga1:amd64 (2:1.1.4-1) ...
Selecting previously unselected package swig3.0.
Preparing to unpack .../swig3.0_3.0.12-1_amd64.deb ...
Unpacking swig3.0 (3.0.12-1) ...
Selecting previously unselected package swig.
Preparing to unpack .../swig_3.0.12-1_amd64.deb ...
```

```
Unpacking swig (3.0.12-1) ...
Selecting previously unselected package x11-utils.
Preparing to unpack .../x11-utils_7.7+3build1_amd64.deb ...
Unpacking x11-utils (7.7+3build1) ...
Selecting previously unselected package xvfb.
Preparing to unpack .../xvfb_2%3a1.19.6-1ubuntu4.9_amd64.deb ...
Unpacking xvfb (2:1.19.6-1ubuntu4.9) ...
Setting up swig3.0 (3.0.12-1) ...
Setting up xvfb (2:1.19.6-1ubuntu4.9) ...
Setting up libxxf86dga1:amd64 (2:1.1.4-1) ...
Setting up swig (3.0.12-1) ...
Setting up x11-utils (7.7+3build1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.3) ...
/sbin/ldconfig.real:
/usr/local/lib/python3.7/dist-packages/ideep4py/lib/libmkldnn.so.0 is
not a symbolic link

ents to build wheel ... etadata ...  (setup.py) ... pi4py (PEP
517) ... ERROR: pip's dependency resolver does not currently take into
account all the packages that are installed. This behaviour is the
source of the following dependency conflicts.
tensorflow-probability 0.15.0 requires cloudpickle>=1.3, but you have
cloudpickle 1.2.2 which is incompatible.
TensorFlow 1.x selected.
```

Project Dependencies

Now, we will import all the packages and dependencies required to run and train the model and record the demo video. Colab does not support env.render() which you typically use to view results in standard python notebooks so I used a workaround that allows us to emulate the display, record the video, and then display it.

```
import gym
import imageio
import numpy as np
import base64
import IPython
import PIL.Image
import pyvirtualdisplay

# Video stuff
from pathlib import Path
from IPython import display as ipythondisplay

from stable_baselines.common.policies import MlpPolicy
from stable_baselines.common.vec_env import VecVideoRecorder,
SubprocVecEnv, DummyVecEnv
from stable_baselines import PPO2
```

```
WARNING:tensorflow:
The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-
contrib-sunset.md
  * https://github.com/tensorflow/addons
  * https://github.com/tensorflow/io (for I/O related ops)
If you depend on functionality not listed there, please file an issue.
```

Define variables/functions

Here I will define the variables and then create an evaluate, record_video, and show_video functions that will be used later to train the model for the bipedal walker and show the results.

```
# set environment variables
env_id = 'BipedalWalker-v2'
video_folder = '/videos'
video_length = 100

# set our inital enviorment
env = DummyVecEnv([lambda: gym.make(env_id)])
obs = env.reset()

# define the evaluate function
def evaluate(model, num_steps=1000):
  """
  Evaluate a RL agent
  :param model: (BaseRLModel object) the RL Agent
  :param num_steps: (int) number of timesteps to evaluate it
  :return: (float) Mean reward for the last 100 episodes
  """
  episode_rewards = [0.0]
  obs = env.reset()
  for i in range(num_steps):
      # _states are only useful when using LSTM policies
      action, _states = model.predict(obs)

      obs, reward, done, info = env.step(action)

      # stats
      episode_rewards[-1] += reward
      if done:
          obs = env.reset()
          episode_rewards.append(0.0)
  # compute the mean reward for the last 100 episodes
  mean_100ep_reward = round(np.mean(episode_rewards[-100:]), 1)
  print("Mean reward:", mean_100ep_reward, "Num episodes:",
len(episode_rewards))
```

```
  return mean_100ep_reward

# make video and set up emulated display (otherwise rendering will
fail)
import os
os.system("Xvfb :1 -screen 0 1024x768x24 &")
os.environ['DISPLAY'] = ':1'

# define the record_video function
def record_video(env_id, model, video_length=500, prefix='',
video_folder='videos/'):
  """
  :param env_id: (str)
  :param model: (RL model)
  :param video_length: (int)
  :param prefix: (str)
  :param video_folder: (str)
  """
  eval_env = DummyVecEnv([lambda: gym.make('BipedalWalker-v2')])
  # Start the video at step=0 and record 500 steps
  eval_env = VecVideoRecorder(env, video_folder=video_folder,
                              record_video_trigger=lambda step: step
== 0, video_length=video_length,
                              name_prefix=prefix)

  obs = eval_env.reset()
  for _ in range(video_length):
    action, _ = model.predict(obs)
    obs, _, _, _ = eval_env.step(action)

  # close the video recorder
  eval_env.close()

# display video
def show_videos(video_path='', prefix=''):
  html = []
  for mp4 in Path(video_path).glob("{}*.mp4".format(prefix)):
      video_b64 = base64.b64encode(mp4.read_bytes())
      html.append('''<video alt="{}" autoplay
                    loop controls style="height: 400px;">
                    <source src="data:video/mp4;base64,{}"
type="video/mp4" />
                </video>'''.format(mp4, video_b64.decode('ascii')))
  ipythondisplay.display(ipythondisplay.HTML(data="<br>".join(html)))
```

Configure the reinforcement learning algorithim

Then I used the PPO2/Proximal Policy Optimization for the reinforcement learning algorithim.

```python
# define the model and output logs to tensorboard for
metric/reward/discounted reward tracking
model = PPO2(MlpPolicy, env, gamma=0.99, verbose=0,
tensorboard_log="./logs/")
```

Result

Train bipedalwalker model 50k steps & evaluate results

Here we train, evaluate, save, record & display video. As you can see, the mean reward before training was -98.7, and then after training, there was not yet an improvement with -104.0, and we see the model is still pretty poor at controlling the walker.

```python
# Random Agent, before training
mean_reward_before_train = evaluate(model, num_steps=10000)

# Train model
model.learn(total_timesteps=50000)

# Save model
model.save("ppo2-walker-50000")

# Random Agent, after training
mean_reward_after_train = evaluate(model, num_steps=10000)
```

```
Mean reward: -98.7 Num episodes: 11
WARNING:tensorflow:From
/tensorflow-1.15.2/python3.7/stable_baselines/common/base_class.py:488
: The name tf.summary.FileWriter is deprecated. Please use
tf.compat.v1.summary.FileWriter instead.

WARNING:tensorflow:From
/tensorflow-1.15.2/python3.7/stable_baselines/ppo2/ppo2.py:240: The
name tf.RunOptions is deprecated. Please use tf.compat.v1.RunOptions
instead.

WARNING:tensorflow:From
/tensorflow-1.15.2/python3.7/stable_baselines/ppo2/ppo2.py:241: The
name tf.RunMetadata is deprecated. Please use tf.compat.v1.RunMetadata
instead.

WARNING:tensorflow:From
/tensorflow-1.15.2/python3.7/stable_baselines/a2c/utils.py:582: The
name tf.Summary is deprecated. Please use tf.compat.v1.Summary
instead.

Mean reward: -104.0 Num episodes: 149
```

```
# Record & show video
record_video('BipedalWalker-v2', model, video_length=1500,
prefix='ppo2-walker-50000')
show_videos('videos', prefix='ppo2-walker-50000')
```

```
Saving video to  /content/videos/ppo2-walker-50000-step-0-to-step-
1500.mp4
```

```
<IPython.core.display.HTML object>
```

Train bipedalwalker model for 100k more steps & evaluate results

Then, after trying it with 500k, the peak at 100k seemed to have the best reward results, and then the rewards started going down with more training. So, I decided on adding additional training for 100k to get the best mean reward. As you can see there is a drastic improvement from -103.7 to -2.7, which can be seen in the video where the bipedal walker is able to walk across the ground.

```
# Random Agent, before training
mean_reward_before_train = evaluate(model, num_steps=10000)

# Train model
model.learn(total_timesteps=100000)

# Save model
model.save("ppo2-walker-100000")

# Random Agent, after training
mean_reward_after_train = evaluate(model, num_steps=10000)
```

```
Mean reward: -103.7 Num episodes: 155
Mean reward: -2.7 Num episodes: 10
```

```
# Record & show video
record_video('BipedalWalker-v2', model, video_length=3500,
prefix='ppo2-walker-100000')
show_videos('videos', prefix='ppo2-walker-100000')
```

```
Saving video to  /content/videos/ppo2-walker-100000-step-0-to-step-
3000.mp4
```

```
<IPython.core.display.HTML object>
```

```
%tensorflow_version 1.x
%load_ext tensorboard
%tensorboard --logdir logs
```

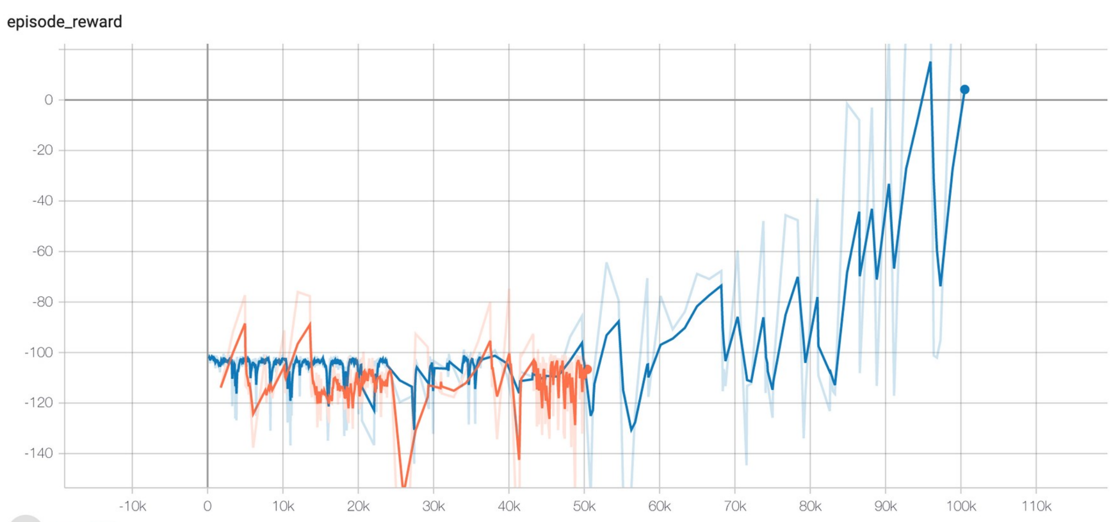
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

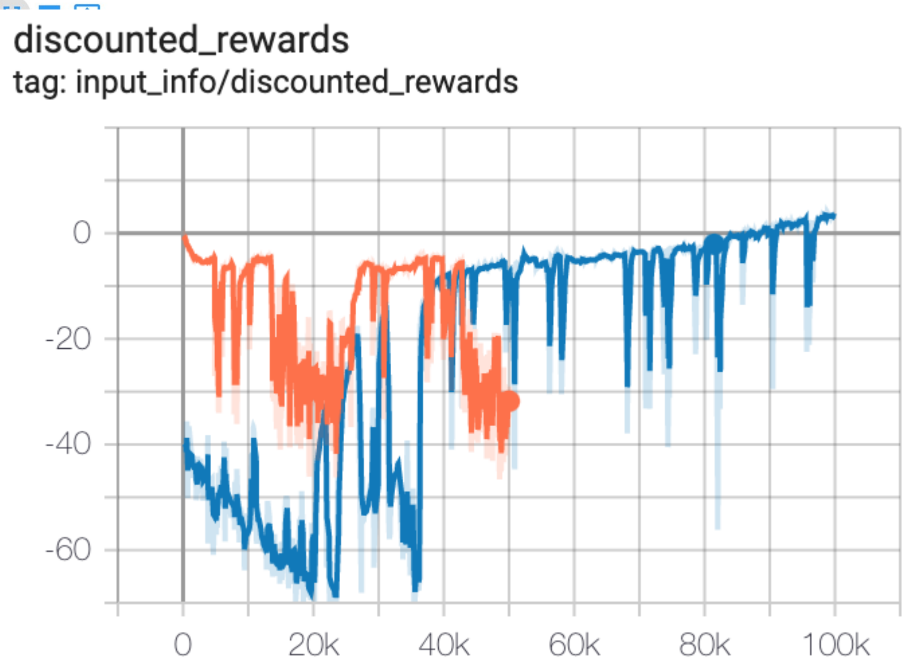
```
Reusing TensorBoard on port 6006 (pid 1051), started 0:04:23 ago. (Use
'!kill 1051' to kill it.)
```

<IPython.core.display.HTML object>

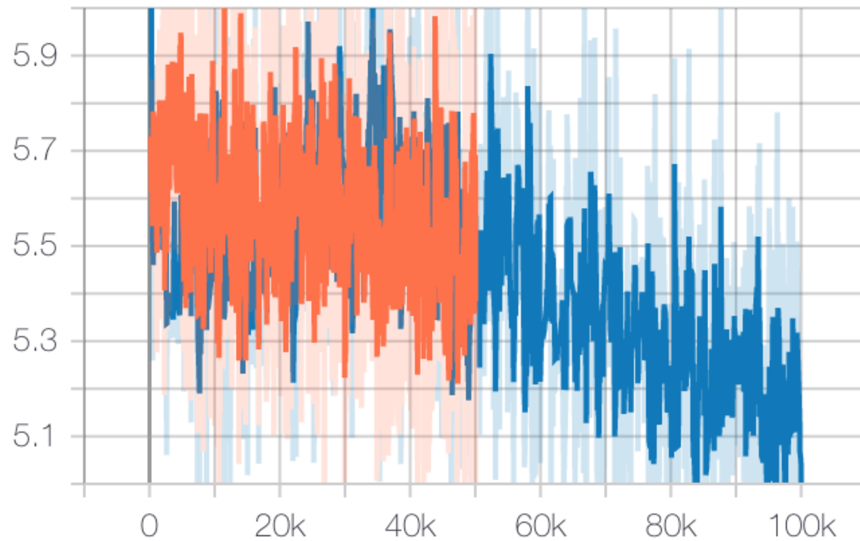
Tensorboard Results

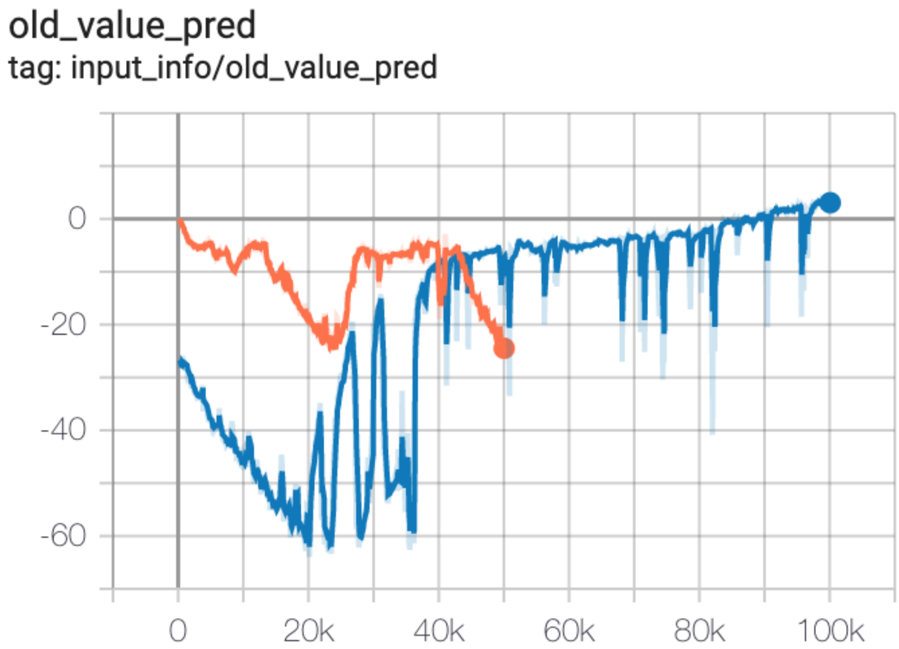
Below, I have documented the tensorboard results comparing the 50,000 steps model to the additional 100,000 steps model where you can see the sharp improvement in the episode rewards and discounted rewards. I have also added to plots for action probability, old value pred, and advantage.

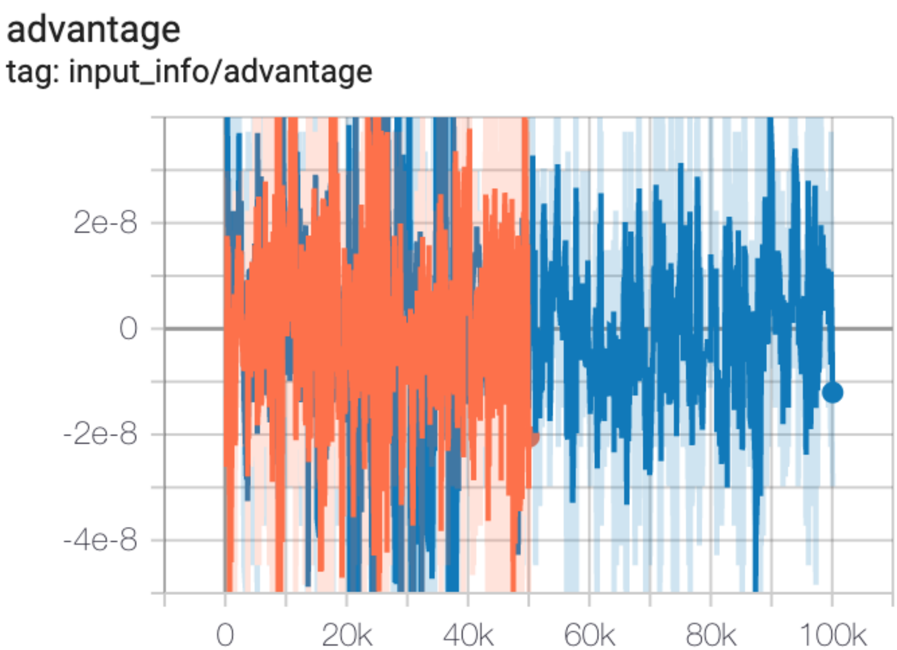




old_neglog_action_probabilty
tag: input_info/old_neglog_action_probabilty







Conclusion

I tried to further parameter optimization as described here, tweaking the default parameters, however I did not see any improvement with a lower gamma. I would also be interested in trying another type of RL learning algorithim implementation. One I found that might have better results is elegant RL which is supposed to be a more stable implementation of state-of-the-art DRL algorithms when compared to stable-baselines which I used for my project. I would also be interested in maybe trying to train a model for the hardcore version of this problem where the robot is able to walk over bumps and ditches in the path, but it does seem much harder to do, and I wonder how to best tackle the problem and if similar techniques can be used.

I think that further exploration into some of these parameter tuning and other implementations might yield even better results, however I am happy with the great improvement and results that I have obtained so far! I learned a lot about toolsets like tensorboard and stable-baselines that really help you to understand and evaluate your model.

References

- https://nextgrid.ai/deep-learning-labs/
- https://stackoverflow.com/questions/50107530/how-to-render-openai-gym-in-google-colab
- https://stackoverflow.com/questions/63012499/tensorboard-in-google-colab-for-tensorflow-1-x
- https://stable-baselines.readthedocs.io/en/master/guide/tensorboard.html
- https://medium.com/data-from-the-trenches/choosing-a-deep-reinforcement-learning-library-890fb0307092
- https://araffin.github.io/post/sb3/
- https://gym.openai.com/envs/BipedalWalker-v2/
- https://arxiv.org/pdf/1707.06347.pdf
- https://github.com/araffin/rl-tutorial-jnrr19/blob/sb3/4_callbacks_hyperparameter_tuning.ipynb
- https://stable-baselines3.readthedocs.io/en/master/guide/examples.html
- https://github.com/DLR-RM/rl-baselines3-zoo
- https://opensourcelibs.com/lib/rl-baselines3-zoo
- https://towardsdatascience.com/elegantrl-a-lightweight-and-stable-deep-reinforcement-learning-library-95cef5f3460b
- https://github.com/mayurmadnani/BipedalWalker