CS230

Formatting instructions for CS230-Winter 2018

Tae Myung. Huh
Department of Mechanical Engineering
Stanford University
taemyung@stanford.edu

1 Introduction

Human hand has outstanding dexterity. This dexterity come not only from the hand kinematics but also from the sophisticated tactile sensing. Researchers have found that a human hand has more than ten thousands of mechanoreceptors(tactile sensing units) which can provide fine tactile acuity. Even with the enormous number of tactile sensors, human can respond very quickly to any dynamic events [1]. One of recently found reason is that the nerve system reduces number of sampling by connecting one neuron to multiple (~ 10) mechanoreceptors. Moreover each neuron's receptive field is highly intermingled to provide complex and profound tactile responses [1].

Inspired by the human nerve system, I present how a robotic tactile sensors can cluster pressure readings from tactile arrays to reduce the number of samplings, thus increasing the sampling rates. As a proof of concept, I built tactile pressure sensor arrays to measure locally distributed pressures. Using deep-learning regression, I built a model to estimate an orientation and location of contacting object. Further more I attempted to obtain the best performing clusters of tactile arrays.

2 Related work

The same authors of [1] have presented neural networks to mimic mimic the human tactile responses [2]. They built a virtual tactile pressure arrays and classified input object shapes (alphabet shapes). They built four fully connected neural network and applied non-negative constraints to weights to address actual human-neural connections. However, when similar schemes are implemented to robotic tactile sensors, more reasonable weights should have discrete values of 0 or 1 because each sensor pixel will be hard wired and the response of clusters are simply summation of the connected pixels. Thus, this project examines how the randomly connected sensor clusters work for estimating object posture.

3 Dataset and Features

In this project I intended to estimate orientation and location of a object (a rod, D=5mm) from the pressure distribution on 20 pixel sensors (Fig. 1). To collect data, I used 3 axis linear stage and custom-made rotating stage to change relative angle and translation. The rod was located at from -8mm to 8mm with respect to the center point (2mm increment). To randomize the position, I added normal distribution noise ranging (-1mm, 1mm). The angle was given at every 1 degree ranging [0deg to 180 deg]. The actual angle output was not exactly 1 deg increment because of the system friction and control; this added some random distribution around target angle and we used the actual angle output measured from the stage. Per each measure, the linear stage keep pressing the sensor about 50 micrometer and 11 pressures were measured while pressing. In total, I collected 32000 data sets of distributed pressure, relative angle and translation.

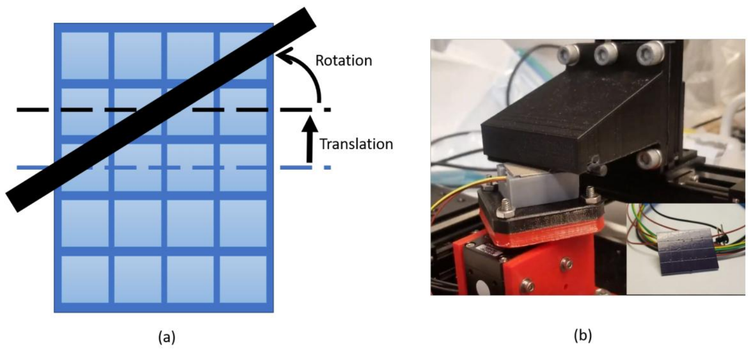


Figure 1: (a) Example of Pressure input to 20 pixel sensor array (b) Automatic stage for applying pressure at different translation and rotation angle. Inset demonstrates an actual sensor array.

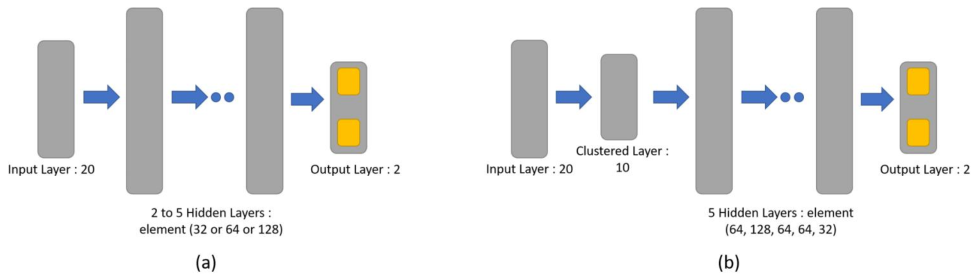

Input Layer : 20
2 to 5 Hidden Layers : element (32 or 64 or 128)
Output Layer : 2
(a)
Input Layer : 20
Clustered Layer : 10
5 Hidden Layers : element (64, 128, 64, 64, 32)
Output Layer : 2
(b)

Figure 2: (a) Neural network design for searching hyper parameters with 20 input elements (b) Neural network design for clustering the input layer to 10 element layer.

Table 1: Batch Size Study

Batch Size	128	256	512	1024
Training Loss	0.00582	0.00539	0.00521	0.00731
Compute Time for 1500 iterate (sec)	442	290	227	169

4 Methods

Because the sensor feature has only 20 elements, I used fully connected layers without any convolution layers. To obtain orientation and translation at the same time, the output layer has two elements without activation function (Fig. 2). Activation functions for hidden layers were all selected to ReLU for regression of continuous values. The measured labels (angles, translations) are normalized to range of 1; it was surmised that uniform distribution of outputs are needed to use single loss function for both labels. For loss function, I chose mean of absolute error. All the processes are done with Keras frameworks.

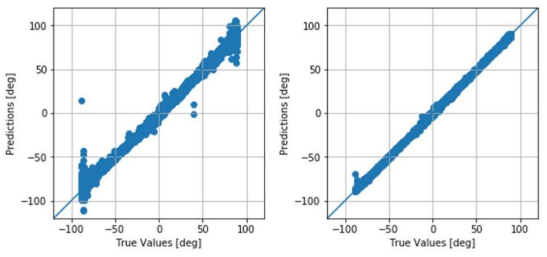
To reduce the sampling time by half, I examined clusters adding two sensors which results in 10 elements reduced from 20. To find the best combinations, I examined feature scoring of each pixel reading and combined them in two ways: a) combining the most attributing pixel and least attributing pixels b) combining two sensors in the order of highest score. The performance is compared with randomly selected combinations to validate the choice.

5 Experiments/Results/Discussion

To explore how the number of layers and neurons affect the regression, I have tested four different layer configurations as (Fig. 3 (a)). As I implemented deeper nerual network, both dev set loss and training set loss decreased significantly. With the deep neural networks, increasing number of neuron

Design label	Layer	Dev, abs(error)	Train, abs(error)
(1)	(20, 64, 32,2)	0.0113	0.0107
(2)	(20,64, 64, 32, 2)	0.0078	0.0075
(3)	(20, 64, 64, 64, 32, 2)	0.0052	0.0048
(4)	(20, 64, 128, 64, 32, 2)	0.0051	0.0046

(a)



(b)

Figure 3: (a)Hyperparameter study on number of layers and neurons (b) Regression ouputs on test sets with Design (1)(left) and (2)(right). Test absolute errors are (1): 2.26deg/0.19mm , (2): 0.75deg/0.084mm

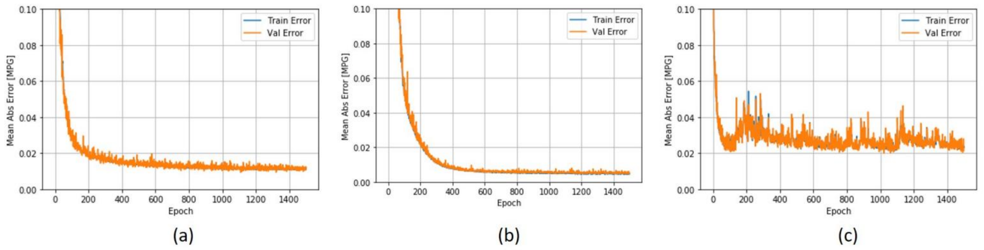


Figure 4: Learning curve for (a) RMSProp (lr = 0.001) (b) Adam (lr = 0.001) (c) Adam (lr=0.01)

Table 2: Sensor Pixel Attribution score

Excluded Pixel Index	10	16	18	8	12	13	11	3	15	5	14	19	7	2	17	1	0	4	6	9
Training Loss (%)	0.59	0.49	0.48	0.46	0.44	0.44	0.44	0.44	0.43	0.43	0.43	0.43	0.42	0.41	0.41	0.41	0.41	0.41	0.41	0.39

of a single layer (from 64 to 128) did not improve the performance noticeably, which suggests that 64 neurons might be sufficient to predict the output. For the following discussion, however, I used the deep and more neuron design (Design (4)) because more complexity would be preferred after I reduce the input elements by clustering.

To select appropriate optimizer, I have implemented with RMSprop and Adam. For both optimizers, learning rate(lr) of 0.01 and 0.001 were tested. With the same number of iteraions, the RMSprop with lr = 0.01 shows much higher training loss (>0.2) than the other cases. (Fig. 4) shows that Adam optimizer with lr=0.001 shows faster and smoother convergence; for this reason, I chose tis optimizer for the rest of the analysis.

Mini-batch size matters for the computation speed and converging speed. As in Table 1, batch size of between 128 and 512 did not show statistically different training loss. However the batch size of 1024 show significantly higher training loss. For this project I used batch size of 256, but 512 would be more desired for the faster computation.

To cluster the 20 input elements into 10 elements, the most attributing features were listed as Table 2. I scored each element by dropping it and checking the training loss; the higher training loss is, the more dropped element was attributing to the regression. As mentioned above, I tested two clustering schemes. a) the most and the least were clustered, i.e. (10, 9), (16, 6), ... b) two elements were clustered in order of score, i.e. (10, 16), (18, 8),... . When compared with randomly selected clusters (70 random clusters), a) and b) marked less loss than average (Fig. 5). Thus, these scheme might be a acceptable way of choosing the clusters yet it is sub-optimal. As a future work, I would like to implement optimization algorithms to find the best performing cluster.

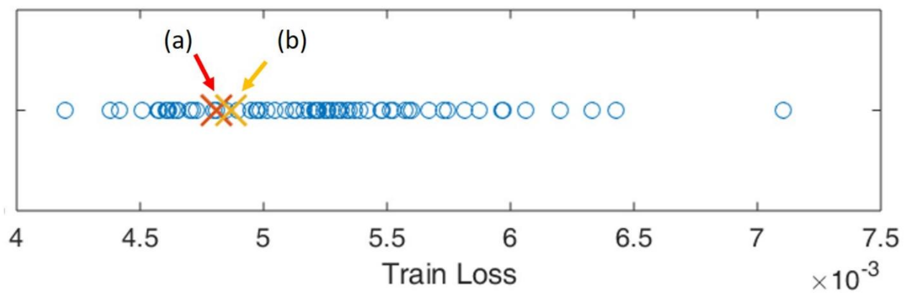


Figure 5: Training loss of clustering scheme (a),(b). Blue circle are training loss of randomly generated 70 clusters.

6 Conclusion/Future Work

Using neural network, I could successfully predict the object orientation and translation on a tactile sensor array. Hyperparameters were study to lower the training loss. Clustering 20 sensor arrays to 10 element layer was examined. Proposed clustering scheme results in less training loss than average of randomly chosen clusters. As a future work, optimizing algorithms for discrete cluster matrix can be studied.

References

[1] Pruszynski, J. Andrew, and Roland S. Johansson. "Edge-orientation processing in first-order tactile neurons." Nature neuroscience 17.10 (2014)

[2] Zhao, Charlie W., Mark J. Daley, and J. Andrew Pruszynski. "Neural network models of the tactile system develop first-order units with spatially complex receptive fields." PloS one 13.6 (2018)