Predicting Delta States from Images with CNN

Mino Nakura, Nishant Elkunchwar, Pratik Gyawali

May 2020

1 Introduction

Convolutional Neural Networks can be used for a variety of tasks, including optical flow estimation [1], image segmentation, and object classification. In this project, we present 4 convolutional neural networks capable of solving the inverse dynamics problem. These neural networks can be used to predict the delta states of a cartpole trajectory at a time step t, given a sequence of images preceding that time step and an image following the time step. Figure 1 gives a visual representation of an example input (five images) and the output (delta states) of the convolutional neural network. We present several neural network architectures to perform a comparative study of how different hyperparameter choices can affect performance. [1] [2]

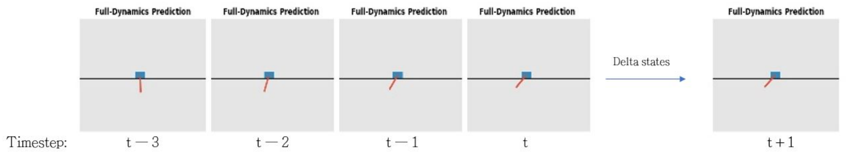


Figure 1: Input: A Sequence of Five Images. Output: Delta States

2 System Overview

2.1 Neural Network Architectures

We explored several different neural networks in our project. Below, we summarize the similarities between each Convolutional Neural Network and also provide a table to highlight the different parameter choices we made. All Neural Networks take in a sequence of 128x128 grayscale images as input. We designed our networks to use ReLU activation functions, use the Adam Optimizer, and rely on mini-batch gradient descent with batches of 128 training examples. After each convolution, we max pool over the images. After all convolutional layers are finished, we sum up along the image dimension and flatten the features into one tensor. The tensor is then passed through a number of fully connected layers. The errors were calculated using Mean-Squared Error. Table 2.1 shows the differences of the network architectures.

Architecture Comparison				
	CartpoleNetLite	CartpoleNetHeavy	CartFourNet	CartThreeNet
Conv3D Layers	3	4	4	4
Fully Connected Layers	3	5	5	5
num. of input images	5	5	4	3
num. training iterations	25k	10k	10k	10k

[1] Github
[2] Youtube

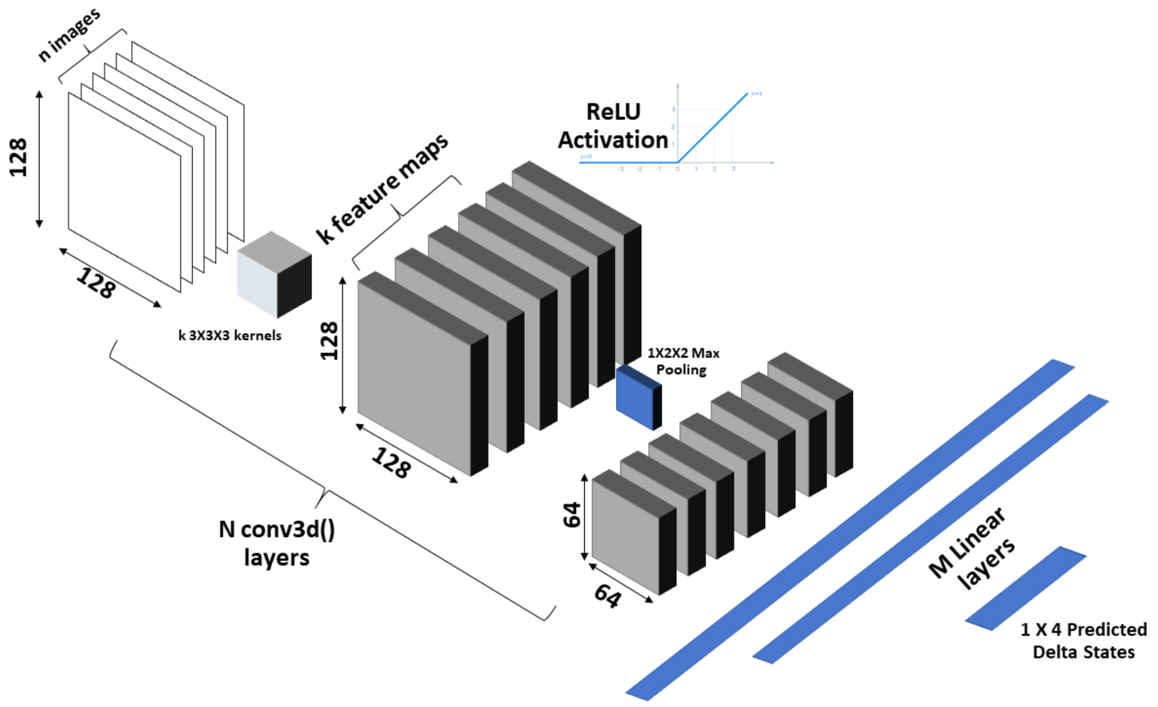


Figure 2: General Architecture Structure

2.2 The Dataset

In our project, we use image dataset to train our networks. The image dataset contains 220 epochs worth of data. Each epoch contains 50 training images of size 128x128. The data.csv file contains the file path to each image, and the delta states associated with each image. Since we use 5 images for one training example, each epoch produces 46 training examples. In total, the image dataset has 10,120 training examples.

2.3 The Dataloader

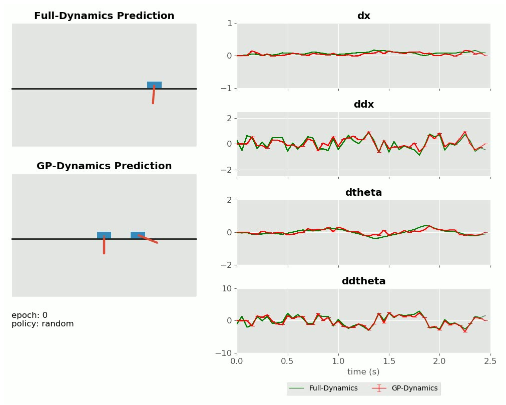
We created a dataloader class which was a iterator that returned the required training inputs and labels. For a given index i from the training data, the dataloader returned a tuple containing the inputs and labels from time steps $(i - n + 1)$ to $(i + 1)$ for n images. The input was a 5d torch tensor with dimensions $(N \times C \times D \times H \times W)$ where N corresponds to the number of training datapoints in a batch, C corresponds to the number of channels in the image (1 for our case since it was grayscale), D corresponds to the number of images in one datapoint (we tested for 5, 4 and 3) and W, H were the image dimensions (128x128 for our case). The label was an $(n \times 4)$ torch tensor containing the delta states for the same time steps. Using a dataloader made it easier for us to send our dataset as input to our CNNs.

3 Results

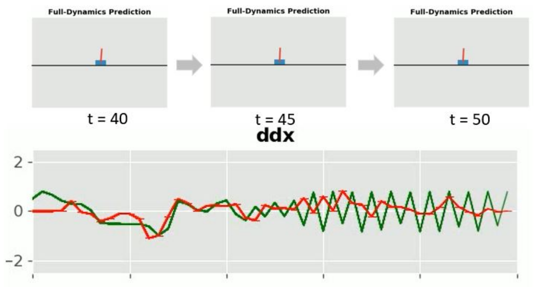
Here, we present the findings of each of our neural networks. We present a quantitative comparison of the four neural networks, as well as a visual comparison of the predictions with their ground truth values. To measure the accuracy of our neural networks, we compared their accuracy on the training data with a new set of test data. Our test data is a data set of around 100 data points. Our findings are summarized in the table below.

Similarly, we included videos [3] of our neural network performances. In the videos, we can see that deeper neural networks are able to predict the delta states with greater accuracy. Similarly, neural networks that take in less images cannot predict the delta states as well. Specifically, we note that even though the

[3]Videos of Cartpole Predictions



(a) CartpoleNetLite Predicting the Delta States



(b) Cartpole stands still from t=40 to t=50

Figure 3: Limitations of our CNNs

theoretical minimum limit for the number of images to predict second derivatives upto the first order is 3, in practice, CartThreeNet does not perform very well for ddx and ddtheta (though, it tracks dx and dtheta well enough).

Mean Squared Error Loss on Test Data

	CartpoleNetLite	CartpoleNetHeavy	CartFourNet	CartThreeNet
Test Data Error	0.06464	0.00531	0.00855	0.10573

In additions to the above findings, we see that the CNNs perform well at predicting the delta states (Figure 3a). However, the CNNs have several limitations. First, they cannot predict delta states until the fourth time step because they require previous images to make the predictions. Similarly, they also cannot predict the delta state at the last time frame because they also requires an ”after” image to make a prediction.

Third, it does not predict well when the pole is not moving. As shown in figure 3b, when the cartpole does not move, the ground truth ddx values show rapid oscillation between positive and negative values. However, the CNN predicts that that the ddx values are near 0 because the cartpole does not show any movement. Since the pole does not show any movement, it is difficult to determine what the ddx values are in these circumstances. Another reason for the poor performance may lie in the fact that the training data has less examples of the swing-up policy.

4 Conclusion

In this report, we present several neural networks to predict the delta states of a cartpole trajectory. We varied architecture choices and input parameters for a Convolutional Neural Network predicting delta states. We also gave an overview of the dataloader we used to help with the training process. Then, we presented a summary of the performance of our neural networks, and show the trade-offs of the different architecture choices. Finally, we presented the limitations of the convolutional neural networks.

References

[1] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks, 2015.