# Google Dopamine

New Framework for Flexible and Reproducible Reinforcement Learning Research

## Introduction

Reinforcement learning is an important domain of machine learning. Which mimics the human level learning. RL has gain a lot of momentum over the past few years. More and more researh has been done in RL results in the improvements in the reinforcement learning methodolgies and techniques. Google has announced new Reinforcement learning tensor flow based framework called Dopamine. Which aims to provide flexibility, stability, and reproducibility for new and experienced RL researchers alike. Inspired by one of the main components in reward-motivated behaviour in the brain and reflecting the strong historical connection between neuroscience and reinforcement learning research, this platform aims to enable the kind of speculative research that can drive radical discoveries.

- Ease of use
- Reproducibility
- Benchmarking

## Here we will code our simple agent using google Dopamine.

```
# @title Install necessary packages.
#dopamine for RL
!pip install --upgrade --no-cache-dir dopar
# dopamine dependencies
!pip install cmake
#Arcade Learning Environment
!pip install atari_py
```

### Install necessary packages.

```
Collecting dopamine-rl
  Downloading https://files.pythonhosted.org/packages/a3/60/ce40162119275f8961b79ee1
    100% |████████████████████████████████| 71kB 4.5MB/s
Requirement already satisfied, skipping upgrade: opencv-python>=3.4.1.15 in /usr/loc
Collecting gin-config>=0.1.1 (from dopamine-rl)
  Downloading https://files.pythonhosted.org/packages/e4/07/c8054ce483f058cd8fa2368c
Requirement already satisfied, skipping upgrade: tensorflow in /usr/local/lib/python
Requirement already satisfied, skipping upgrade: absl-py>=0.2.2 in /usr/local/lib/py
Collecting gym>=0.10.5 (from dopamine-rl)
  Downloading https://files.pythonhosted.org/packages/9b/50/ed4a03d2be47ffd043be2ee5
    100% |████████████████████████████████| 1.5MB 16.3MB/s
Requirement already satisfied, skipping upgrade: numpy>=1.11.3 in /usr/local/lib/pyt
Requirement already satisfied, skipping upgrade: six>=1.10.0 in /usr/local/lib/pytho
Requirement already satisfied, skipping upgrade: gast>=0.2.0 in /usr/local/lib/pytho
Requirement already satisfied, skipping upgrade: wheel>=0.26 in /usr/local/lib/pytho
Requirement already satisfied, skipping upgrade: grpcio>=1.8.6 in /usr/local/lib/pyt
Requirement already satisfied, skipping upgrade: protobuf>=3.6.0 in /usr/local/lib/p
Requirement already satisfied, skipping upgrade: termcolor>=1.1.0 in /usr/local/lib/
Requirement already satisfied, skipping upgrade: tensorboard<1.11.0,>=1.10.0 in /usr
Requirement already satisfied, skipping upgrade: astor>=0.6.0 in /usr/local/lib/pyth
Requirement already satisfied, skipping upgrade: setuptools<=39.1.0 in /usr/local/li
Requirement already satisfied, skipping upgrade: requests>=2.0 in /usr/local/lib/pyt
Collecting pyglet>=1.2.0 (from gym>=0.10.5->dopamine-rl)
  Downloading https://files.pythonhosted.org/packages/1c/fc/dad5eaaab68f0c21e2f906a9
    100% |████████████████████████████████| 1.0MB 26.8MB/s
Requirement already satisfied, skipping upgrade: markdown>=2.6.8 in /usr/local/lib/p
Requirement already satisfied, skipping upgrade: werkzeug>=0.11.10 in /usr/local/lib
Requirement already satisfied, skipping upgrade: urllib3<1.23,>=1.21.1 in /usr/local
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in /usr/local/li
Requirement already satisfied, skipping upgrade: chardet<3.1.0,>=3.0.2 in /usr/local
Requirement already satisfied, skipping upgrade: idna<2.7,>=2.5 in /usr/local/lib/py
Requirement already satisfied, skipping upgrade: future in /usr/local/lib/python3.6/
Installing collected packages: gin-config, pyglet, gym, dopamine-rl
  Running setup.py install for gym ... done
```

```python
# @title Necessary imports and globals
import numpy as np
import os
#DQN for baselines
from dopamine.agents.dqn import dqn_agent
from dopamine.atari import run_experiment
from dopamine.colab import utils as colab_
#warnings
from absl import flags

#where to store training logs
BASE_PATH = '/tmp/colab_dope_run'  # @para
#which arcade environment?
GAME = 'Pong'  # @param
```

### Necessary imports and globals

**BASE_PATH:** '/tmp/colab_dope_run'

**GAME:** 'Pong'

```
  Stored in directory: /root/.cache/pip/wheels/ac/79/85/b21b404d3469c3028aea3b/a1dfe
```

```python
# @title Create a  new agent from scratch.

#define where to store log data
LOG_PATH = os.path.join(BASE_PATH, 'basic_

class BasicAgent(object):
  """This agent randomly selects an action
  actions with probability switch_prob."""
  def __init__(self, sess, num_actions, sw
    #tensorflow session
```

### Create a new agent from scratch.

```
      self._sess = sess
      #how many possible actions can it take
      self._num_actions = num_actions
      # probability of switching actions in 
      self._switch_prob = switch_prob
      #initialize the action to take (random
      self._last_action = np.random.randint(
      #not debugging
      self.eval_mode = False

    #How select an action?
    #we define our policy here
    def _choose_action(self):
      if np.random.random() <= self._switch_
        self._last_action = np.random.randin
      return self._last_action

    #when it checkpoints during training, an
    def bundle_and_checkpoint(self, unused_c
      pass

    #loading from checkpoint
    def unbundle(self, unused_checkpoint_dir
                 unused_data):
      pass

    #first action to take
    def begin_episode(self, unused_observati
      return self._choose_action()

    #cleanup
    def end_episode(self, unused_reward):
      pass

    #we can update our policy here
    #using the reward and observation
    #dynamic programming, Q learning, monte 
    def step(self, reward, observation):
      return self._choose_action()

  def create_basic_agent(sess, environment):
    """The Runner class will expect a functi
    return BasicAgent(sess, num_actions=envi
                      switch_prob=0.2)

  # Create the runner class with this agent.
  # to terminate quickly, as this is mostly 
  # use the framework. We also explicitly te
  # of the standard 200) to demonstrate the 
  basic_runner = run_experiment.Runner(LOG_P
                                       crea
                                       game_
                                       num_
                                       trai
                                       eval
                                       max_
```

```
  # @title Train Basic Agent.
  print('Will train basic agent, please be p
  basic_runner.run_experiment()
  print('Done training!')
```

### Train Basic Agent.

```
Will train basic agent, please be patient, may be a while...
INFO:tensorflow:Beginning training...
INFO:tensorflow:Starting iteration 0
INFO:tensorflow:Average undiscounted return per training episode: -2.00
INFO:tensorflow:Average training steps per second: 676.90
INFO:tensorflow:Average undiscounted return per evaluation episode: -2.00
INFO:tensorflow:Starting iteration 1
INFO:tensorflow:Average undiscounted return per training episode: -2.00
INFO:tensorflow:Average training steps per second: 697.19
INFO:tensorflow:Average undiscounted return per evaluation episode: -2.00
INFO:tensorflow:Starting iteration 2
INFO:tensorflow:Average undiscounted return per training episode: -2.00
INFO:tensorflow:Average training steps per second: 704.06
INFO:tensorflow:Average undiscounted return per evaluation episode: -1.00
INFO:tensorflow:Starting iteration 3
INFO:tensorflow:Average undiscounted return per training episode: -2.00
INFO:tensorflow:Average training steps per second: 695.77
INFO:tensorflow:Average undiscounted return per evaluation episode: -2.00
INFO:tensorflow:Starting iteration 4
INFO:tensorflow:Average undiscounted return per training episode: -2.00
INFO:tensorflow:Average training steps per second: 690.75
INFO:tensorflow:Average undiscounted return per evaluation episode: -2.00
INFO:tensorflow:Starting iteration 5
INFO:tensorflow:Average undiscounted return per training episode: -2.00
INFO:tensorflow:Average training steps per second: 700.27
INFO:tensorflow:Average undiscounted return per evaluation episode: -2.00
INFO:tensorflow:Starting iteration 6
INFO:tensorflow:Average undiscounted return per training episode: -1.00
INFO:tensorflow:Average training steps per second: 699.86
INFO:tensorflow:Average undiscounted return per evaluation episode: -1.00
INFO:tensorflow:Starting iteration 7
INFO:tensorflow:Average undiscounted return per training episode: -1.00
INFO:tensorflow:Average training steps per second: 673.64
INFO:tensorflow:Average undiscounted return per evaluation episode: -2.00
INFO:tensorflow:Starting iteration 8
INFO:tensorflow:Average undiscounted return per training episode: -1.00
INFO:tensorflow:Average training steps per second: 663.00
INFO:tensorflow:Average undiscounted return per evaluation episode: -2.00
INFO:tensorflow:Starting iteration 9
INFO:tensorflow:Average undiscounted return per training episode: -2.00
INFO:tensorflow:Average training steps per second: 692.02
INFO:tensorflow:Average undiscounted return per evaluation episode: -1.00
INFO:tensorflow:Starting iteration 10
INFO:tensorflow:Average undiscounted return per training episode: -2.00
INFO:tensorflow:Average training steps per second: 685.70
INFO:tensorflow:Average undiscounted return per evaluation episode: -1.00
INFO:tensorflow:Starting iteration 11
INFO:tensorflow:Average undiscounted return per training episode: 0.00
INFO:tensorflow:Average training steps per second: 683.18
INFO:tensorflow:Average undiscounted return per evaluation episode: -2.00
INFO:tensorflow:Starting iteration 12
INFO:tensorflow:Average undiscounted return per training episode: -1.00
INFO:tensorflow:Average training steps per second: 692.66
INFO:tensorflow:Average undiscounted return per evaluation episode: -2.00
```

```
# @title Load baseline data# @titl
!gsutil -q -m cp -R gs://download-dopamine-rl/preprocessed-benchmarks/* /content/
```

```
experimental_data = colab_utils.load_baselines('/content')
```

```
# @title Load the training logs.
basic_data = colab_utils.read_experiment(l
basic_data['agent'] = 'BasicAgent'
basic_data['run_number'] = 1
experimental_data[GAME] = experimental_dat
```

## Load the training logs.

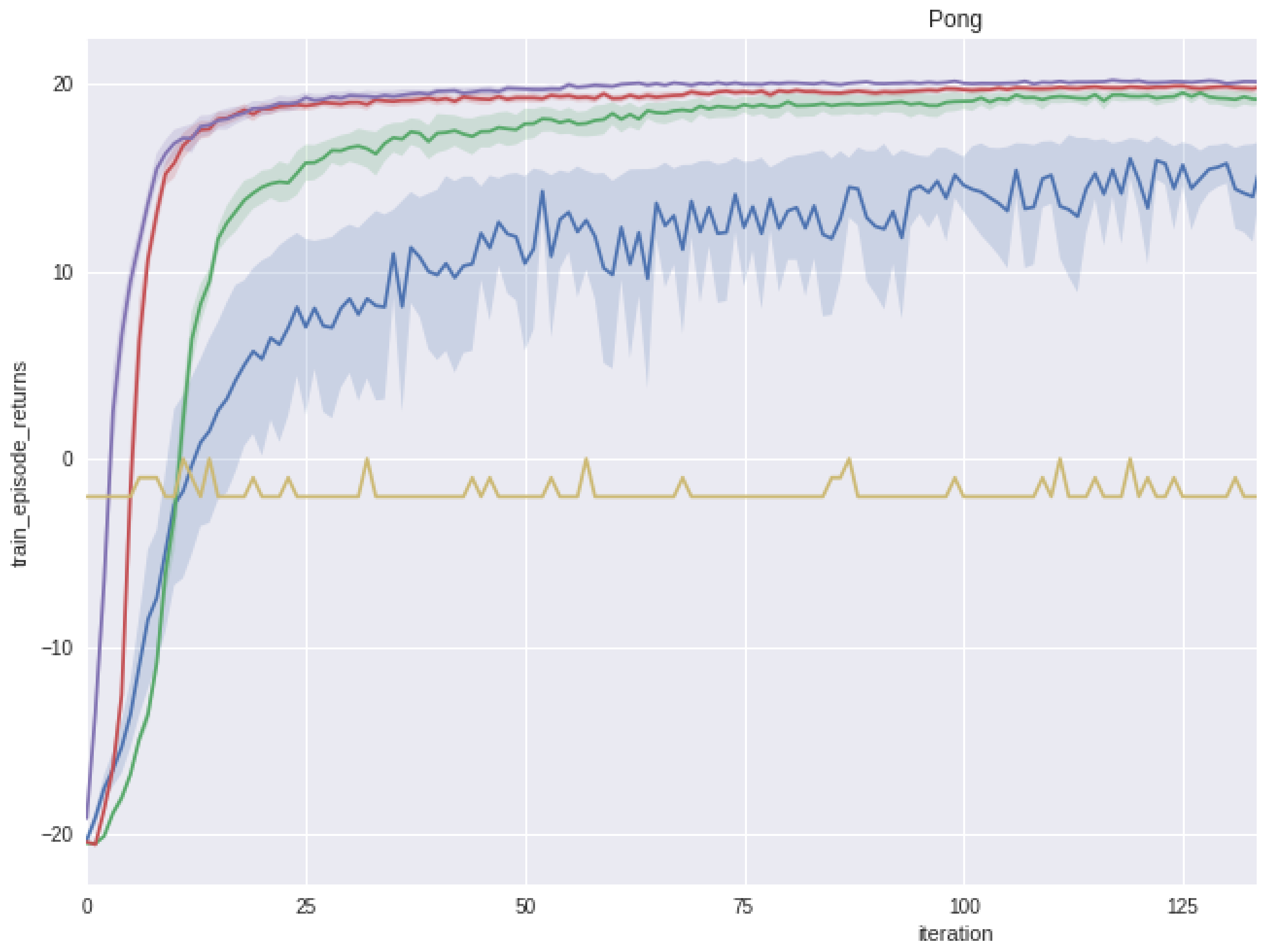
```
Reading statistics from: /tmp/colab_dope_run/basic_agent/Pong//logs/log_199
```

```
# @title Plot training results.

import seaborn as sns
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(16,8))
sns.tsplot(data=experimental_data[GAME], t
           condition='agent', value='train
plt.title(GAME)
plt.show()
```

## Plot training results.



So is the training results of agent which we trained above.

```
You can find this in github too.
Follow AI based punlication on Medium **"The 21st Century"**
https://medium.com/the-21st-century


**Hamza Abdullah **
```

You can find this in github too. Follow AI based punlication on Medium **"The 21st Century"** https://medium.com/the-21st-century

**Hamza Abdullah**