QIM: Quantifying Hyperparameter Importance for Deep Learning

Dan Jia[1,2], Rui Wang[1], Chengzhong Xu[2], and Zhibin Yu[2(✉)]

[1] Beihang University, Beijing, China
[2] Shenzhen Institute of Advanced Technology, CAS, Shenzhen, China
zb.yu@siat.ac.cn

Abstract. Recently, Deep Learning (DL) has become super hot because it achieves breakthroughs in many areas such as image processing and face identification. The performance of DL models critically depend on hyperparameter settings. However, existing approaches that quantify the importance of these hyperparameters are time-consuming.

In this paper, we propose a fast approach to quantify the importance of the DL hyperparameters, called QIM. It leverages Plackett-Burman design to collect as few as possible data but can still correctly quantify the hyperparameter importance. We conducted experiments on the popular deep learning framework – Caffe – with different datasets to evaluate QIM. The results show that QIM can rank the importance of the DL hyperparameters correctly with very low cost.

Keywords: Deep learning · Plackett-burman design · Hyperparameter

1 Introduction

Deep learning (DL) is a sub-field of machine learning (ML) that focuses on extracting features from data through multiple layers of abstraction. While DL algorithms usually behave very differently with variant models such as deep belief networks [8], convolutional networks [13], and stacked denoising autoencoders [17], all of which have up to hundreds of hyperparameters which significantly affect the performance of DL algorithms.

Due to the inability for any one network to best generalize for all datasets, a necessary step before applying DL algorithm to a new dataset is to select an appropriate set of hyperparameters. To address this issue, a number of approaches are developed and the most popular three ones are (1) manual search, (2) grid search, and (3) random search [3]. These approaches have their respective advantages and disadvantages. However, how to optimize the hyperparameter settings for DL algorithms is still an open question.

There has been a recent surge of interest in more sophisticated hyperparameter optimization methods [1,3,9,15]. For example, [3] has applied Bayesian optimization techniques for designing convolutional vision architectures by learning

G.R. Gao et al. (Eds.): NPC 2016, LNCS 9966, pp. 180–188, 2016.
DOI: 10.1007/978-3-319-47099-3_15

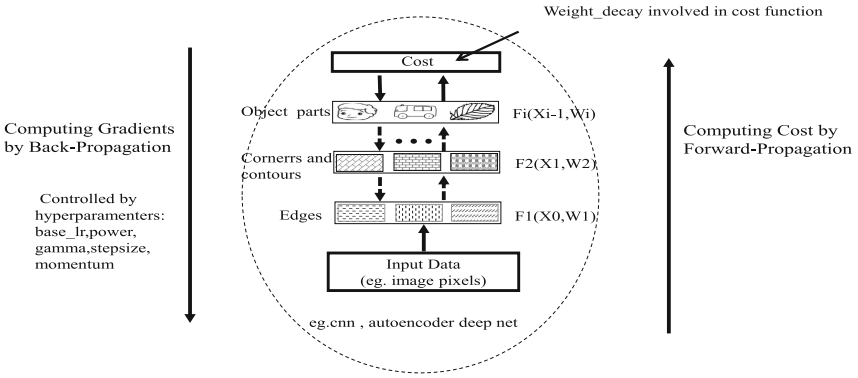

Fig. 1. Deep learning architecture

a probabilistic model over the hyperparameter search space. However, all these approaches have not provide scientists with answers to questions like the following: how important is each of the hyperparameters, and how do their values affect performance? The answer to such questions is the key to scientific discoveries. However, not much work has been done on quantifying the relative importance of the hyperparameters that does matter.

In this paper, we propose to quantify the importance of hyperparameters of DL using PB design [14], called QIM. To evaluate QIM, we employ Caffe [10] to implement the DL algorithm with the lenet [13] and auto-encoder [5] model. The results show that QIM is able to assess the importance of the five hyperparameters consistently with the assessments of ANOVA in both models. On the other hand, we demonstrate that QIM is 3× faster than ANOVA on average. In particular, we make the following contributions in this paper:

- We propose a PB design based approach to quantify the importance of hyperparameters of DL algorithms, called QIM.
- We leverage Cafe to implement two versions of the DL algorithm to evaluate QIM. The results show that QIM is able to correctly assess the importance of the hyperparameters of DL but is 3× faster than other approaches.

This paper is organized as follows. Section 2 describes the background of the DL and the PB design approach. Section 3 introduces our QIM. Section 4 describes the experimental setup for evaluating QIM. Section 5 presents the results and analysis. Section 6 describes the related work and Sect. 7 concludes the paper.

2 Background

2.1 Deep Learning (DL)

Generally, DL is a type of machine learning (ML) but is much more powerful than traditional ML. The great power of DL are obtained by learning to represent the world as a nested hierarchy of concepts, with each concept defined in

Table 1. The PB design matrix with 8 experiments.

	Parameters or factors						
Assembly	x1	x2	x3	x4	x5	x6	x7
1	+1	+1	+1	−1	+1	−1	−1
2	−1	+1	+1	+1	−1	+1	−1
3	−1	−1	+1	+1	+1	−1	+1
4	+1	−1	−1	+1	+1	+1	−1
5	−1	+1	−1	−1	+1	+1	+1
6	+1	−1	+1	−1	−1	+1	+1
7	+1	+1	−1	+1	−1	−1	+1
8	−1	−1	−1	−1	−1	−1	−1

relation to simpler concepts, as shown in Fig. 1. Each DL algorithm generally includes two sub-algorithms: *forward-propagation* and *back-propagation*. Most DL algorithms come with many hyperparameters that control many aspects of the learning algorithm behavior. Generally, properly setting the values of the hyperparameters is utter important but it is also difficult. The hyperparameters assessed in this paper include *learning rate*, *momentum*, *weight decay*, *gamma*, *power*, and *stepsize*.

Learning rate is a crucial hyperparameter for stochastic gradient descent(SGD) algorithm [2] which is used in the back-propagation algorithm. Momentum is designed to accelerate the learning process. Weight decay is designed to prevent the 'overfitting'. In other words, it governs the regularization term of the neural net which is added to the network's loss. The other hyperparameters, gamma, power, and stepsize are used to adjust the value of learning rate.

2.2 PB Design

The Plackett-Burman (PB) design [14] approach is one of the fractional factorial designs. PB design utilizes two values for each parameter: the high value is denoted as '+1' and the low value is denoted as '-1'. The '+1' and '-1' signs for the individual trial experiments are assigned in a cyclical manner. It involves $4k$ experiments, where $k = 1, 2, 3..., n$. In each case the maximum number of parameters that can be studied is $4k - 1$. For example, an 8-experiment PB design can study no more than 7 parameters.

We employ an example to describe how PB design works. Suppose that we want to quantify the parameter importance of a system with five parameters. The complete design is shown in Table 1. The importance of parameter x_1 is computed as:

$$m_1 = [r_1 + r_4 + r_6 + r_7 - r_2 - r_3 - r_5 - r_8] \tag{1}$$

where r_i is the performance measurement on experiment i. The importance of m_2, m_3, m_4, m_5 can be computed similarly.

Note that we want to quantify the importance of only 5 parameters but we construct a matrix with rows of 7 parameters; this is required by the PB design approach. However, we can use the quantities of the dummy parameters (m_6 and m_7) to represent experimental errors.

3 QIM

3.1 Overview

Figure 2 shows an overview of QIM. As can be seen, step 1 determines the value range of each hyperparameter by random search. The upper bound is used as the high value (+1) and the lower bound is used as the low value (-1) for each parameter. Step 2 employs the PB deign approach to quantify the importance of each hyperparameter of DL. This step contains several sub-steps which will be elaborated later. Step 3 validates whether the results obtained by QIM are correct or not. We use ANOVA in this paper because many studies use ANOVA as a golden reference. To demonstrate the efficacy of QIM, we employ two learning models: *lenet* and *auto-encoder*, each with two data sets: *CIFAR10* and *MNIST*. We form four types of experiments:*Lenet-cifar10*, *Lenet-mnist*, *Auto-cifar10* and *Auto-mnist*.

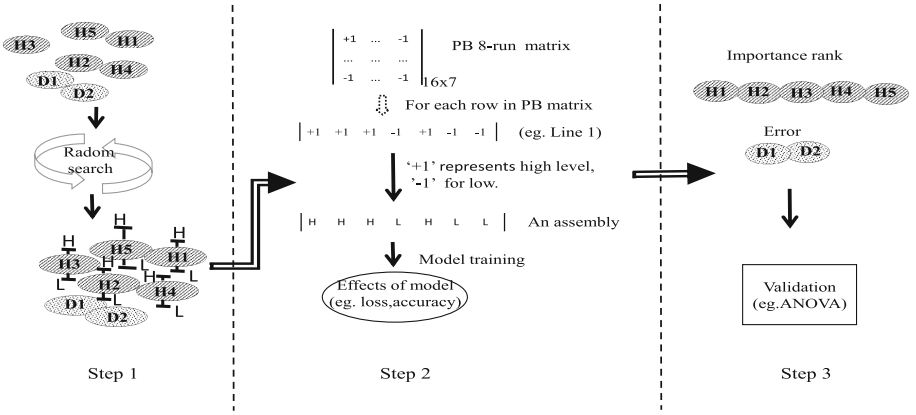

Fig. 2. Overview of QIM

3.2 Identifying the Value Range for Each Hyperparameter

In order to use QIM correctly, we need to know the value range of each hyperparameter for a certain DL algorithm. We propose a way named tentatively search(TS) to decide the value ranges of the hyperparameters. As shown in

Fig. 3, we iteratively decrease or increase the value of a parameter by a certain step-size while keep the values of all other parameters fixed and measure the performance. When we increase the value of the hyperparameter again and again until the gradient between the last two points achieves zero like CD shows, we choose the value of the hyperparameter corresponds to point C as the upper bound of the parameter. Another case is that DL algorithm fails to run successfully when we increase or decrease the value of the parameter further. It indicates that we already find the upper or lower bound of the hyperparameter in the previous try. In summary, we can find the bounds of the hyperparameters in either case.

3.3 QIM

We now turn to describe QIM in detail. We first describe the hyperparameters used in this study. The common hyperparameters used for all four types experiments include *base_lr*, *momentum*, *weight_decay* and *gamma*. The hyperparameter *power* is only used for *lenet-cifar10* and *lenet-mnist*, and *stepsize* only for *auto-cifar10* and *auto-mnist*. Since $4 \times 1 < 5 < 4 \times 2$, we use the $N = 8$ PB design as showed in Table 1 and we have two dummy parameters. To improve the confidence of QIM, we design 16-run trails instead of the 8-run one proposed by PB design. This is achieved by adding a mirroring line for each line in Table 1. For each type of experiment, we run 16 trails with different hyperparamenters settings corresponding to PB matrix. Then the importance of each hyperparameter is computed by using Eq. (1).

4 Experimental Setup

We conduct our experiments on two Intel 8-core Xeon(R) E5-2650, 2.60 GHz CPUs equipped with one Nvidia Tesla K20Xm GPU card. Caffe is used to implement the deep learning networks in our experiment. We use two versions of deep learning networks which are *Lenet* and *Auto-encoder*. The datasets used for these learning algorithms are *mnist* and *CIFAR10* respectively. The *mnist* [12] has 28×28 pixel grey-scale images of digits, each belonging to one of ten handwritten digit classes. The *CIFAR10* [11] consists of 60000 32×32 colour images in 10 classes, with 6000 images per class.

5 Evaluation

We first report the results with supervised learning and then with unsupervised learning algorithm.

5.1 Supervised Learning

The supervised learning algorithm used in this DL study is *lenet*. We feed lenet with data set CIFAR10 and MNIST respectively.

Results on CIFAR10 — Case 1. Figure 4 shows the importance obtained by QIM and ANOVA on CIFAR10. The importance rank given by QIM, from the most important to least important, is *base_lr*, *weight_decay*, *power*, *gamma* and *momentum*. On the other hand, ANOVA gives the similar rank except the

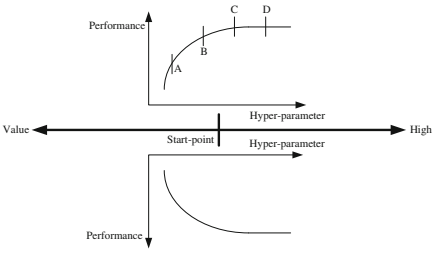

Fig. 3. Determining the value range of each hyperparameter.

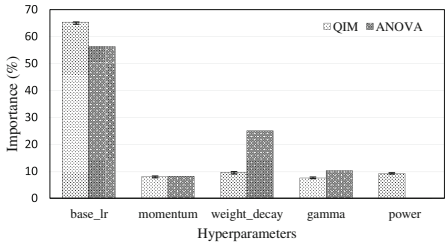

Fig. 4. Comparison of hyperparameter importance evaluated by QIM and ANOVA with *lenet* on data set CIFAR10.

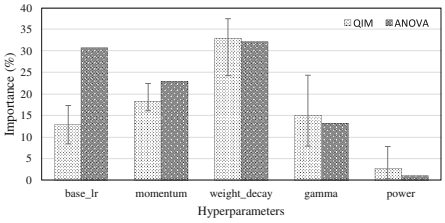

Fig. 5. Comparison of hyperparameter importance evaluated by QIM and ANOVA with *lenet* on data set MNIST.

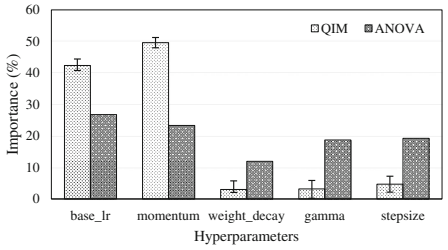

Fig. 6. Comparison of hyperparameters importance evaluated by QIM and ANOVA with *Auto − encoder* on data set CIFAR10.

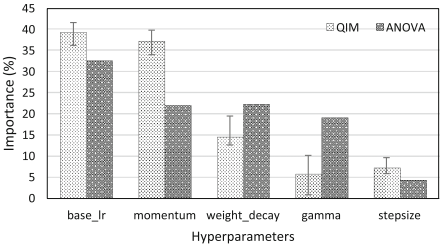

Fig. 7. Comparison of hyperparameter importance evaluated by QIM and ANOVA with Auto-encoder on data set MNIST.

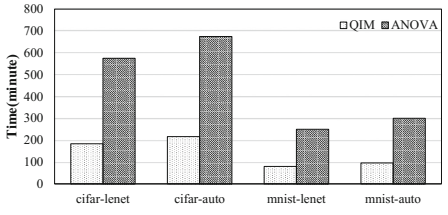

Fig. 8. Comparison of time used by QIM and ANOVA to rank the hyperparameter importance.

importance of the hyperparameter power. In this experiment, QIM introduces an error of 10.52 %. This indicates that the importance rank obtained by QIM is generally correct and we can use it in practice. Moreover, we find the importance of *base_lr* is much higher than those of other hyperparameters, which implies that the *base_lr* dominates the performance of DL with *lenet* on CIFAR10 (Fig. 6).

Results on MNIST — Case 2. The task is to classify the images into 10 digit classes. Figure 5 compares the results of hyperparameter importance with QIM and ANOVA. As can be seen, both methods rank the *weight_decay* as the most important parameter and the *power* as the least important one. QIM treats the *base_lr* less important than ANOVA does while the two approaches give the similar importance for both *momentum* and *gamma*. In this experiment, QIM introduces an error of 5.12 %, which is smaller than the error obtained in the first case. This indicates that the the importance rank of the hyperparameters obtained by QIM is more convincible than the first case.

5.2 Unsupervised Pre-training

As a unsupervised pre-training model, deep auto-encoder [17] is trained on CIFAR10 and MNIST respectively.

Results on CIFAR10 — Case 3. QIM gives the top two importance to *base_lr* and *momentum*, which are consistent with the results of ANOVA. The error rate of QIM in this experiment is 5.4 %. For the less important hyperparameters such as *weight_decay*, *gamma*, *power*, *stepsize* assessed by ANOVA, QIM also gives the similar importance rank but with different absolute importance values. Comparing to the case 1,we find that the learning algorithm used in DL significantly affects the importance of its hyperparameters as well.

Results on MNIST — Case 4. QIM and ANOVA rank the same top three important hyperparameters including *base_lr*, *momentum*, and *weight_decay* as shown in Fig. 7. In this experiment, the error of QIM is 14.32 % which seems high. However, QIM assesses the importance of hyperparameters consistently with ANOVA but with less iterations.

5.3 Time Cost

Figure 8 compares the time used by QIM and ANOVA to rank the importance of hyperparameters of DL. As can be seen, the time used by QIM is 3× less than that used by ANOVA on average. As evaluated above, QIM can correctly rank the hyperparameter importance. This indicates that QIM is indeed a fast and efficient approach for quantifying the importance of hyperparameters.

6 Related Work

There are a lot of studies focusing on optimizing hyperparameters of DL algorithm [1,3,4,9]. In low-dimensional problems with numerical hyperparameters, the best available hyperparameter optimization methods use Bayesian optimization [6] based on Gaussian process models, whereas in high-dimensional and discrete spaces, tree-based models [4], and in particular random forests [9,16], are more successful [7]. Such modern hyperparameter optimization methods have achieved considerable recent success. For example, Bayesian optimization found a better instantiation of nine convolutional network hyperparameters than a domain expert, thereby achieving the lowest error reported on the CIFAR-10 benchmark at the time [15]. However, these studies do not quantify the importance of the hyperparameters while QIM does.

7 Conclusion

In this work, we propose an efficient PB design based approach to quantify the importance of the hyperparameters of DL algorithms named QIM. With 5–15 % of error, QIM can effectively assesses the importance of each hyperparameter with much smaller number of computation iterations. We empirically validate QIM with two deep models on two data sets. The results show that QIM can rank the importance of hyperparameters of DL algorithms correctly in four cases.

Acknowledgements. We thank the reviewers for their thoughtful comments and suggestions. This work is supported by national key research and development program under No.2016YFB1000204, the major scientific and technological project of Guangdong province (2014B010115003), Shenzhen Technology Research Project (JSGG20160510154636747), Shenzhen Peacock Project (KQCX20140521115045448), outstanding technical talent program of CAS, and NSFC under grant no U1401258.

References

1. Bardenet, R., Brendel, M., Kégl, B., Sebag, M.: Collaborative hyperparameter tuning. In: 30th International Conference on Machine Learning (ICML 2013), vol. 28, pp. 199–207. ACM Press (2013)
2. Bengio, Y., Goodfellow, I.J., Courville, A.: Deep learning. An MIT Press book in preparation (2015). Draft chapters available at http://www.iro.umontreal.ca/~bengioy/dlbook
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(1), 281–305 (2012)
4. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, pp. 2546–2554 (2011)
5. Bourlard, H., Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. Biol. Cybern. **59**(4–5), 291–294 (1988)

6. Brochu, E., Cora, V.M., De Freitas, N.: A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling, hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599 (2010)
7. Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., Leyton-Brown, K.: Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In: NIPS Workshop on Bayesian Optimization in Theory and Practice (2013)
8. Hinton, G.E., Osindero, S., Teh, Y.-W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
9. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25566-3_40
10. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the ACM International Conference on Multimedia, pp. 675–678. ACM (2014)
11. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
12. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural comput. **1**(4), 541–551 (1989)
13. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
14. Plackett, R.L., Burman, J.P.: The design of optimum multifactorial experiments. Biometrika **33**(4), 305–325 (1946)
15. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, pp. 2951–2959 (2012)
16. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K., Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 847–855. ACM (2013)
17. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)