Deep reinforcement learning for calibration of 2D-3D registration

Eungjune Shim,
Division of Bio-Medical Science & Technology, KIST School
Korea University of Science and Technology, Seoul 02792, Republic of Korea

Youngjun Kim,[*]
Center for Bionics, Korea Institute of Science and Technology, Seoul 02792, Republic of Korea

2017-05

Abstract

Deep Q Learning method is a novel approach to approximate value functions of reinforcement learning. This has been succesfully applied to solve problems such as robot control, elevator scheduling, telecommunication networks. We applied this method to a simplified 2D-3D registration problem; Point-based visual servoing simulator. The simulator environment has been virtually organized in three-dimensional space: four three-dimensional target point vectors and two-dimensional correct point vector, and a virtual camera are defined. For each timesteps, camera moves according to the output of a neural network(Q-network), and the 3D target points are projected onto the viewport. The purpose of this simulator is to reduce 2D vector error between target and correct point vectors. The Q-network takes states of current timestep, decide a action, receive rewards and update weights. The actins are defined in six: camera moves forward, backward, top, bottom, right, left. The state is defined as four 2D error vectors. As learning processes, the network moves camera with higher possibility of reducing errors. When using well-trained network, there are several benefits compare to conventional methods, such as random searching algorithms or jacobian matrix estimation. While conventional method requires computation of error variation or matrix inverse in every timestep, our proposed method only requires simple network forwarding to find its solution. Since this method used too much simplified registration environment and camera actions, the performance looks a little bit awkward, but there still are a lot to improve from this approach. Firstly, we can define each step's state with much more complex and sophisticated form by replacing the neural network. There are many Convolutional Neural Networks(CNNs) proposed to handle 2D images, the state can be defined simply using virtual camera's rendered image, not eight-digit 2D point error vectors. Secondly, the camera actions can be more natural and efficient by using the whole output possibility of the network, and combining action. As our proposed method shows considerable benefit over the conventional method, the future work from this approach can be expected to be applicable to real 2D-3D registration works.

Key words: Deep Q Learning, Reinforcement Learning, 2D-3D Registration, Visual Servoing

[*]Corresponding author email: junekim@kist.re.kr

1. Introduction

There are various methods using 2D features to perform in 3D space. As image processing and 3D technology evolves and computing power increases, many related works with this approach have become possible. Some methods using monoscopic camera when 3D scanning[1], camera calibration[2], motion capture[3], or visual servoing[4] are all using 2D features to somehow perform in 3D space. In this paper, we proposed an approach to solve 2D-3D registration problems. 2D-3D registration also uses 2D features to conduct registration in 3D. This method is largely used in medical field. Representatively, registering 3D dicom volume obtained from Computed Tomography(CT), or Magnetic Resonance Imaging(MRI) to 2D X-ray image can produce very valuable information in diagnosing patients' conditions. For example, when diagnosing patient's orthopedic condition, well-aligned 3D data gives a lot of advantages that we can visualize 3D data with information that can only be obtained from X-ray images [5, 6].
There have been many optimization approaches proposed and deviced to apply on this registration task. When 2D features are simple as points, lines or binary or faces, jacobian matrix estimation[7] method is well known as robust and largely used. This method extracts meaningful features from target image, define and calculate error between target and correct 2D feature, conduct calibration task. This method requires considerable amount of 2D data's pre-processing and feature matching in every timestep, and also need large inverse matrix operation. Decisively, this method is not applicable for most of 2D-3D registration that uses various 2D images such as x-ray radiograph. There have been many methods for image comparison have been proposed, such as edge-filtering[8] to pre-process image data in this task. Too much simplified image feature can possibly lose most of information, it might loss most of valuable information of image resulting inefficiency. Because of these reasons jacobian matrix estimation method normally cannot perform well on most of 2D-3D registration methods. Also, there are a lot of method to define errors between 2D features [9], and optimize errors. The optimization is normally applied in an iterative way such as simulated annealing algorithms[8]. This optimization task also has limitations such as local minima, or unnecessary searching.
In recent years, deep learning method has attracted a lot of attention when solving optimization problems. Especially in image or natural language processing, Most state-of-the-art methods are based on deep learning[10, 11]. in problems with very large searching space such as playing games, or 2D-3D registration, deep reinforcement learning method is commonly used. Current state-of-art reinforcement learning method is Deep Q-learning(DQN)[12] proposed by *DeepMind*. This learning method is successfully applied to learn and play complex tasks such as Atari games[13]. We also have chosen DQN method to conduct 2D-3D registration. Since our method is just approach to figure out if it can be applied to registration problems, we made models and problems as simple as possible. We succesfully applied DQN method to perform 4-point based visual servoing simulator, recorded learning outcomes and effects, and proposed a possible and strong future study plan that can be derived from this method.

2. Methods

Virtual point-based servoing simulator is composed of a camera and a plane containing four circles on it in a virtual 3D space. The plane normal and the camera view vectors are parallel to world coordinate's Y-axis. Since no rotation is possible in the condition of this simulator, camera view vector will not be changed. Normally in visual servoing, four point features are extracted from the rendered image by using image processing algorithms. Since our purpose is to test our DQN-based 2D-3D registration algorithm, we decided to skip this process. Instead of that, we pre-defined the 3D positions of four feature points, extract to the 2D rendering viewport by unproject them. This has the same effect as feature detection. This simulator runs on web browser, used javascript-based *WebGL* library named *Three.js*.

DQN training is based on Q-learning[14] algorithm, using Q-network for approximating Q-table. Experience replay and target Q are also applied to our method to prevent divergence and stablize the process. For every timestep t, environment ε, composed of statesx_t actions a_t, and rewards r_t are defined. The agent of DQN has Q-network (1), predicts rewards of each action values, so user-defined reward can backpropagate and update weight inside the network[13]. In our proposed method, we simply set the state as four error vectors of 2D target points and corresponding ground-truth points. The reward is given according to the variation of error vector size. If the total amount of the error is decreased in x_{t+1}, the agent receives reward of value 1.0, and if the error is increased, the reward is -1.0. The action a are consist of six actions as mentioned: forward, backward, up, down, right, left. For each actions, all the distance the camera move is set to 1. The agent interacts with the simulator by selecting actions in a

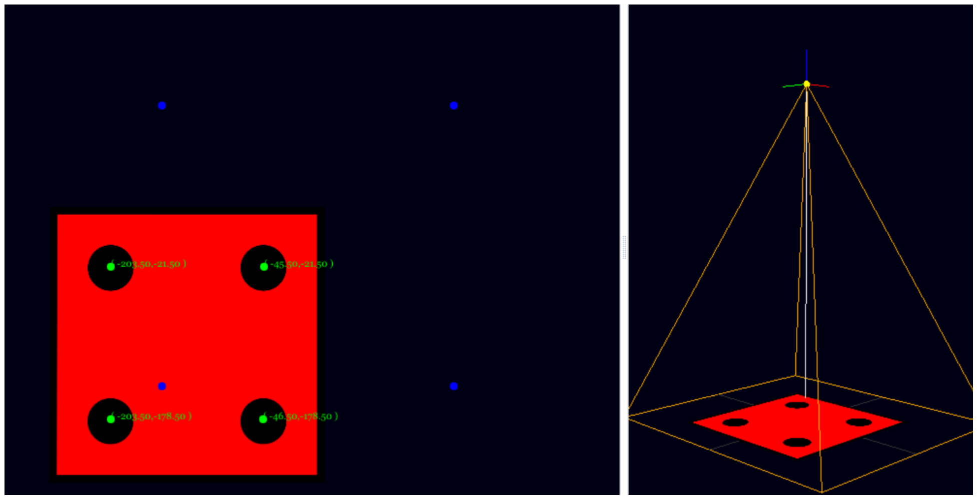

Figure 1: 3D Servoing Environment, renderer image(left), and 3D objects(right)

way that maximises future rewards. The Q-network is composed of a simple four-layered neural network: input layer, two fully-connected layers with 50 neurons, and regression output layer Fig. 2.

$$Q(x, a) \Rightarrow r \tag{1}$$

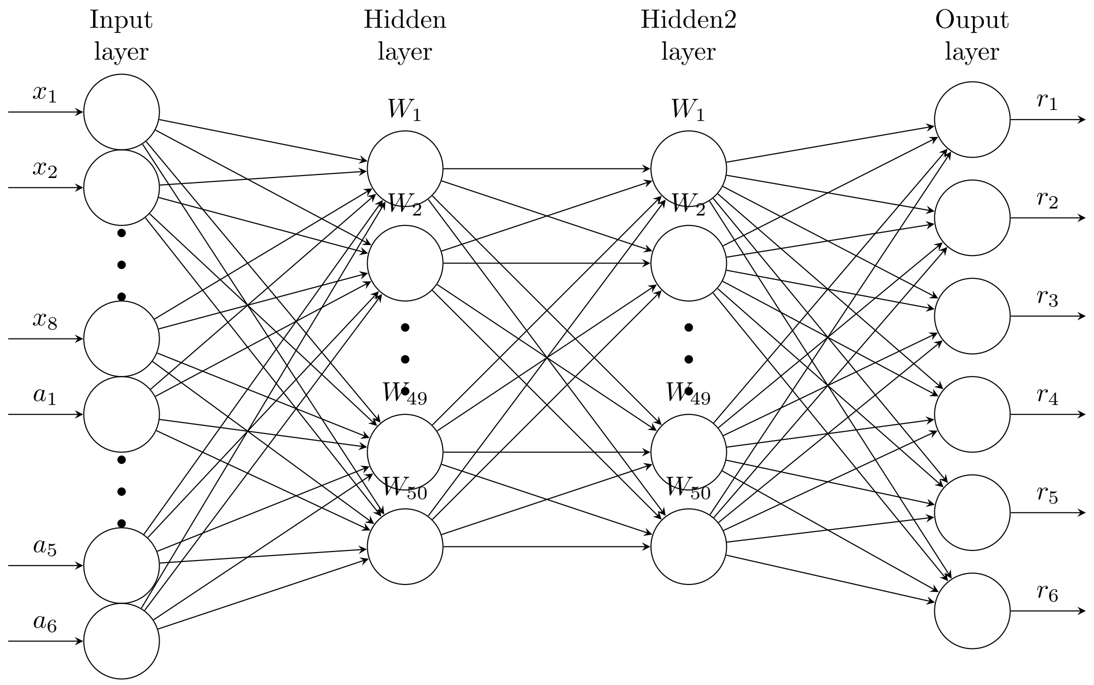

Figure 2: Q-network in the agent. Input layer is composed of states s and actions a. The output layer is rewards

In the training process, camera position is initialized randomly where all four target feature points are visible. If the error value variation E_t between x_{t+1} and x_t is less than 10.0 the camera is repositioned according to the camera position initialization function. After 100 experiences are stacked, the training starts. The agent takes the state x_t and decide action according to DQN network policy; As learning progresses, the exploration epsilon ϵ slightly decreases, and this value makes agent to rely more on Q-network then randomly select in order to choose action. We set the minimum ϵ_{min} to 0.005, and when (eq??), the learning task is done. Setting ϵ_{min} to zero is can make our process to fall into local minima really easily, so we remained to use this value to choose random action even when running calibration task using trained network. The camera performs the action specified by the agent ($camera.GoTo(a_t)$). The reward r_t is fixed according to E_t, and the agent takes reward and updates weights of Q-network ($agent.Backward(r_t)$). The epoch of training session is set from the state when camera is repositioned to random, and reach the optimal transformation that sets error value to lower than 10.0.

$$\epsilon \leq \epsilon_{min} \tag{2}$$

Algorithm 1 DQN training process for point-based visual servoing

```
1:  camera.RandomPosition()
2:  for $t$ in $T$ do
3:      $a_t$ = argmax(Q(s, a;$\theta$))
4:      camera.GoTo($a_t$)
5:      $E_t$ = Error($x_{t+1}$) - Error($x_t$)
6:      if E $\geq$ 0 then
7:          $r_t$ = 1.0
8:      else
9:          $r_t$ = -1.0
        agent.Backward($r_t$)
10:     if $r_t$ $leq$ 10.0 then camera.RandomPosition()
```

3. Results

We have plotted results of time consumption of each epochs during training(Fig.3), and after training(Fig. ????). Fig.5 shows that the error rate distribution of each epochs. Total 546 epochs were conducted to get the whole training done (??). In Fig. 3, it took 492.17 seconds to finish epoch 0. From second epoch, nothing exceeded 100.0 seconds to get the optimal state, and everything has been done within almost 10.0 seconds after epoch 76 (Fig. 3, (b)). Fig. 4 shows the

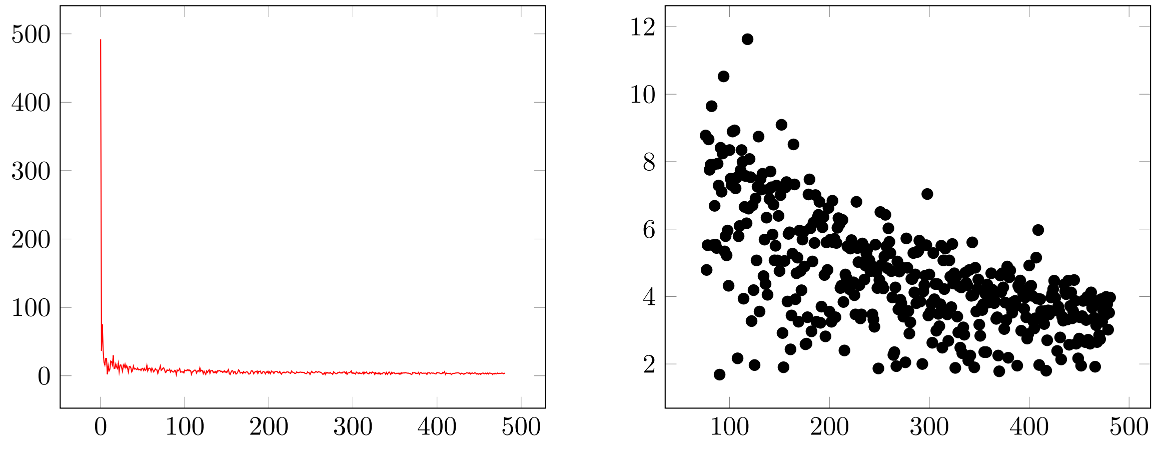

Figure 3: Searching time during learning progress

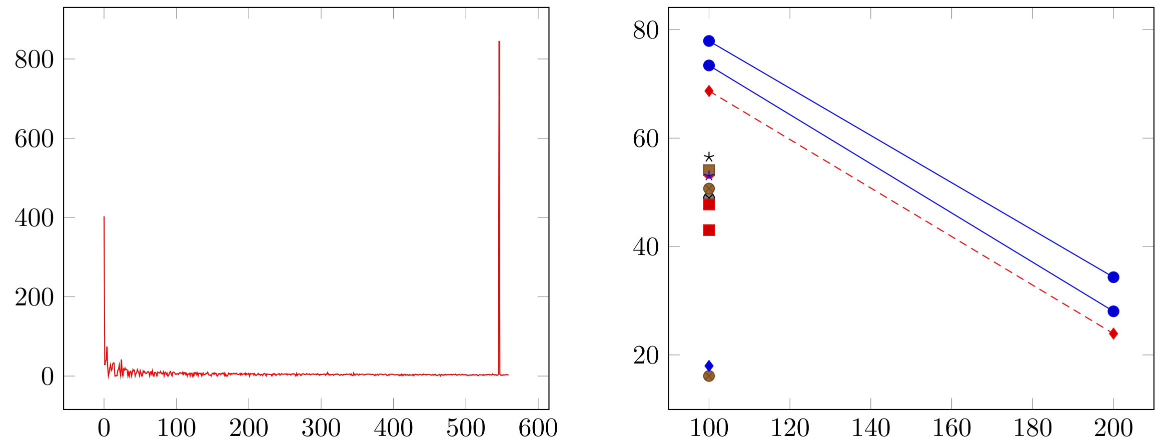

Figure 4: Well-trained network calibration error variation

4. Discussion

Our proposed DQN model is trained using 2D feature vector error as the environment ε, decided correct action a for servoing virtual camera in 3D space. Under our DQN policy, the model became very dependent the Q-network deciding action after ??????? steps. From this point, the network is well-trained as far as possible, so further training is meaningless. This well-trained model can conduct registration from a random position to the optimal position within average of ??? seconds. The visualized camera moved smoothly with almost no unnecessary movement. Compare to conventional method, such as random tree searching algorithm or jacobian matrix estimation, our proposed method has several benefits. First, the amount of calculation is greatly reduced while conventional method requires large amount of computations. Both methods requires calculation of error varation in every timestep. Furthermore, jacobian matrix estimation method requires to calculate inverse (or pseudoinverse) matrix everytime. In random searching algorithm, such as simulated annealing can easily to fall into local minima, and also, there are too much unnecessary movement durign registration. The DQN method requires such calculation only when it is being trained. Once training task is done, this does not require any complex computation, and shows efficient servoing movement. Second, the proposed method is very easy to implement. Once the form of the network, environment, actions, rewards and some parameters are configured, no more difficult tasks are needed except waiting the agent to be trained.

For now, there are several limitations that need to be improved. Firstly, Camera movement is somewhat inefficient like staircase. Because the direction and range in which camera can move is fixed in six actions. The jacobian matrix estimation method can put out all-round 3D translation and rotation vector, this method currently shows much smoother motion during registration. Since all the distance the camera move is 1, it is also inefficient in terms of utilization of error sizes. Secondly, states x_t is too simple to apply to real 2D-3D registrations. Our current method defined x_t as four 2D point vector errors, but real 2D-3D registration, larger and much more complex features need to be used to define current state.

Most of this limitations can soon be resolved in the future works. The camera movement problem can be solved by combining actions. Instead of choosing a single action predicted by $argmax(Q(s,a;\theta))$, we can use the whole values in output layer of Q-network, and add the action vectors multiplied by the corresponding predicted reward values. This can give a much higher degree of freedom(DOF) to camera motion. Also, adding rotation action is necessary to perform complicated 2D-3D registration. More sophisticated form of stats x can also be easily defined. This is the greatest potential and advantage of using DQN rather then other methods; Comparing jacobian matrix estimation method, as the feature of input state gets more complicated, the computation and complexity for deciding proper action gets exponentially more difficult. This method can use CNN that can self-study the feature information of input images, this method is extreamly easy to implement, and also, can expect good results. There already are many DQN methods succesfully applied to play games that uses rendered game images as input state, deep CNN as its Q-networks[13]. Setting stochastic reward can also efficiently utilizes the size of error variation, and reduces registraiton time. To sum it up, our future DQN model uses deep CNN for its value network. takes rendered image of each timestep as input state, predicts actions, and perform registration by combining whole output Q-network layer.

5. Conclusion

We have proposed and developed a DQN method for 2D-3D registartion. This method is succesfuly applied to optimize and solve the point-based visual servoing simulator. Since this simulator takes 2D feature states to perform in 3D, this basically can be regarded as very simplified 2D-3D registration simulator. Although this is simplified problem, the proposed approach has its strength in terms of solving similar but more complex problems compare to conventional registration methods; For example, jacobian matrix estimation method requires exponential computation and high difficulty as the problem becomes more complicated. Also, most of searching algorithm such as simulated annealing, heel climbing methods for optimizing 2D-3D registration requires to calculate the similarity of two states, and the complexity of two states gets higher, the amount of computation also gets larger. While the well-trained DQN model only forwards input states to the network to find proper actions. Thanks to this very powerful strengths we can expect to resolve most of limitations in the near future, and apply this apporoach to the higher-order 2D-3D registration problems.

Acknowledgement

This research was supported by the KIST institutional program (??????, ???????).

References

[1] W. Niem, J. Wingbermuhle, Automatic reconstruction of 3d objects using a mobile monoscopic camera, in: 3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in, IEEE, 1997, pp. 173–180.

[2] D. Li, J. Tian, An accurate calibration method for a camera with telecentric lenses, Optics and Lasers in Engineering 51 (5) (2013) 538–541.

[3] H. Wang, H.-T. Li, S. Manjunath, Real-time capturing and generating stereo images and videos with a monoscopic low power mobile device, uS Patent 8,970,680 (Mar. 3 2015).

[4] F. Chaumette, S. Hutchinson, P. Corke, Visual servoing, in: Springer Handbook of Robotics, Springer, 2016, pp. 841–866.

[5] J.-M. Kim, S.-H. Hong, B.-S. Lee, D.-E. Kim, K.-A. Kim, S.-I. Bin, Femoral shaft bowing in the coronal plane has more significant effect on the coronal alignment of tka than proximal or distal variations of femoral shape, Knee Surgery, Sports Traumatology, Arthroscopy 23 (7) (2015) 1936–1942.

[6] K. Radtke, C. Becher, Y. Noll, S. Ostermeier, Effect of limb rotation on radiographic alignment in total knee arthroplasties, Archives of orthopaedic and trauma surgery 130 (4) (2010) 451–457.

[7] D. Kosmopoulos, Robust jacobian matrix estimation for image-based visual servoing, Robotics and Computer-Integrated Manufacturing 27 (1) (2011) 82–87.

[8] Y. Kim, K.-I. Kim, J. hyeok Choi, K. Lee, Novel methods for 3d postoperative analysis of total knee arthroplasty using 2d–3d image registration, Clinical Biomechanics 26 (4) (2011) 384–391.

[9] C. Gendrin, P. Markelj, S. A. Pawiro, J. Spoerk, C. Bloch, C. Weber, M. Figl, H. Bergmann, W. Birkfellner, B. Likar, et al., Validation for 2d/3d registration ii: The comparison of intensity-and gradient-based merit functions using a new gold standard data set, Medical physics 38 (3) (2011) 1491–1502.

[10] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising, IEEE Transactions on Image Processing.

[11] Y. Goldberg, Neural network methods for natural language processing, Synthesis Lectures on Human Language Technologies 10 (1) (2017) 1–309.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602.

[14] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (3-4) (1992) 279–292.

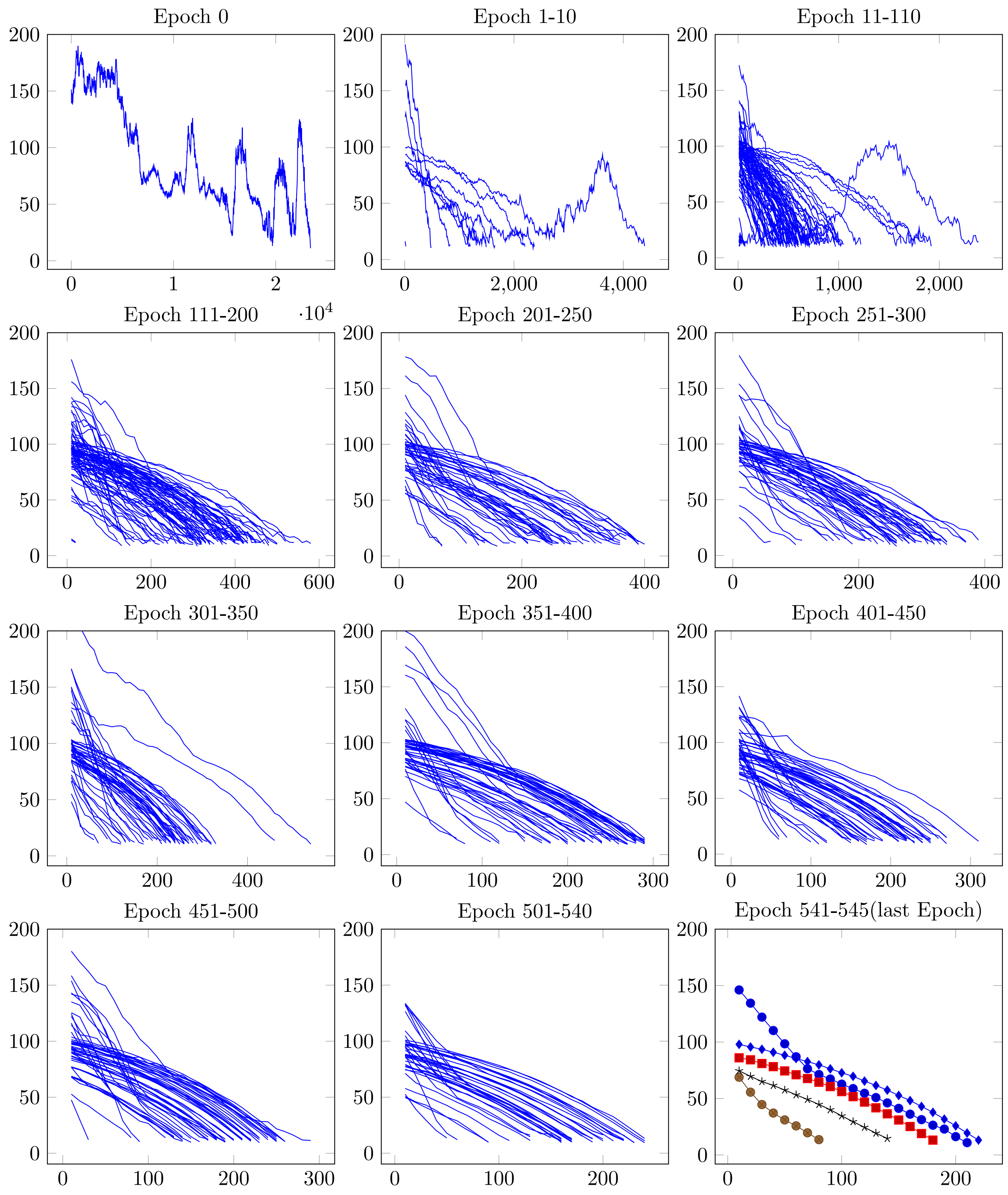

Figure 5: Error rate variation