Behavioral Cloning for Lunar Lander

Naya Baslan, Megan Klaiber, Ahmed Krimi, Abdelrahman Younes

Project Description

Lunar Lander is an OpenAIGym Environment whose goal is to navigate a space-ship and make it land between the two yellow flags with up right position and minimum landing velocity. As shown in the figure, there are four discrete action classes: upwards, left, right and no-action.

In this project, we approach this task through imitation learning, where a human plays the game and feeds the training data so that the agent can learn from it, thereby reformulating the problem as a supervised learning problem.

Obtaining the Training Data

- This was done by collecting data from a human playing the lunar lander game
- Total samples collected: 27500

Training the Behavioral Cloning Agent

- Experiments were performed by implementing a fully connected network (FCN) and a convolutional neural network (CNN).
- FCN takes the state (e.g. angle, velocity) as input whereas the CNN learns directly from images.

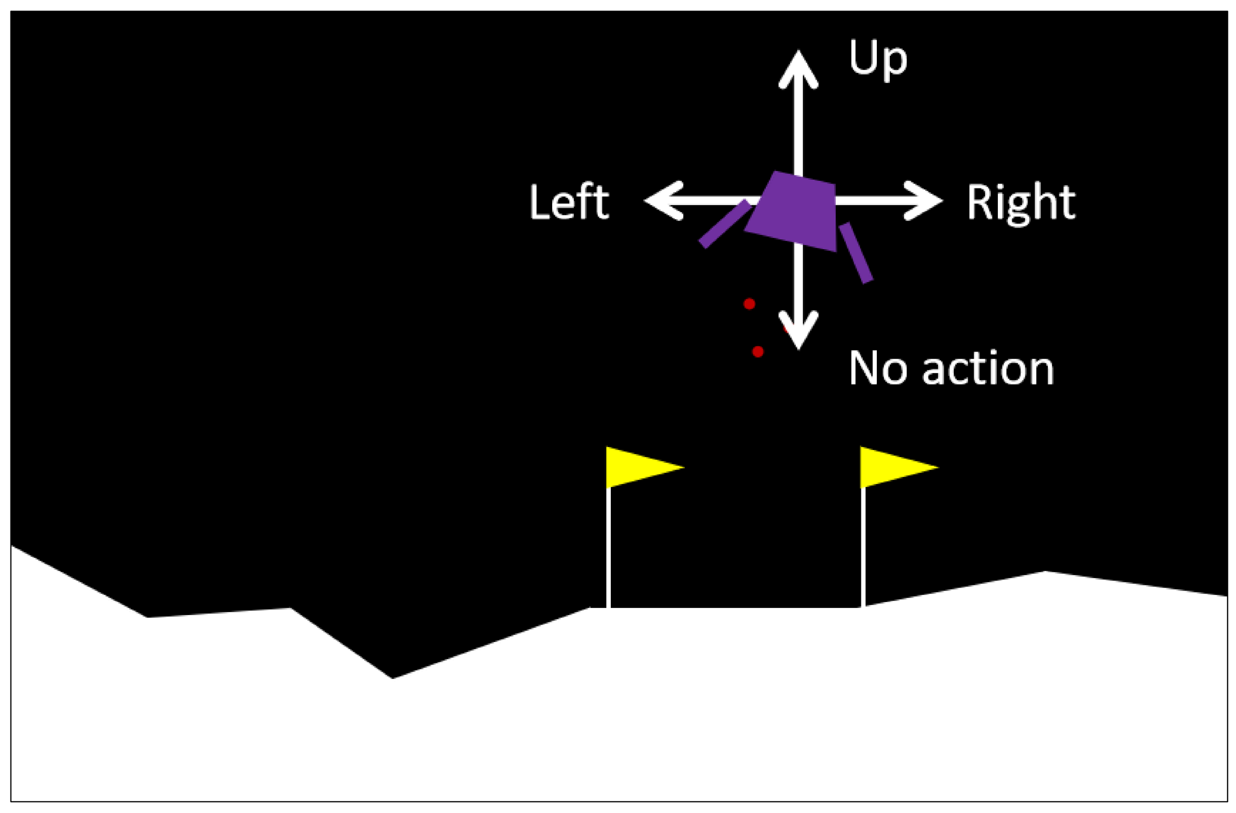

Figure 1: Lunar Lander Environment

Fully Connected Neural Network

Network architecture:

- Input Layer: 8 states

[x_pos, y_pos, x_vel, y_vel, angle, ang_vel, left_leg_ground, right_leg_ground]

- 3 fully connected layers with dropout (0.5) and activation functions ReLU, cross-entropy loss
- Output: one of four action classes (right, left, up, no action)
- Optimizer: Adam with learning rate = 0.001

Table 1: Test result over 15 test episodes

	mean	std
FCN	26.83	22.34

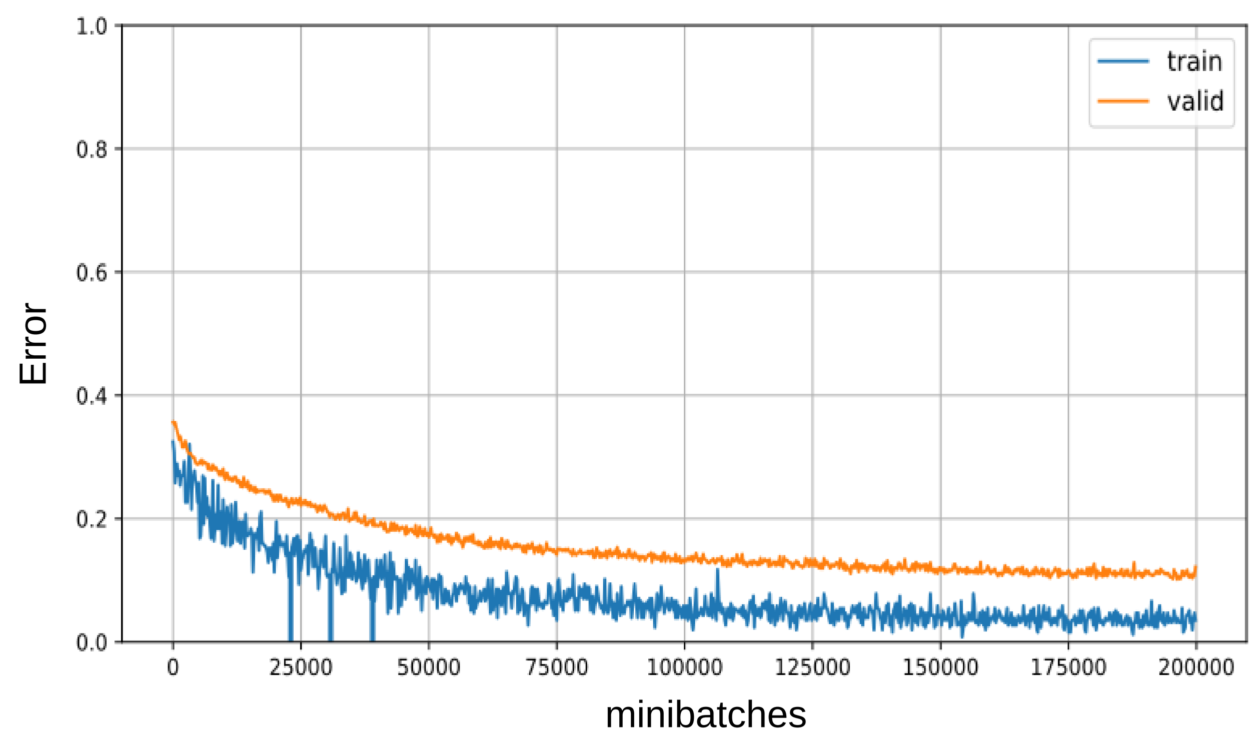

Figure 2: Training and Validation of FCN

Convolutional Neural Network (5 different architectures)

1. First approach: CNN with 2 convolutional layers

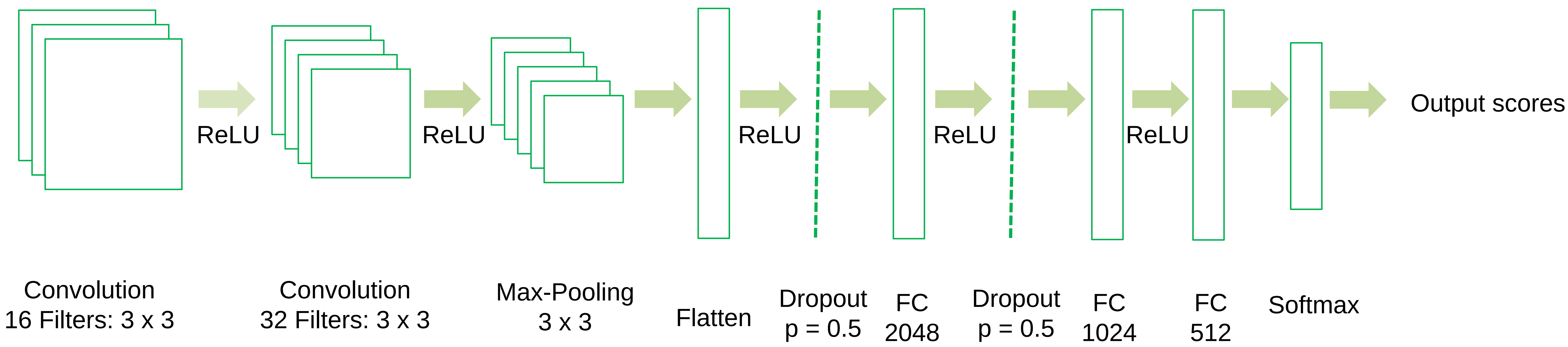

Figure 3: CNN Architecture

Other hyperparameter settings:

<u>Learning Rate</u> = 0.001, <u>Batch Size</u> = 64, <u>Optimizer</u> = Adam, <u>Activation Function</u> = ReLU, #<u>Minibatches</u>: 1400, <u>Stride</u> = 1, <u>Loss</u> = Cross Entropy Loss

Behavioral Cloning for Lunar Lander

2. Second approach: CNN with 5 convolutional layers + uniform sampling

The collected data is highly imbalanced because the no-action class occurs more frequently than the others. Uniform sampling was implemented here to equally sample between the different action classes.

Network Architecture Description:

5 convolutional layers were used with respective filter sizes = [7 x 7], [5 x 5], [5 x 5], [3 x 3], [3 x 3], followed by 3 fully-connected layers.

Other Hyperparameter Settings:

Learning Rate = 0.001, Batch Size = 128, Optimizer = Adam, Activation Functions = ReLU,
#Minibatches: 1500, Stride = 1, Dropout = 0.8, Loss = Cross Entropy Loss, Number of Filters = 32

3. Third Approach: Modified ZFNet: (Different Configurations - simplified to 6 layers)

Different Sampling Methods were used here:

- Equal Sampling, No-action limiter to one fourth and one half of the training set size

Other Hyperparameter Settings:

Learning Rate = 0.001, Batch Size = 128, Optimizer =Adam

Activation Functions = ReLU, #Minibatches: 1500, Stride = 1, Dropout = 0.5, Loss = Cross Entropy Loss, Kernel size = [7 x 7], [3 x 3]

4. Fourth Approach: Modified ResNet18:

Network Architecture Description:

Similar to ResNet18 architecture; remove average pooling, and reduce the number of hidden layers to 4

Other Hyperparameter Settings:

Learning Rate = 0.0005, Batch Size = 16, Optimizer = Adam, Activation Functions = ReLU,
#Minibatches: 1200, Stride = 1, Loss = Cross Entropy Loss

Experimental Results for the Convolutional Neural Networks

Table 2: Mean performance and standard deviation over 15 test episodes

CNN 2 convs	mean	std
history length 0	12.76	26.88
history length 2	28.33	24.38
history length 4	4.56	28.10

CNN 5 convs	mean	std
history length 0	14.34	20.80
history length 1	-11.34	2.21
history length 3	8.14	22.35
history length 5	8.70	18.50

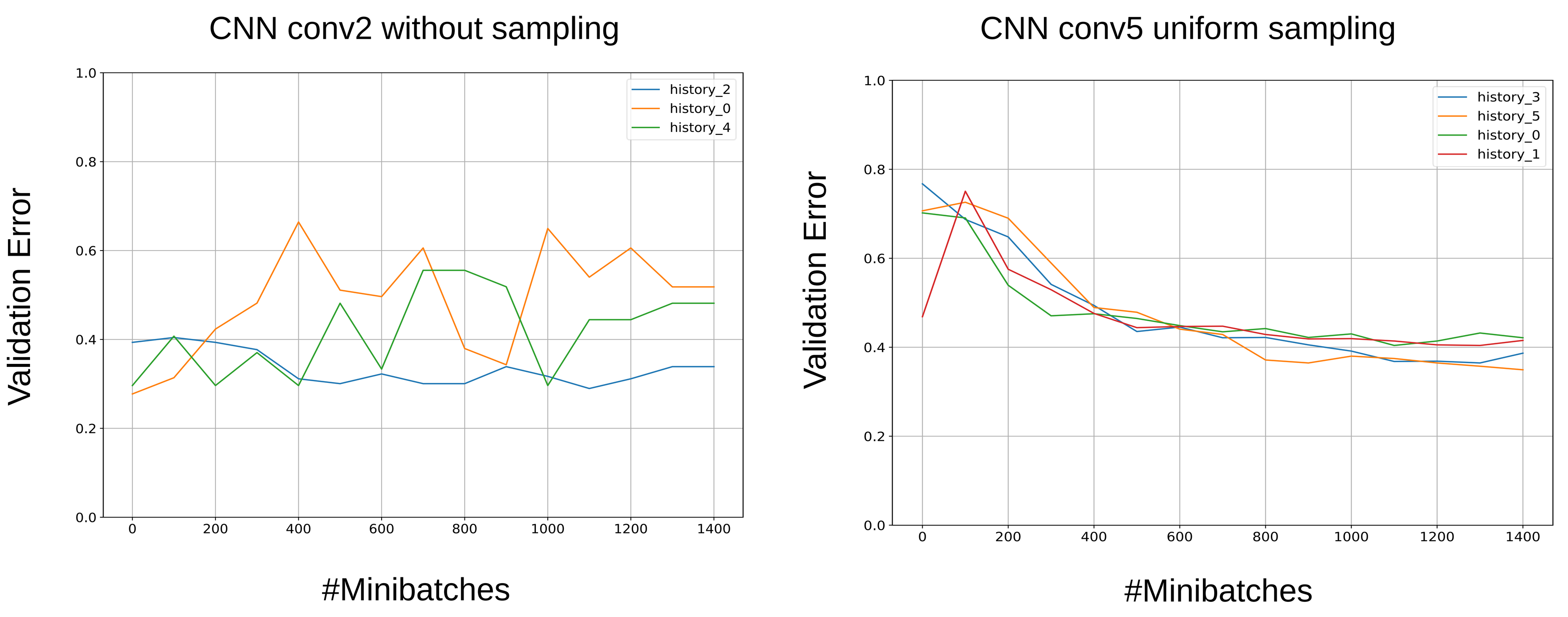

Figure 4: Validation of two CNN Architectures using History

Conclusion

- FCN performs quite well in testing but requires more time to train
- CNN has the best performance for history N = 2
- LSTMs were also implemented and seem to have preliminary promising results
- Attached video shows the performance of the different agents

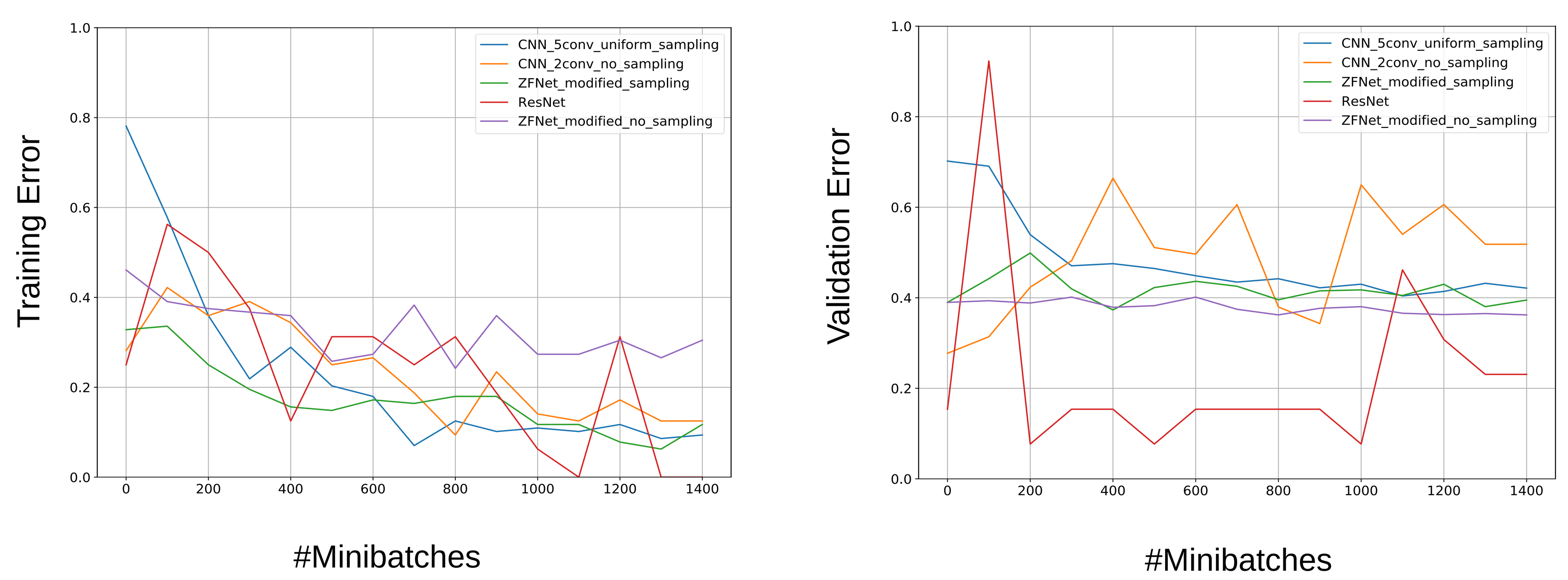

Figure 5: Training and Validation of all five CNN Architectures without history