Imitation learning with network pruning in the OpenAI Gym CarRacing environment

Report written by: Andor Diera

25. November 2024

1 Introduction

In this project we used imitation learning to train an agent for solving the OpenAI Gym CarRacing environment. The agent was based on a custom convolutional network which was trained on manually generated samples from the environment. This base model was later compressed with different pruning approaches and the fine-tuned pruned networks were tested in the environment. We consistently achieved comparable results to the base model even when parameter count was reduced by 80%. Our best model achieved an average test reward of 854 (with a standard deviation of 52) over 20 episodes in the environment.

2 Expert data

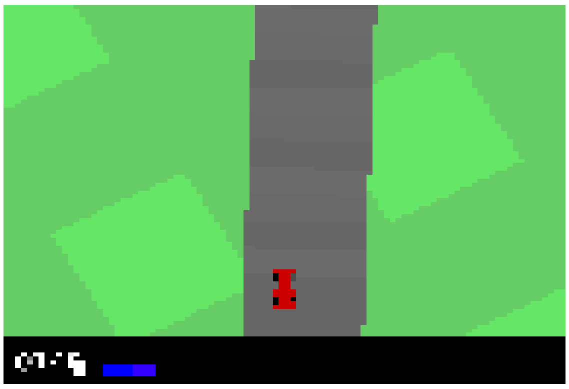
In imitation learning a model is trained to mimic an expert behavior. This behavior was based on the sample data we manually collected from the environment. We used the default action settings of the environment when playing manually. In the end we collected 20 episodes worth of data, consisting of 19054 action-state pairs, with an average test reward of 914,911.

3 Data preprocessing

Before the expert data was used for training, some preprocessing was done on both the state samples and the corresponding actions. We split the data in to a training set and test set, 20% of the data was saved for testing, the remainder was used for training the network

3.1 State preprocessing

Each episode in the CarRacing environment starts with zooming on the racetrack. This effect makes the first 50 frames to have a different zoom level than the rest of the sample. In order to make the expert data more homogeneous we discard this initial 50 frames from each episode. The samples that remained underwent image preprocessing to reduce the number of features for the network. In the first step we recolored the grass and the road to two distinct uniform colors. Although the recoloring of the road results in loosing information of the visited - not visited feature of the parcels shown on the road, we decided it was an acceptable loss, since our network was based on imitation and had no notion of reward during the race. We also removed most information from the indicator bar on the bottom, leaving only the acceleration and brake indicator in the images. In the end we converted the images to grayscale.


(a) Before preprocessing

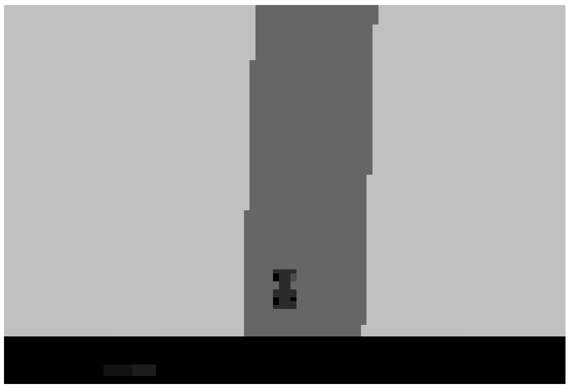

(b) After preprocessing

Abbildung 1: Image preprocessing

3.2 Action preprocessing

The expert data contained 9 action labels: 5 base actions (straight, accelerate, brake, turn left, turn right) and 4 combinations of turning with accelerate or brake. These 4 complex actions were rare in the sample, and were discarded. Since the recorded data was highly unbalanced, the remaining actions underwent some balancing: 70% of the straight and 50% of the acceleration actions were discarded, brake events and right turns were multiplied. Without balancing the network was biased to minimize loss by always choosing the more dominant actions in the data set.

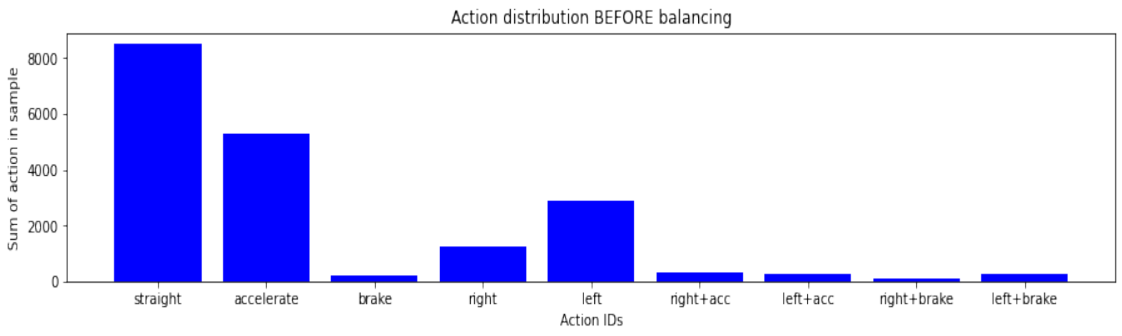


Abbildung 2: Before balancing

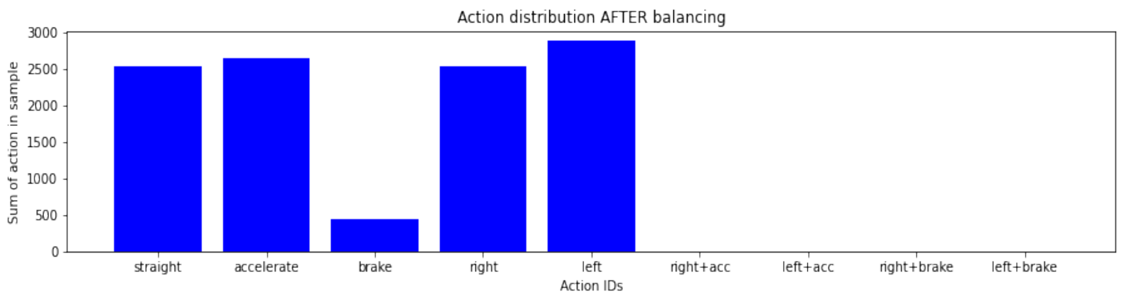


Abbildung 3: After balancing

4 The agent

The agent's behavior in the environment was based on a convolutional neural network, with some changes in speed and an override logic in some specific cases.

4.1 Neural networks

During our project we used two different custom networks: a smaller one with 3 convolutional layers and 2 fully connected layers, and a larger network with 3 convolutional and 3 fully connected layers. Both networks used Adam with a learning rate of 0.0001 as the optimizer for minimizing cross-entropy loss. The large network had more than 5 times the parameter count of the small network.

Small network				
#	Input channels	Output channels	Kernels	Strides
conv1	1	8	5	1
conv2	8	16	3	1
conv3	16	32	3	3
fc1	32*3*3	128		
fc2	128	5		

Large network				
#	Input channels	Output channels	Kernels	Strides
conv1	1	16	5	1
conv2	16	32	4	1
conv3	32	64	3	3
fc1	64*3*3	256		
fc2	256	128		
fc3	128	5		

4.2 Override logic

Using only the output of the neural network in the environment led to unreliable behavior. The agent rarely used the brake and was prone to slide out of the road when taking turns. To mitigate this behavior we reduced the agent's speed by 70% compared to the settings used for recording the expert data. Another failure of the network was observable in a few special cases: when the vehicle approached sharp turns, sometimes it happened that the agent stopped before the turn, and remained frozen until the end of the episode. To counteract this behavior we introduced an anti-freeze mechanism. When the agent is stuck in the same state for more than 50 frames, an override is called upon, which selects accelerate as the next five actions.

5 Testing

The networks were trained for 5000 epochs. After that, the models were tested both on the test set and in the environment.

5.1 Test set

Testing the models on the test set was not very useful in predicting the performance in the environment. Most of our models reached an accuracy of 55-65%. The classification reports showed us, that 3 out of the 5 actions were hard to tell apart. Actions straight, accelerate and brake consistently achieved an F1 score of less than 0.5. When balancing was not used in the preprocessing, the overall accuracy was higher, but it came at a cost of near 0 F1 score of the brake action.

Classification report for the large network after training			
Action id	Precision	Recall	F1 score
straight	0.429	0.382	0.404
accelerate	0.407	0.581	0.479
brake	0.732	0.300	0.426
right	0.857	0.613	0.714
left	0.700	0.804	0.748

5.2 Environment performance

The testing in the environment consisted of 20 episodes with a 5000 frame limit each episode. The average episode reward for the small network was 664 with a standard deviation of 118 and a max reward of 819. The large network achieved an average reward of 778 with a standard deviation of 110 and a max reward of 911.

6 Network pruning

In order to compress the models and observe the effects of compression on performance, network pruning was implemented. We implemented 4 slightly different pruning approaches, which were all based on weight-magnitude based pruning, where a given amount of the smallest weights were set to 0 during the pruning process. To achieve this, L1 regularization was adopted to the weights.

The first approach consisted of applying a global one-shot unstructured pruning to all layers at the same time. This meant that the layers were not symmetrically pruned, after pruning all layers had a different level of sparsity.

In the second approach the same global one-shot unstructured pruning was used, but first it was applied on the convolutional layers, and then separately to the fully connected layers.

The third approached used global pruning for the fully connected layers, but implemented a layer-wise unstructured pruning for the convolutional layers. This resulted in all the convolutional layers having the same amount of sparsity after pruning.

The last approach used was structured pruning. In it we applied global pruning to the fully connected layers once again, but on the convolutional layers a layer-wise structured approach was used. This meant that instead of removing individual weights, whole output channels were removed in the convolutional layers.

6.1 Pruning results

Testing the pruned models in the environment without fine-tuning resulted in significant performance loss. In some cases the model was completely faulty, and achieved an average reward of less than zero. This performance loss was solved with fine-tuning. After pruning, we trained the models for another 500 epoch with half of the original rate. In the end we compared 4 sparsity levels with the 4 different approaches applied to the large network.

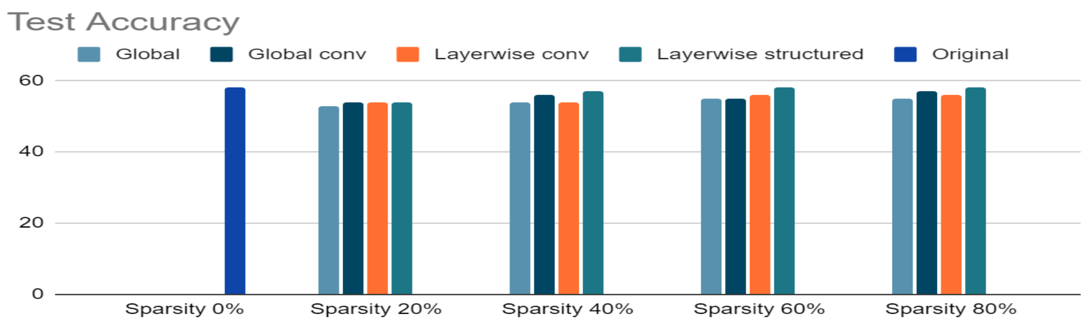


Abbildung 4: Test accuracy comparison

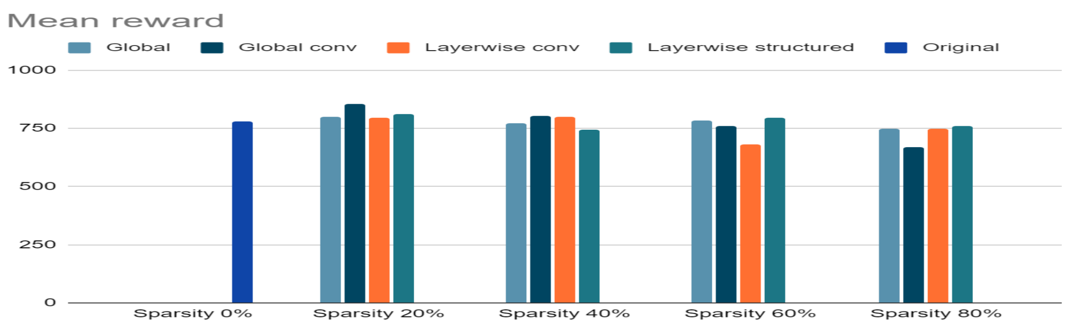


Abbildung 5: Mean reward comparison

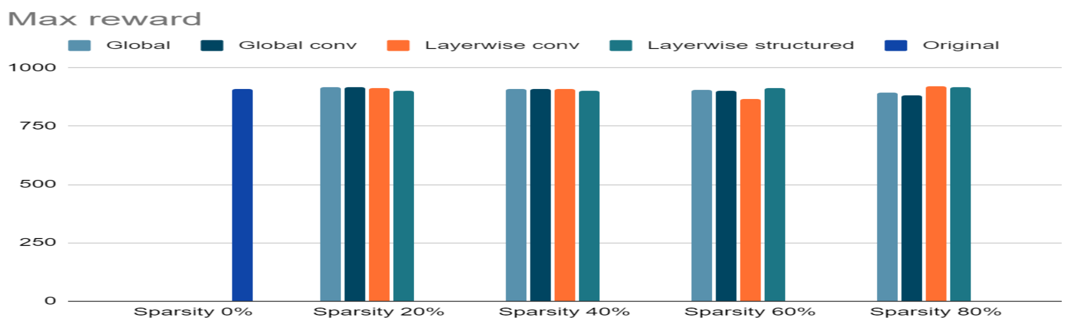


Abbildung 6: Max reward comparison

As can bee seen from plots, the results are comparable across all approaches. Although a slight decrease in the environment performance can be seen as the sparsity increases, its hard to extract any meaningful difference between the pruning approaches. The first pruning approach(global pruning with all layers) seems to be have the least amount of changes between sparsity levels, but even the others show only a slight variance. In the end our best performance was achieved with the second pruning approach at 20% sparsity level.

7 Discussion

7.1 General results

As mentioned before we could reliably achieve an average episode reward between 700 and 800, with the best result at 854. Imitation learning based on discrete actions is not the best method suited for solving this environment, thus we consider our results satisfactory. With applying the most common pruning methods to our models, we were able to keep and even increase our base performance in some cases. This reinforces the general observations in the literature, that pruning is indeed capable of retaining inference performance even when parameter count is drastically reduced. In the end comparing the different approaches to weight-magnitude based pruning did not show significant differences in performance

7.2 Adversarial attacks

During this project I collaborated with a lab-partner, whose main focus was adversarial attacks, and training with adversarial samples. When comparing performances, we found that the large network at 80% sparsity was more robust to adversarial attacks than the unpruned small network, even though the parameter count was the same. Although there was some variance in performance between the pruning methods when adversarial attack was applied to them, this part of the research remains inconclusive.

7.3 Code base

Code available at https://github.com/drndr/selfdriving_project