Pouring Dynamics Estimation using Recurrent Neural Network

Madhu Sowmya Bandi

***Abstract*— Pouring is one of the routine tasks that humans do. It can be cooking, planting, lawn watering, e.tc. Robots need to be able to perform such actions to do the needful. Among many actions that Robots do, Pouring requires lot of effort while cooking. The liquid estimation requires lot of efforts to understand how much amount to be added. The estimation can be made by building the Recurrent Neural Network based on the parameters that affect the poring while cooking. In this experiment different neural network architectures are tested for prediction with the given data obtained from the sequence of robotic actions.**

***Keywords*— Recurrent Neural Networks, Poring Dynamics, Cooking Object state classification.**

I. INTRODUCTION

To be useful for our daily living tasks, robots need to be able to perform many different manipulations. Among them, pouring is the second most frequently executed motion (after pick-and-place) in cooking scenarios. Programming or controlling a robot to perform pouring water, we would usually need to model the water's dynamics so that the program or controller can predict the water's behavior. In pouring, it is how much and how fast water comes out of a pouring container at a certain rotation angle and rotation velocity. Robotic pouring became a popular research topic recently. [3] uses a deep neural network to estimate liquid volume in a cup and a pouring PID controller, while [4] estimates liquid height from an RGB-D point cloud and apply a PID controller to control the pouring cup. [5] learns the pouring policy in simulation using reinforcement learning and tested the policy with an actual robot. Like many other daily-living manipulation tasks, it is very difficult to build an accurate pouring dynamics model since it is difficult to model liquid dynamics. However, humans can pour water regularly and accurately without explicitly modeling pouring dynamics. Humans learned to pour at their very early age by observing other people first and then practicing pouring along with other manipulation skills frequently. Therefore, we have developed a robot learning approach that learns pouring skills from human's demonstration [6]. The human demonstration data is from an openly available daily interactive manipulation dataset [7]. A comprehensive review of object manipulation datasets could be found at [8]. The manipulation learning uses Recurrent neural networks (RNN) that taken input and a hidden state from the last time step and produce an output for that time step. The mechanism of RNN makes it inherently suitable for learning the behavior of a dynamic system [9, 10]. RNN has been very successful in processing languages and handwriting. For example, [11] generates English writing trajectories by predicting pen tip locations. [12] applies a similar strategy to generate Chinese characters. To perform simulation, the water behavior should be modeled while pouring so that the simulation system could produce an accurate simulation of pouring outcomes for various pouring velocities, pouring amounts, and different pouring containers. This project is mainly to design an RNN that can learn water behavior in pouring so that it could be used to predict the water behavior. The predicted water behavior could be used in a model-predicted control framework to generate accurate robotic pouring [13].

II. DATA PRE-PROCESSING

A. Dataset

The working dataset is a .npy file which consists of 688 motion sequences and their metrics. The shape of the data is (688,700,7) that is (number of sequences, maximum length of a sequence, feature dimensions). Maximum length gives the information about the maximum a sequence can be. Since each sequence in the given data has varying length. If the length of the sequence is than 700 that sequence length is made 700 by padding zeroes. Any zeroes in the sequences are not the actual data collected. The parameters in the motion sequence describe as following.

θ(t) - rotation angle at time t (degree)

f(t) - weight at time t (lbf)

finit weight before pouring (lbf)

ftarget - weight aimed to be poured in the receiving cup (lbf)

hcup - height of the pouring cup (mm)

dcup - diameter of the pouring cup (mm)

θ(t) - velocity of the pouring cup (rad/s)

All the features are considered for building RNN model to predict the target feature, because there are only three features that change with respect to time, but the effect of constants also factor the features that change w.r.t time.

B. Data Loading and Preprocessing

The whole process to build the model, Training and Testing is performed on Google collab using TensorFlow. Therefore, the dataset npy file is stored on Google Drive associated with the google collab account. Directory path for Training Data and Validation Data is defined to access the features data from the drive. Feature's data is split into train and test in 80:20 ratio. Train data is further split into train and validation in 80:20 ratio. Final shape of train, test and validation data is (440,700,7), (183,700,7), (110,700,7). The target data should be separated from the features that will used for prediction. Data is split in two different lists as features and test. Where Features will only be 6 except the

Target, weight aimed to be poured in the receiving cup. In this process the data is transformed form a 3d array to a list and again the list is transferred back to a 3d array. Mean and standard deviation is calculated for Normalizing the data. The calculated mean and standard deviation should be calculated only for non-zero features, padded zeroes should not be included.

III. METHODOLOGY

Normalization

Training features Data, Validation features data undergoes features scaling, Scaling is applied using standardization technique only on the non-zero values. There are many techniques for features' scaling.

1. Min max normalization

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

2. Mean normalization

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

3. Standardization

$$x' = \frac{x - \text{average}(x)}{\text{std}(x)}$$

Standardization is also called as Z-score normalization, the selection of normalization technique depends on the task and the application of the model. Normally Z-score is very commonly used normalization technique. In this experiment Z-Score normalization is used. For regression problems, it is the best technique.

Optimizers and learning Rate.

Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. Optimizers are used to solve optimization problems by minimizing the function.

Model

This experiment is built on the Transfer Learning and Fine hyperparameter tuning. Therefore, this model will have more than 1000 parameters. Therefore, the proposed deep neural network is used in estimating the target, weight aimed to pour in the receiving cup.

There are several optimizers and drop out combinations that are used to train the model. Out of which Adam, outperform other optimizers. The weight is initialized using initialization strategies and is updated with each epoch according to the update equation used.

Loss

Neural networks are trained using Stochastic gradient descent, it requires loss function to design and configure model. A loss function is used to optimize an algorithm. The loss is calculated on training and validation data and these values resemble how good the model is performing in these two sets. It is the sum of errors made for each sequence in training or validation sets. Loss value implies how poorly or well a model behaves after each iteration of optimization. Neural Networks are trained using optimization process that requires a loss function to calculate error. Maximum likelihood provides a framework for choosing a loss function when training a neural network. Cross entropy and mean squared error are the two main types of loss functions to use when training neural networks. In this experiment, loss is calculated using mean_square_error function [5].

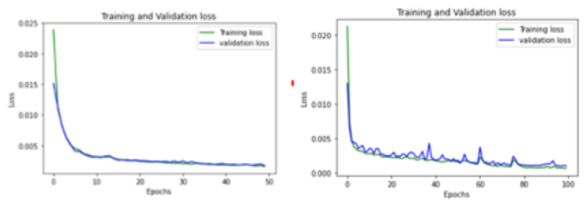


Figure 1a, 1b

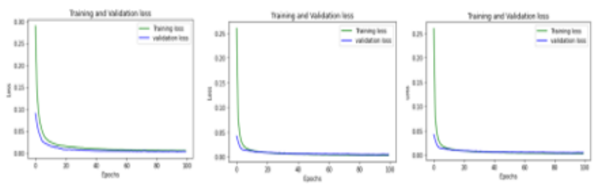


Figure 2a,2b,2c

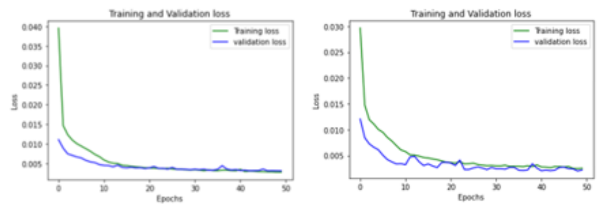
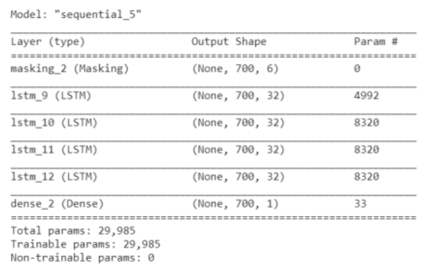


Figure 3a, 3b



```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
masking_2 (Masking)          (None, 700, 6)            0
_________________________________________________________________
lstm_9 (LSTM)                (None, 700, 32)           4992
_________________________________________________________________
lstm_10 (LSTM)               (None, 700, 32)           8320
_________________________________________________________________
lstm_11 (LSTM)               (None, 700, 32)           8320
_________________________________________________________________
lstm_12 (LSTM)               (None, 700, 32)           8320
_________________________________________________________________
dense_2 (Dense)              (None, 700, 1)            33
=================================================================
Total params: 29,985
Trainable params: 29,985
Non-trainable params: 0
```

Figure 4

In this experiment, three popular architectures for Recurrent neural networks are used, starting with Simple RNN with 4 layers each having 16 neurons at each layer and 1 dense layer since we have one output. The same architecture is applied to LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) and Number of neurons at hidden layers are increased to 32 and observed the loss and RMSE. Figure 1a, 1b represents training loss vs validation loss in LSTM, Figure 2a, 2b, 2c represent training loss vs validation loss in SimpleRNN, Figure 3a, 3b represent training loss vs validation loss in GRU.

IV. EVALUATION AND RESULTS

TABLE I.

SLNo.	Accuracy for different		
	Architecture, Optimizer, dropout, number of epochs used, Batch size, Number of hidden layers, Number of neurons at each layer	***Train Accuracy***	***Validation Accuracy***
1.	SimpleRNN, Adam Optimizer, 0.15, 100, 32,4,16	0.0045	0.0032
2.	SimpleRNN, Adam Optimizer, 0.15, 100, 32,4,32	0.0028	0.0046
3.	SimpleRNN, Adam Optimizer, 0.15, 100, 32,4,64	0.0034	0.0029
4.	LSTM, Adam Optimizer, 0.15, 100, 32, 4, 16	0.0091	0.0083
5.	LSTM, Adam Optimizer, 0.15, 100, 32, 5, 32	0.0017	0.0010
6.	LSTM, Adam Optimizer, 0.15, 100, 32, 4, 32	0.00068	0.0010
7.	GRU, Adam Optimizer, 0.15,50,32,4,16	0.0028	0.0027
8.	GRU, Adam Optimizer, 0.15,50,32,4,32	0.0026	0.0022

TABLE II.

Sl no	Model Used	RMSE on random Test Data(%.2f)
1.	LSTM 16 (4 layers)	0.04
2.	LSTM 32(4 layers)	0.03
3.	LSTM 32(5 layers)	0.04
4.	Simple RNN 16	0.06
5.	Simple RNN 32	0.05
6.	Simple RNN 64	0.05
7.	GRU 16	0.05
8.	GRU 32	0.05

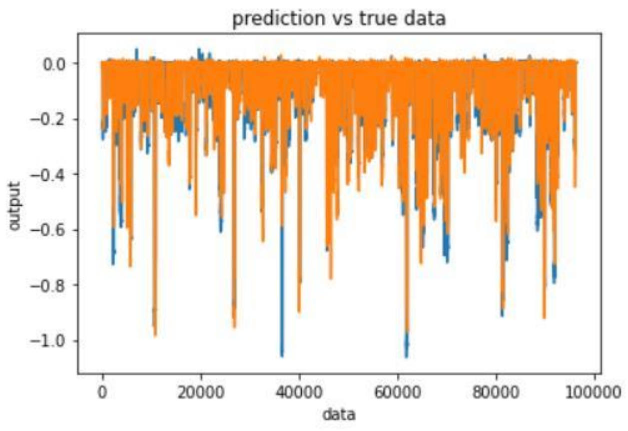


Figure 5.

V. DISCUSSION

In this experiment, three popular architectures for Recurrent neural networks are used, starting with Simple RNN with 4 layers each having 16 neurons at each layer and 1 dense layer since we have one output. The same architecture is applied to LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) and Number of neurons at hidden layers are increased to 32 and observed the loss and RMSE. Figure 1a, 1b represents training loss vs validation loss in LSTM, Figure 2a, 2b, 2c represent training loss vs validation loss in SimpleRNN, Figure 3a, 3b represent training loss vs validation loss in GRU.

VI. CONCLUSION

Several Model architectures have been tried on the given dataset by tuning parameters like changing number of layers, batch size and, Increasing number of epochs, dropout. Table 1 describes the loss obtained in each method and table2 describes the RMSE obtained on random test data. Figures 1,2,3 show the training loss vs validation loss for each model. The Validation Loss and Training loss are significantly low in LSTM model with training loss being 0.00068 and RMSE on random test data obtained is 0.03. According to the observation and results, LSTM with 32 neurons at 4 hidden layers is giving the least RMSE and the model summary is given in Figure4.

REFERENCES

[1] David Paulius, Yongqiang Huang, Roger Milton, William D Buchanan,Jeanine Sam, and Yu Sun. Functional object-oriented network for manipu-lation learning. In2016 IEEE/RSJ International Conference on IntelligentRobots and Systems (IROS), pages 2655–2662. IEEE, 2016.
[2] David Paulius, Ahmad B Jelodar, and Yu Sun. Functional object-orientednetwork: Construction & expansion. In2018 IEEE International Confer-ence on Robotics and Automation (ICRA), pages 1–7. IEEE, 2018.
[3] C. Schenck and D. Fox. Visual closed-loop control for pouring liquids. In2017 IEEE International Conference on Robotics and Automation (ICRA), pages 2629–2636, May 2017.
[4] Chau Do and Wolfram Burgard. Accurate pouring with an autonomousrobot using an RGB-D camera.CoRR,

abs/1810.03303, 2018.

[5] Chau Do, Camilo Gordillo, and Wolfram Burgard. Learning to pour usingdeep deterministic policy gradients. In2018 IEEE International Conferenceon Robotics and Automation (ICRA), 2018.

[6] Yongqiang Huang and Yu Sun. Learning to pour. In2017 IEEE/RS International Conference on Intelligent Robots and Systems (IROS), pages7005–7010. IEEE, 2017.4

[7] Yongqiang Huang and Yu Sun. A dataset of daily interactive manipulation.International Journal of Robotics Research, 2019.

[8] Yongqiang Huang, Matteo Bianchi, Minas Liarokapis, and Sun Yu. Re-cent data sets on object manipulation: A survey.Big Data, 4(4):197–216,December 2016

.[9] Min Han, Zhiwei Shi, and Wei Wang. Modeling dynamic system by recur-rent neural network with state variables. InAdvances in Neural Networks-ISNN 2004, 2004.

[10] Adam P. Trischler and Gabriele M.T. DEleuterio. Synthesis of recurrentneural networks for dynamical system simulation.Neural Networks, 80:67– 78, 2016.

[11] Alex Graves. Generating sequences with recurrent neural networks.CoRR,abs/1308.0850, 2013.

[12] Xu-Yao Zhang, Fei Yin, Yan-Ming Zhang, Cheng-Lin Liu, and YoshuaBengio. Drawing and recognizing chinese characters with recurrent neuralnetwork.CoRR, abs/1606.06539, 2016.

[13] Tianze Chen, Yongqiang Huang, and Yu Sun. Accurate pouring using model predictive control enabled by recurrent neural network. In2019IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.

[14] Juan Wilches, Yongqiang Huang, and Yu Sun. Generalizing learned manipulation skills in practice. In2020 IEEE/RSJ International Conferenceon Intelligent Robots and Systems (IROS), pages 9322–9328, 2020.

[15] Yongqiang Huang, Juan Wilches, and Yu Sun. Robot gaining accuratepouring skills through self-supervised learning and generalization.Roboticsand Autonomous Systems, 136:103692, 2021.

[16] https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

[17] Delgado, Jorge. (2013). Re: What are the best normalization methods (Z-Score, Min-Max, etc.)? How would you choose a data normalization method?. Retrieved from: https://www.researchgate.net/post/What_are_the_best_normalization_methods_Z-Score_Min-Max_etc_How_would_you_choose_a_data_normalization_method/5189e4c0cf57d7a063000031/citation/download.