# A Convolutional Neural Networks Approach to Devise Controller

*Xiangdi* Liu[1,2,⋆] and *Yunlong* Dong[1,2,⋆⋆]

[1] *Huazhong University of Science and Technology*
[2] *School of Automation*

**Abstract.** PID controller is widely used in many fields. The input sequence and output sequence of a well-parameterized PID controller are transformed into a matrix presented in images in this paper, and a Every-Timestep data augmentation algorithm is operated.This paper propose a image to image Convolutional Neural Networks(CNN) structure to learn the PID controller, which incorporates the successful methods in computer vision and deep learning to control field. The simulation is performed using Matlab/Simulink and Keras[1]. The simulation demonstrates that the CNN controller is capable in performance.
**Keywords: PID controller, CNN, Deep learning, Control.**

## 1 Introduction

In practice PID controller is the most widely used common control algorithm in many fields. PID controller has very simple structure and if parameters tuned well can achieve high performance. Neural Network is an information processing structure abstracted by the biological structure, consisting of processing element interconnected together with unidirectional signal channels called connections.As Neural Networks have three advantages, which have been widely used in pattern classification, system identification, image processing and other fields. First, Neural Networks are data-driven self-adaptive methods. Second, they are universal function approximator[2]. Third, Neural Networks are powerful nonlinear models, which make them flexible in handling real world relationships. Over the last few years, machine learning algorithms have led to more efficient methods and traditional neural networks have been developed into different structures for different problems. Neural Network(NN) was proposed to compose a PID controller[3–5]. Cheon[6] proposed a deep learning method to learn the PID controller.

This paper combines the deep learning algorithms with the control filed, and propose a CNN controller to replace the PID controller. The simulation is performed using Matlab/Simulink and the details of traditional PID controller and CNN controller were discussed.This paper is organized as follows. CNN is described in section II. In section III, Devise Controller is explained. The comparison details between the proposed CNN controller and traditional PID controller presented with the simulation results are shown in section IV. Finally, the conclusion follows in section V.

⋆e-mail: lxd@hust.edu.cn
⋆⋆e-mail: dyl@hust.edu.cn

## 2 Convolutional Neural Networks

Convolutional Neural Network(CNN) is a kind of structural variation of traditional neural network, utilizing layers with convolutional filters that are applied to extract features . After Alex Krizhevsky used Deep Convolutional Neural Networks in ImageNet classification problem[7], machine learning researcher reach a consensus that general CNN structure composed of convolution layer, pooling layer and fully connected layer. The structure is shown in Fig 1[7].

Originally invented for computer version, CNN models have subsequently been shown to be effective for many different problems including, Simultaneous Localization and Mapping(SLAM)[8], Decision Making[9] and Automatic Driving[10].

The Convolutional Neural Network(CNN) plays a significant role in image classification[7], image caption[11], image semantic segmentation[12] and so on. CNN can directly learn from raw images and automatically extract the feature which usually outperform handcrafted feature engineering. CNN uses filters to extract feature from high dimensional sensory data(especially images) and have reached enormous success in many fields.

## 3 Devise Controller

### 3.1 Traditional PID controller

To devise a novel controller, a traditional PID controller was first proposed to be the baseline. A well-parameterized PID controller was the target we want to mimic. The PID controller calculates the output in the following rule(continous time) (1), where the $k_p, k_i, k_d$ are the parameter to be settled in practice, $e$ stands for the error in feedback control loop. The control scheme is showed in Fig 2 .

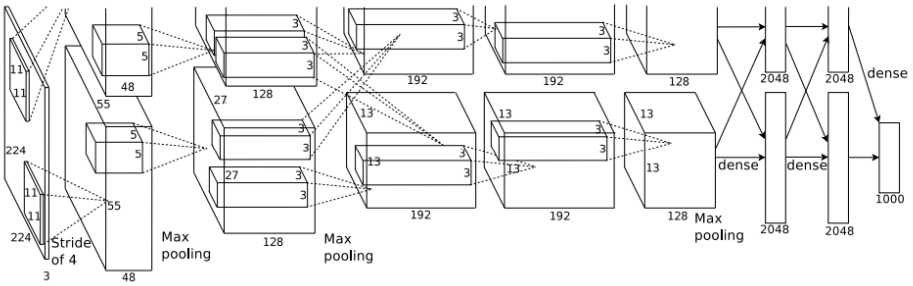


**Figure 1.** CNN structure

$$u = k_p \times e + k_i \times \int edt + k_d \times \frac{de}{dt} \tag{1}$$

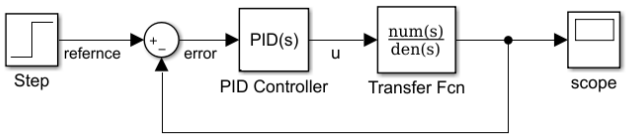


**Figure 2.** PID control scheme

The PID controller calculates the output based on the error sequence $\{e(0), e(1), e(2), ..., e(T)\}$ in the rule(Discrete time) (2). Kangbeom Cheon[6] used a Deep Believe Network to learn the PID controller's input and output, but they do not analysis the underlying principle in PID controller. The PID controller in fact consists of three filter, proportional, integrate, differential. Using the powerful efficiency of Deep learning methods, the map between PID controller input and output can be easily learned via supervising learning. The learning scheme is shown in Fig 3.

$$u(k) = k_p \times e(k) + k_i \times \Delta t \sum_{i=0}^{K} e(i) + k_d \times \frac{e(k) - e(k-1)}{\Delta t} \tag{2}$$

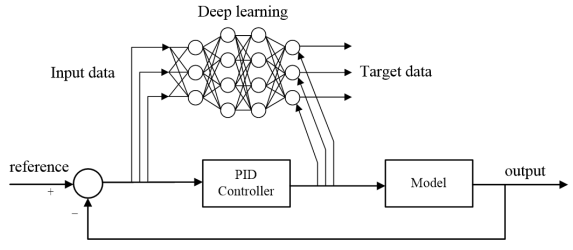


**Figure 3.** learning scheme

### 3.2 Transfer to Image

The error sequence $\{e(0), e(1), e(2), ..., e(T)\}$ is the input of our learning algorithm(Deep learning methods). Instead of directly dropping the Sequence to the model, We analysis the sequence first. The sequence is obviously a time sequence which implies the system response in time domain. The RNN[13] like models are capable of dealing with such sequence. But the time sequence is just a One-Dimensional relationship, we want further exploration. So we transform the error sequence to a matrix which has a spatial structure, which can contain more information. We samples 20s time(the default terminated time) in error with sample time 0.01s to get a 2000 length sequence. Then the Sequence is transformed into a $40 \times 50$ matrix called **error plane**. The same process was done in the output of PID controller called **u plane**. For visualization convenience, The matrix(One channel) are shown with colorful format in Fig 4.

Nilai yg benar/sinyal kontrol

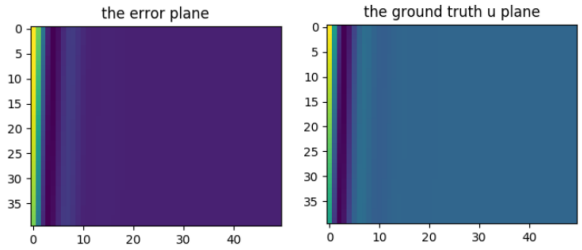


**Figure 4.** error plane(left), u plane(right)

As can be clearly seen, there is a pattern in the error plane and u plane. The u plane is like the blurry error plane. We can treat the sequence like image. There are many well-developed theory, models and algorithm processing images. Now we transfer the learning controller problem to a image to image supervising learning problem.

### 3.3 Devise CNN controller

The architecture used to devise the controller is shown in Fig 5. The architecture has purely convolutional layers. The output of the CNN controller should have the same size with the input, so there no no pooling layers and Flattern layers . The first layer has 32 $3 \times 3$ filters with relu

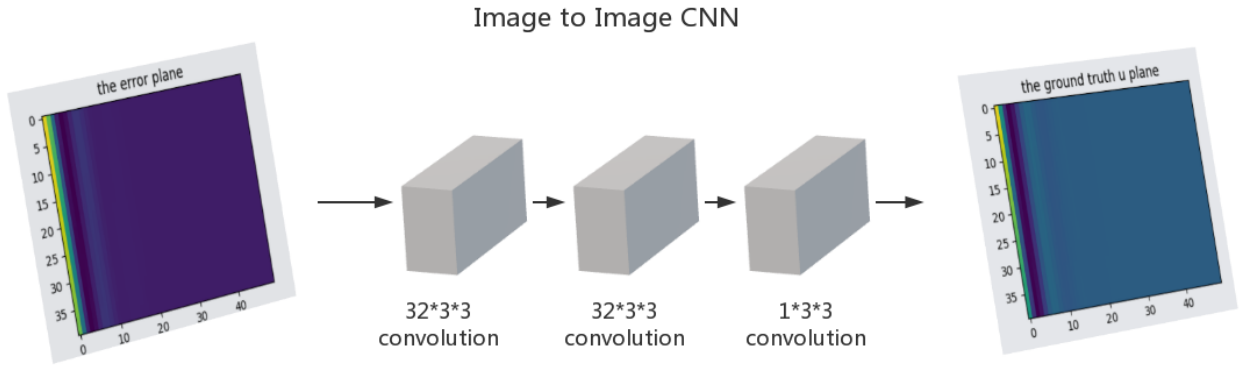


**Figure 5.** The CNN controller architecture.

activation function, the second layer has the same filters as the first layer, and the last layer have 1 $3\times3$ filter with relu activation function to make the output of CNN has size $40 \times 50$.

Two-Dimentional convolutional operation can be written in eqn (3), where $K$ is the filter with size $k_r \times k_c$, $In(s,t), out(s,t)$ are the input and output of the convolutional operation with size $N_c \times N_r$ respectively. Through convolutional operation, the CNN is able to capture the relationship and maps between error plane and u plane. Semantic, we want the filter to learn the pid caclulating rule eqn (2).

$$Out(s,t) = \sum_{m=0}^{k_r-1}\sum_{n=0}^{k_c-1} K(m,n) \times In(s-m, t-n)$$
$$s.t.\ 0 \le s \le N_r - 1,\ 0 \le t \le N - c - 1 \qquad (3)$$

### 3.4 Training

In practice the whole length of error sequence is not able to obtain, cause the length of error increases as the controlling time passing. In general, in time step $k, k < 2000$, the error sequence is $\{(e(0), e(1), e(2), ..., e(k))\}$. We fullfill the sequence to length of 2000 with zeros to obtain $\{(e(0), e(1), e(2), ..., e(k), 0, 0, ...)\}$. In training we only gather the full-length error plane and u plane, in other words, only one pair of data is available. In controlling loop, every time step should obey the rule(???), so the error plane and u plane can be split in every time step to generate a called **Every-Timestep error-u** dataset. The detail goes in **Algorithm 1**. $error, u$ are the error and u sequence with length $T$(default terminated time). Once the augmented dataset has been obtained. A mini-batch Stochastic Gradient Descent opimizer with $lerning\_rate = 0.1, batch\_size = 32, momentum = 0.9, epochs = 10$ is used to train the CNN. Keras [1] was used to achieve these stuff. The results are shown in Fig 6. We use 70% of dataset to train and the rest 30% to test, and proving that overfitting did not appear. The visualization of training peroformance is shown in Fig 7.

**Algorithm 1** Every-Timestep($error, u, T$)

```
error_gen = Zeros[T, T]
u_gen = Zeros[T, T]
for i = 1 to T do
    error_gen[i, 1 : i] = error[1 : i]
    u_gen[i, 1 : i] = u[1 : i]
end for
return {error_gen, u_gen}
```

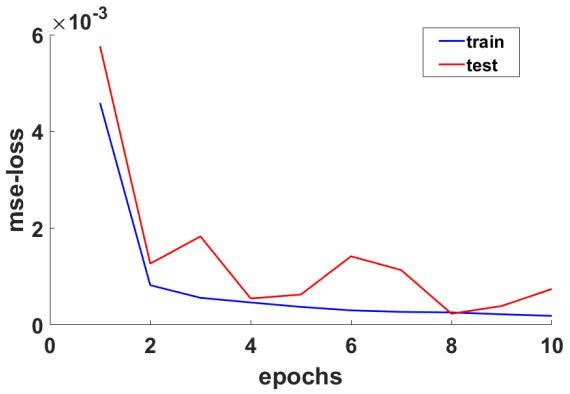


**Figure 6.** training results

## 4 Evaluation

In evaluation, we first parameterized a traditional PID controller in Simulink to simulate. The closed loop is shown in Fig 8. Then we replace the PID controller with our proposed CNN controller. Simulating in 2s. The comparing results of step response are shown in Fig 9.The CNN controller perform almost the same with PID controller even a little more smooth.

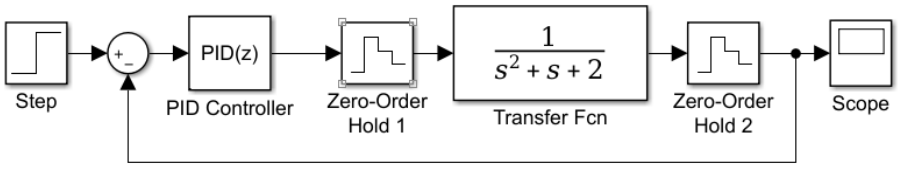


**Figure 8.** simulink model

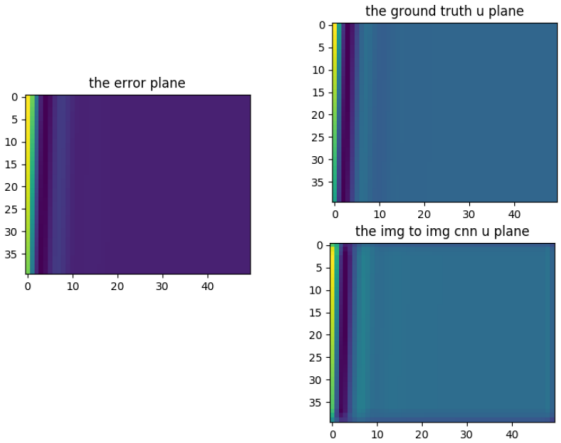


**Figure 7.** training visualization

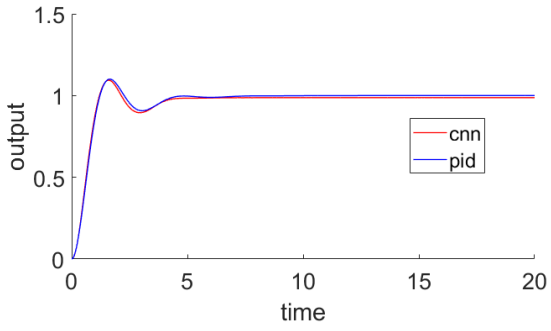


**Figure 9.** results

## 5 Conclusion

In this paper, a Convolutional Neuron Networks controller was devised as an approach to replace the PID controller. So far this paper is the first to combine images with control, incorporating the successful mothods in Computer Vision to control field, and propose a image to image CNN to learn the controller. The simulation demonstrates that the CNN controller is capable.

## References

[1] F. Chollet, *keras*, https://github.com/fchollet/keras (2015)
[2] K. Hornik, M. Stinchcombe, H. White, Neural Networks **2**, 359 (1989)
[3] F. Shahraki, M. Fanaei, A. Arjomandzadeh, Chemical Engineering Transactions pp. 1395–1400 (2009)
[4] H. Shu, Y. Pi, Computers & Chemical Engineering **24**, 859 (2000)
[5] H. Shu, Y. Pi, networks **1**, 8 (2005)
[6] K. Cheon, J. Kim, M. Hamadache, D. Lee, Journal of Automation and Control Engineering Vol **3** (2015)
[7] A. Krizhevsky, I. Sutskever, G.E. Hinton, in *Advances in Neural Information Processing Systems 25*, edited by F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Curran Associates, Inc., 2012), pp. 1097–1105
[8] T. Naseer, M. Ruhnke, C. Stachniss, L. Spinello, W. Burgard, *Robust visual SLAM across seasons*, in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on* (IEEE, 2015), pp. 2529–2535
[9] A. Giusti, J. Guzzi, D.C. Cireşan, F.L. He, J.P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro et al., IEEE Robotics and Automation Letters **1**, 661 (2016)
[10] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L.D. Jackel, M. Monfort, U. Muller, J. Zhang et al., arXiv preprint arXiv:1604.07316 (2016)
[11] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, Y. Bengio, *Show, attend and tell: Neural image caption generation with visual attention*, in *International Conference on Machine Learning* (2015), pp. 2048–2057
[12] J. Long, E. Shelhamer, T. Darrell, *Fully convolutional networks for semantic segmentation*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3431–3440
[13] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, S. Khudanpur, *Recurrent neural network based language model.*, in *Interspeech* (2010), Vol. 2, p. 3