Imitation Learning

Done by:

Tarek Abou Chakra: 5450275

Submitted for:

Deep Learning Lab

Exercise 1

Abstract:

This report aims to provide a comprehensive overview of imitation learning. Mainly focusing on the application of imitation learning using neural networks and the Python programming language. Additionally, the report will examine how various parameters can affect the final result of these networks

1- Introduction:

Imitation learning has been successfully applied in various domains such as robotics, gaming, and autonomous vehicles. In this report, we explore the use of imitation learning for controlling a race car in the OpenAI Gym environment. We use a deep neural network to learn a policy that mimics the behavior of an expert driver. We evaluate the performance of the imitation learning algorithm on different race tracks (15 episodes) and we compare different history length and architecture of the network.

2- Hyperparameters:

First I would like to begin discussing Two different architectures that were used in this study, each with its own strengths and weaknesses:

1- The first architecture utilized a purely convolutional neural network (CNN) approach. This network consisted of four convolutional layers, each with a kernel size of 3 and padding of 1 (same convolution). A max pooling layer (2,2) and a ReLU activation function. The output of the network was flattened and fed into four different linear units, followed by a dropout layer with a probability of 0.2 to prevent overfitting, and a ReLU activation function (except for the last layer).
2- The second architecture combined a CNN with a long short-term memory (LSTM) layer. Since CNNs are effective at recognizing spatial patterns, and LSTMs excel at capturing temporal dependencies, this architecture aimed to leverage both strengths. Specifically, the model was trained on a history of 1 to observe the difference that the LSTM layer could make in comparison to the other models.

The hyperparameters used in this project included cross-entropy loss (LogSoftmax + NLLLoss combined into one function from Pytorch), which eliminated the need for a softmax function after the last linear layer. Adam was used as the optimizer, with a learning rate of 0.001.

3- Challenges and Solutions:

During the lab, I faced several challenges that required creative solutions to overcome:

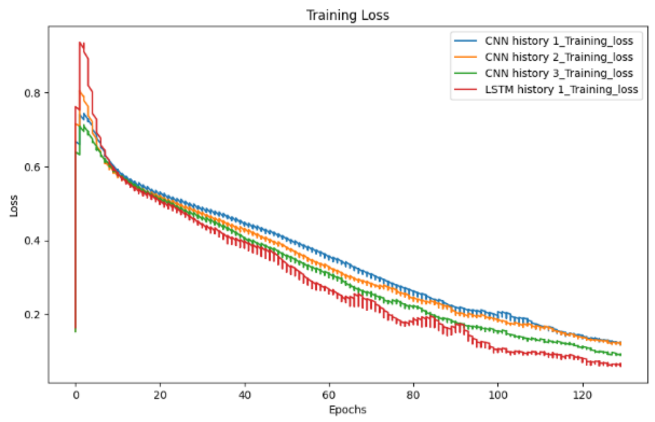
1- One of the major challenges was adding a history aspect to the first images. To address this, I created a dummy image that was an all-zero tensor and concatenated it based on the desired history length, then popped and stacked the tensors to ensure the correct sequence was maintained.
2- Another challenge was the high dependency on expert data. The model's performance was highly dependent on the track environment used during data capturing. If the environment had turns that were not encountered during data capturing, the model would struggle during testing and deliver suboptimal results.

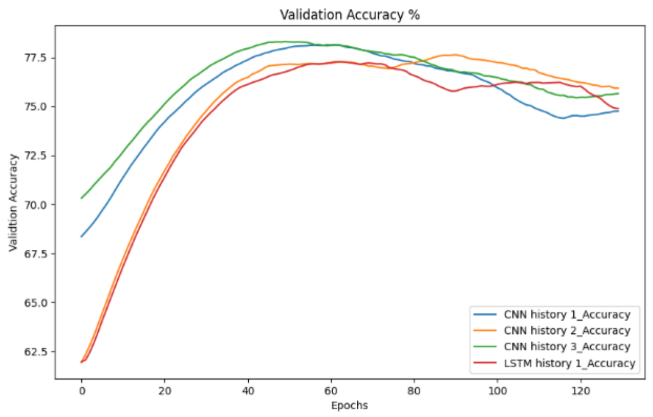
3- One significant issue I faced was losing momentum. Although the model was able to navigate turns accurately, it struggled to maintain momentum after consecutive turns. This was due to the inability to press the gas and turn at the same time during data capturing.

4- The validation accuracy was not as high as expected. This is not surprising, given that the expert makes different decisions, even for the same corner. As a result, the model was unable to achieve a high level of validation accuracy.

4- <u>Graphs and results:</u>

Figures of the training losses and validation accuracies:





A table showing the test result over 15 episodes of different networks:

Model	Highest value	Standard deviation	Mean
CNN history 1	905.39	217.35	659.03
CNN history 2	908.09	202.75	767.82
CNN history 3	888.36	160.36	617.55
LSTM history 1	917.79	95.20	852.62

5- <u>Conclusion:</u>

In conclusion, this project aimed to apply imitation learning using neural networks to control a car in a simulated environment. All models were able to represent the data and achieve high results in some tracks. However, the standard deviation showed the inconsistency of the models across the 15 different episodes. As the history length was increased, the models became more cautious and tended to rarely skip turns but also lose momentum in response to the fear of accelerating. The LSTM network, however, was able to make good decisions about the turns without much sacrifice of momentum and trained faster than the CNN model with history 3.