CNNpolicy

CNNLSTM feature extraction

Cloud accounts, GPU

PPO CNN 10lvl

1 mil timesteps: 1.56

PPO CNN 1lvl

1 mil timesteps: 2.47

PPO CNN 50lvl

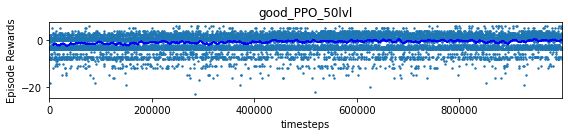
1 mil timesteps: 0.00

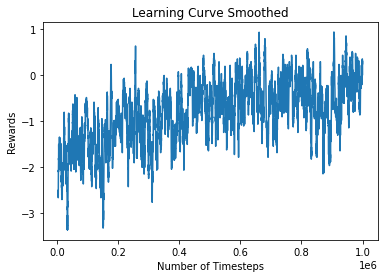
ACER CNN 50lvl

0.7 mil timesteps: 0.54

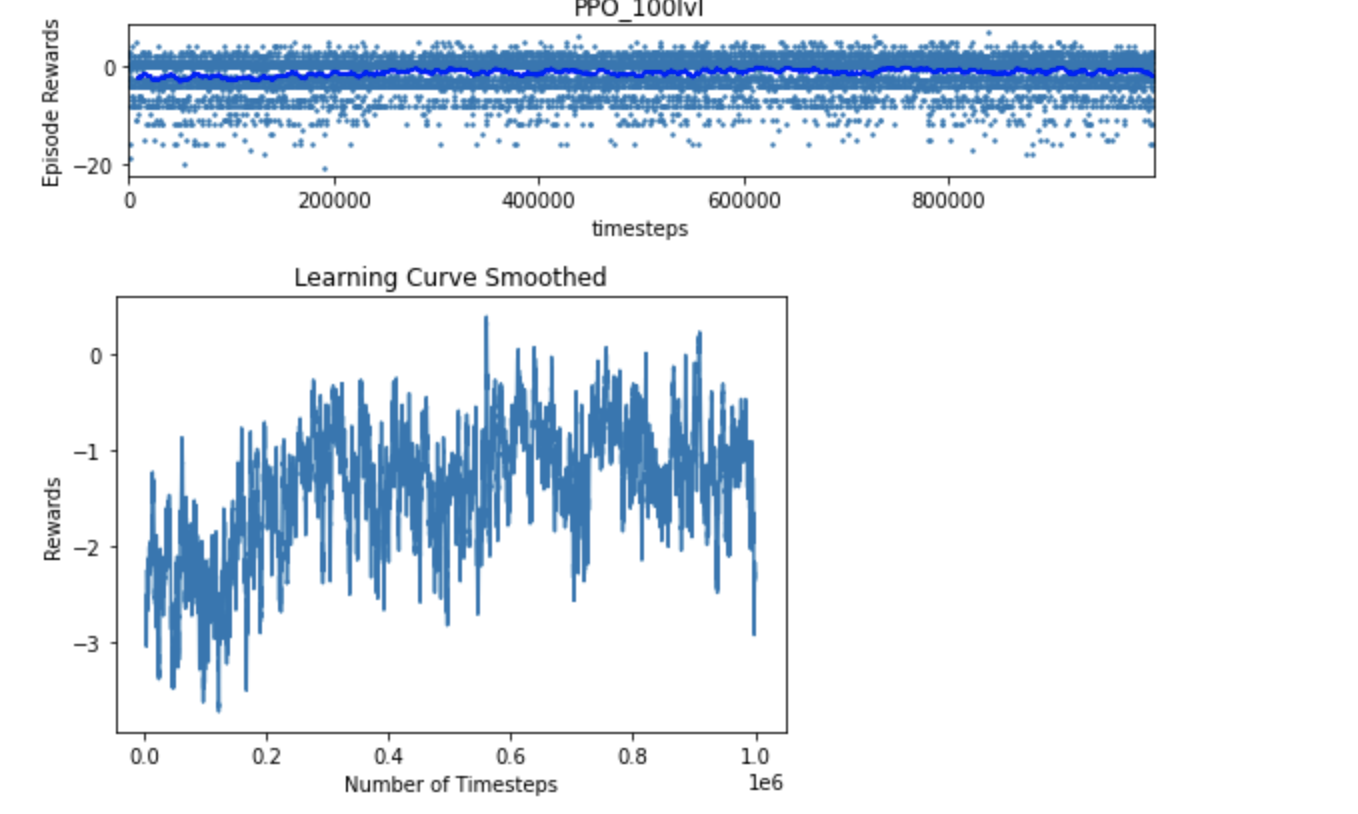
Good ppo

1 mil : 0.37





Good PPO 100 level

1mil: -0.27

Local baseline PPO:

1.62 mil timesteps: 19.4 reward

--------------------------------------

| eplenmean | 364 |

| eprewmean | 19.4 |

| fps | 81 |

| loss/approxkl | 0.0088 |

| loss/clipfrac | 0.11 |

| loss/policy\_entropy | 2.23 |

| loss/policy\_loss | -0.0056 |

| loss/value\_loss | 0.423 |

| misc/explained\_variance | 0.789 |

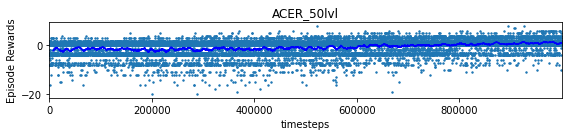
| misc/nupdates | 99 |

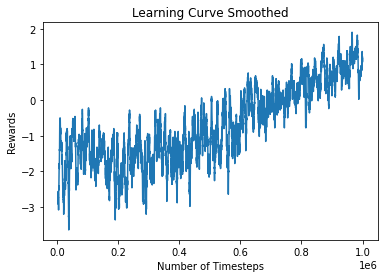
| misc/serial\_timesteps | 2.53e+04 |

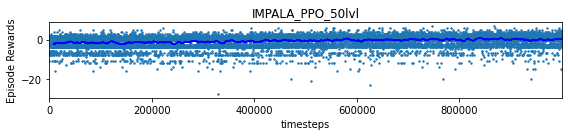
| misc/time\_elapsed | 1.92e+04 |

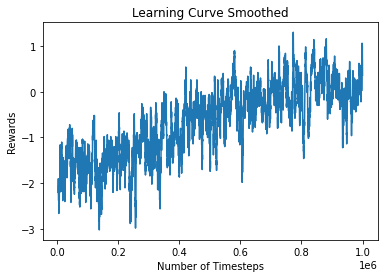
| misc/total\_timesteps | 1.62e+06 |

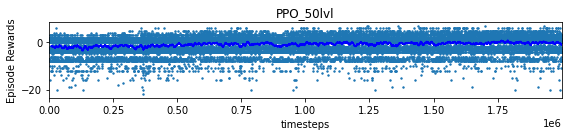
--------------------------------------

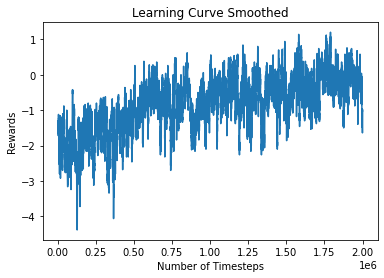


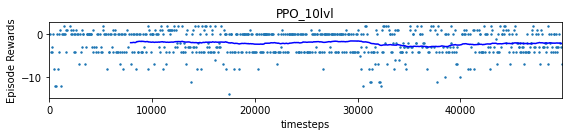


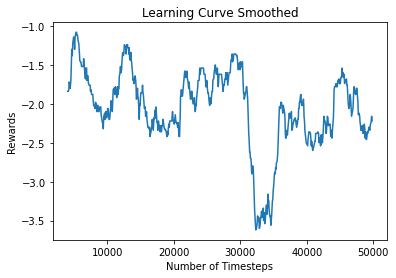












Resnet perception

Pretrain perception, bottleneck autoencoder? Use for many policies

If we can get a small meaningful state, this is a very easy task

Add batchnorm, figure out implementation

VAE <https://github.com/google-research/planet/tree/master/planet/training>

Use good policy

Different policy network?

Baselines <https://stable-baselines.readthedocs.io/en/master/modules/policies.html>

Resnet <https://pytorch.org/hub/pytorch_vision_resnet/>

Could use pretrained net

Transformer <https://pytorch.org/docs/stable/generated/torch.nn.Transformer.html>

Figure out Convolutions with self attention layers

Procgen baseline code <https://github.com/openai/train-procgen/blob/master/train_procgen/train.py>

Model object

<https://github.com/openai/baselines/blob/master/baselines/ppo2/model.py>

Policy <https://github.com/openai/baselines/blob/ea25b9e8b234e6ee1bca43083f8f3cf974143998/baselines/common/policies.py#L121>

Custom CNN <https://github.com/openai/baselines/blob/master/baselines/common/models.py>

Visualizing training <https://colab.research.google.com/drive/1ovlzqHO9Pn79MJopdNPMZ81U8Gw3SKHb>

Monitoring env

<https://stable-baselines.readthedocs.io/en/master/common/monitor.html>

Evaluating

<https://github.com/openai/baselines/blob/master/baselines/ppo2/runner.py>

Stable baselines policies <https://stable-baselines.readthedocs.io/en/master/modules/policies.html>

Representation learning in HRL <https://arxiv.org/pdf/1810.01257.pdf>

Meta learning <https://github.com/learnables/learn2learn>

PPO <https://arxiv.org/abs/1707.06347>

Multimodal distribution: tree example for self driving cars. Currently using categorical distribution for actions so not unimodal, but if we were to represent actions in a meaningful way corresponding to directions then this might be interesting to experiment with.

Could train perception first, Autonencoder, VAE?, then use that for policy models

Can train smaller model, then load bigger model and train the rest.

Perhaps 2 policy models, competing/collaborating. Every other timestep one takes over, or after a run, or until they get a nonzero reward! Batching and multienv is tricky

Best run

|  |  |  |
| --- | --- | --- |
| misc/total\_timesteps | eplenmean | eprewmean |
| 16384 | 85.16 | -1.83 |
| 32768 | 84.27 | -3.1800001 |
| 49152 | 79.8 | -2.48 |
| 65536 | 88.41 | -2.4100001 |
| 81920 | 86.48 | -2.3199999 |
| 98304 | 87.17 | -2.8599999 |
| 114688 | 85.46 | -2.6199999 |
| 131072 | 85.32 | -2.25 |
| 147456 | 80.2 | -2.47 |
| 163840 | 86.77 | -2.3800001 |
| 180224 | 84.4 | -2.23 |
| 196608 | 86.29 | -1.75 |
| 212992 | 82.23 | -1.98 |
| 229376 | 83.45 | -1.5700001 |
| 245760 | 87.3 | -0.33 |
| 262144 | 84.8 | -1.72 |
| 278528 | 80.38 | -1.5700001 |
| 294912 | 86.7 | -0.96 |
| 311296 | 86.78 | -0.74 |
| 327680 | 78.23 | -0.11 |
| 344064 | 80.33 | -0.3 |
| 360448 | 78.29 | 0.02 |
| 376832 | 80.11 | -0.12 |
| 393216 | 82.94 | -0.12 |
| 409600 | 84.17 | 0.19 |
| 425984 | 79.06 | 0.38999999 |
| 442368 | 86.85 | 0.41 |
| 458752 | 79.63 | 0.62 |
| 475136 | 80.57 | 0.70999998 |
| 491520 | 77.54 | 0.56 |
| 507904 | 83.29 | 0.94999999 |
| 524288 | 78.22 | 0.52999997 |
| 540672 | 78.22 | 0.70999998 |
| 557056 | 93.95 | 1.07000005 |
| 573440 | 86.84 | 1.35000002 |
| 589824 | 95.66 | 1.65999997 |
| 606208 | 90.54 | 1.58000004 |
| 622592 | 91.35 | 1.67999995 |
| 638976 | 102.32 | 1.84000003 |
| 655360 | 86.12 | 1.32000005 |
| 671744 | 94.42 | 1.57000005 |
| 688128 | 95.68 | 1.80999994 |
| 704512 | 110.49 | 2.29999995 |
| 720896 | 118.4 | 2.68000007 |
| 737280 | 114.21 | 2.50999999 |
| 753664 | 131.13 | 3.20000005 |
| 770048 | 147.81 | 3.94000006 |
| 786432 | 135 | 3.23000002 |
| 802816 | 134.18 | 3.58999991 |
| 819200 | 148.8 | 4.46000004 |
| 835584 | 175.19 | 6.13000011 |
| 851968 | 172.88 | 5.51999998 |
| 868352 | 186.82 | 5.75 |
| 884736 | 186.57 | 5.90999985 |
| 901120 | 202.25 | 6.71000004 |
| 917504 | 209.59 | 7.34000015 |
| 933888 | 219.82 | 8.10999966 |
| 950272 | 206.12 | 6.78000021 |
| 966656 | 201.81 | 6.71000004 |
| 983040 | 207.57 | 7.76000023 |
| 999424 | 222.55 | 7.36999989 |
| 1015808 | 226.86 | 8.55000019 |
| 1032192 | 252.53 | 10.5299997 |
| 1048576 | 256.73 | 11.3299999 |
| 1064960 | 210.07 | 8.52999973 |
| 1081344 | 234.28 | 10.0600004 |
| 1097728 | 254.35 | 11.4399996 |
| 1114112 | 253.55 | 10.1800003 |
| 1130496 | 261.58 | 11.25 |
| 1146880 | 274.21 | 12.8900003 |
| 1163264 | 281.56 | 13.6899996 |
| 1179648 | 269.64 | 12.9099998 |
| 1196032 | 298.28 | 15.0299997 |
| 1212416 | 304.15 | 15.0100002 |
| 1228800 | 292.16 | 14.8999996 |
| 1245184 | 300.95 | 15.4300003 |
| 1261568 | 304.95 | 14.7700005 |
| 1277952 | 312.1 | 15.3100004 |
| 1294336 | 327.2 | 16.9400005 |
| 1310720 | 307.72 | 14.9200001 |
| 1327104 | 291.6 | 14.1099997 |
| 1343488 | 300.34 | 14.5299997 |
| 1359872 | 294.29 | 14.1899996 |
| 1376256 | 301.34 | 14.6400003 |
| 1392640 | 320.38 | 16.2299995 |
| 1409024 | 330.21 | 16.5699997 |
| 1425408 | 347.52 | 17.3899994 |
| 1441792 | 322.06 | 16.0499992 |
| 1458176 | 315.18 | 15.3000002 |
| 1474560 | 340.2 | 17.5200005 |
| 1490944 | 326.78 | 16.2700005 |
| 1507328 | 303.25 | 14.0600004 |
| 1523712 | 295.42 | 14.4799995 |
| 1540096 | 308.55 | 15.0699997 |
| 1556480 | 309.85 | 14.6800003 |
| 1572864 | 299.03 | 14.3100004 |
| 1589248 | 316.81 | 16.0300007 |
| 1605632 | 348.9 | 19.1299992 |
| 1622016 | 364.16 | 19.3500004 |
| 1638400 | 360.2 | 19.5300007 |