Estimating Optimal Depth of VGG Net with Tree-Structured Parzen Estimators

Sunghwan Yoo
Masoom A. Haider
Farzad Khalvati

Dept of Medical Imaging
Sunnybrook Research Institute
University of Toronto, Toronto, ON, Canada

Abstract

Deep convolutional neural networks (CNNs) have shown astonishing performances in variety of fields. However, different architectures of the networks are required for different datasets, and finding right architecture for given data has been a topic of great interest in computer vision communities. One of the most important factors of the CNNs architecture is the depth of the networks, which plays a significant role in avoiding over-fitting. Grid Search is widely used for estimating the depth, but it requires huge computation time. Motivated by this, a method for finding an optimal architecture depth is introduced, which is based on a hyper-parameter optimizer called Tree-Structured Parzen Estimators (TPE). In this work, we show that the TPE is capable of estimating the CNNs architecture depth with an accuracy of 83.33% with CIFAR-10 dataset and 60.00% with CIFAR-100 dataset while it reduces the computation time by more 70% compared to the Grid Search.

1 Introduction

Ever since Alex Krizhevsky *et al.* introduced CNNs [1], Alex Net [2], in 2012, constructing variety of the architectures of CNNs to achieve better performance in variety of fields such as computer vision [2] and natural language processing [3] has been a topic of great interest in computer vision communities. Each year, multiple academic and industrial research groups have come up with new CNNs‘ architectures such as ZF Net [4] and VGG Net [5, 6]. Depending on the size and types of datasets, different architectures are required for optimal performance. The depth of CNNs is one of the most important factors because as images sizes become larger, it requires a deeper network to capture the features of images effectively. However, as CNNs gets deeper, it would easily be over-fitted [7]. To overcome the over-fitting by the very deep networks, designers should come up with innovative ideas such as residual layers[7]; otherwise precise hand-tuning of hyper-parameters becomes inevitable, which is a very challenging task. On the other hand, if the network is shallow, CNNs cannot effectively capture the features of images leading to under-performing architectures. Therefore, finding optimal depth of CNNs in a given dataset is one of the most important tasks in deep learning.

Nevertheless, the depth of the CNNs architecture is one of the hyper-parameters, which requires great amount of effort and time to hand-tune for best performance. In order to design the VGG Net, which was placed 2^{nd} in 2015 ImageNet Large-Sacle Visual Recognition Challenge (ILSVRC) [6], 6 different CNNs Configurations with depths of 11 weight layers to 19 weight layers were tested [5]. For Resdual network (Res Net) [7], which has 152 weighted layers, the Microsoft Research team has tested networks with the depths of 20 weight layers to the one with the depths of 1202 weight layers, which required a significant amount of effort and resources.

Grid search is one of the widely used algorithms for such tasks. The idea of the grid search is straight forward; testing out every possible choice and combinations of hyper-parameters and picking the best performing choices. This algorithm works effectively when the total number of choices and combinations is not overwhelming and the training time of CNNs is short; but that is not the case in real life scenarios and hence, it is not practical in most of cases. This the main reason that the CNNs designers prefer hand-tuning the architectures. Humans can learn from mistakes and their experience and analyze it to make better architectures for a better performance, but it requires great amount of manual labor, which takes time and effort. Moreover, inexperienced CNNs designers lack the experience and are unable to decide the starting line to make an architecture from scratch and thus, end up wasting time and resources. In addition, a human brain is not capable of memorizing all results and reasons corresponding to those results. If the process of deciding the depth of architecture can be automated, it will not only set the best starting architecture, but it will reduce design time as well.

We approach this problem by using TPE [8] as a hyper-parameter optimizer to estimate the optimal depth of CNNs. This approach has the following advantages. Since TPE can memorize all the previous results and implement that information into next selection of parameters, it is efficient in cases where a huge number of iterations and trials are required. In addition, unlike the grid search, the TPE does not require to test out every single choice of hyper-parameters for estimating the best architecture among the given set of choices because it simply skips the choices that are likely to under-perform compared to other choices, which leads to a more timely and efficient method than the grid search. Although the TPE algorithm has been used to optimize the hyper-parameters such as learning rate, drop rate, and etc. [9, 2], in this work, we apply TPE to optimize the depth of CNNs, which is an important factor in designing optimal CNNs architectures. We gave the TPE different choices for the depth of the VGG Net and tested how TPE is capable of estimating the optimal depth for the networks. We picked the VGG Net as our core architecture because it is relatively shallow compared to the state of the art but at the same time delivers decent accuracy, and unlike Res Net, it only has basic CNNs structure, which makes it easier to tune. However, ResNet could also be used as a core architecture because the convolutional blocks can be added to make different choices. We performed our tests on the datasets of CIFAR-10 and CIFAR-100 [10], as they are common benchmarks in machine learning for computer vision.

2 Methodology

The main goal of this research is to test the capability of TPE to estimate the optimal depth (the best choice) of CNNs architecture. The CNNs used in this study is VGG Net [5, 6] because it is constructed only by basic CNNs‘ components making it easier to have multiple choices. The result of gird search was used as the ground truth, and the accuracy and computation time of TPE were compared to the ground truth. No data augmentation was performed to the data sets.

2.1 Choice of VGG Net

VGG Net based CNNs were built with 9 different depths of networks. Their depth range was from 3 layers deep to 15 layers deep. These choices were inspired by the ConvNet Configurations from the paper *Very Deep Convolutional Networks for Large-Sacle Image Recognition* [5], which has shown promise for computer vision problems. Also, they follow the rule that previous convolution blocks should not contain more convolution layers (conv) before a 2×2 Max pooling layer to mimic the ConvNet Configuration. Each convolutional layer has a 3×3 filter followed by Zero Padding and ReLu activation [2]. At the end of each convolution block, there is a 2×2 Max pooling layer. After all of the convolution blocks, fully connected layers (FC) follow. FC is composed of a Flatten layer, the

Dense layer with 256 neurons followed by Relu activation, the Drop out layer with 0.50 dropout rate, and another Dense layer with the 10 neurons followed by softmax activation. These parameters were selected because CIFAR-10 and CIFAR-100 images are relatively small. *Fig. 1* shows the details about each set of the architectures.

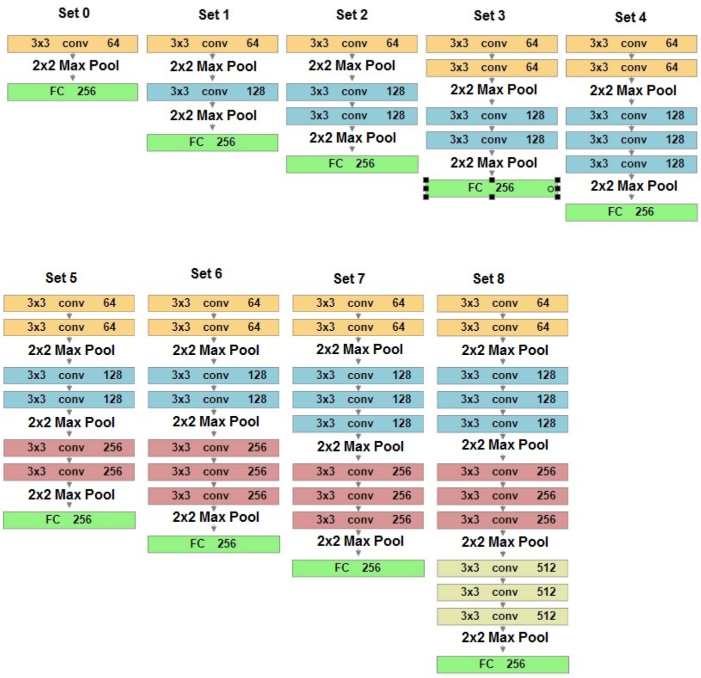

Fig. 1: The details of the VGG architectures. The depth of the architecture increases from Set 0 (3 layers) to Set 8 (15 layers).

2.2 Grid Search

To account for the randomness nature of CNNs [11], each set was tested 3 times and results were averaged, where total number of iterations was 27. The validation accuracies after 50 epochs were recorded. The stochastic gradient descent was used as an optimizer with 0.9 momentum [12] and 0.001 learning rate. Categorical cross entropy was used as the loss function. The total computation time was about 20 hours. *Fig. 2* shows the accuracies of each architecture set with CIFAR-10 dataset.

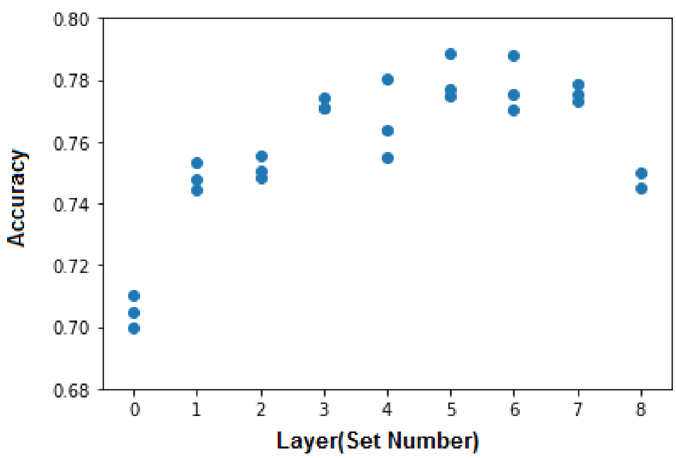

Fig. 2: The accuracy of each set in Cifar-10 dataset. Set 5 and Set 6 have the best accuracy. Due to the randomness of CNNs, each set accuracy has variances. As the networks becomes deeper, CNNs performs better. However, it starts to under-perform due to the over-fitting after the peak; Set 5/6.

When the architecture is shallow, it under-performs because it is not able to capture the features of images effectively. When the architecture is too deep, it also under-performs due to the over-fitting. For CIFAR-10 datset, Sets 5 and 6 were the best performing architectures among the 9 sets. Their best validation accuracies were 78.87% and 78.79%, respectively. The corresponding p-value ($p = 0.78$) from McNemar's Test of the difference of performances between Set 5 and Set 6 indicates there is no statistically significant difference in the performances.

For CIFAR-100 datset, Set 1 has the best performance; accuracy of 42.81%. It produces very low accuracy because CIFAR-100 has 100 classes and each class has only 600 samples (CIFAR-10 has 6000 samples per class.) Therefor, we consider Set 5 and Set 6 as the ground truth for CIFAR-10 and Set 1 as the ground truth for CIFAR-100. If our algorithm can successfully predict best architecture sets determined by the grid search (i.e., Set 5 and Set 6 for CIFAR-10, and Set 1 for CIFAR-100), it validates the effectiveness of TPE as hyper-parameter optimizer. *Fig. 3* shows the accuracies of each set with CIFAR-100 dataset.

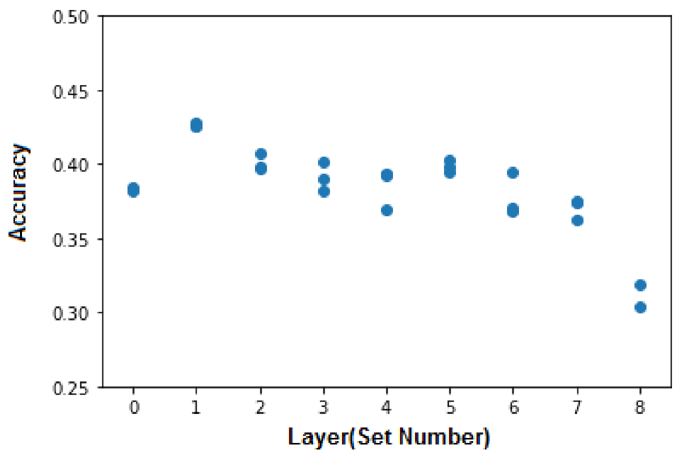

Fig. 3: The accuracy of each set in Cifar-100 dataset. Set 1 has the best accuracy. Due to the randomness of CNNs, each set accuracy has variances. There is the peak performance at relatively shallow architecture (Set 1.) It is because CIFAR-100 has 100 different classes and there is only 600 samples per classes, which lead to over-fit relatively easier that CIFAR-10 dataset.

2.3 Tree-Structured Parzen Estimators

For Each iterations, the TPE collects new observation and defines a prior distribution with given observations, thus, for the first few iterations, the TPE searches the parameter space randomly. When there are enough data to define prior distribution, the TPE divides collected observations into two groups. The first group is the best performing observations which are 10% of total observations, and the second groups is the remaining observations. Next, the TPE finds a set of parameters that are more likely in the first group and less likely in the second group. After this process, the TPE defines the models of likelihood probability for each of the two groups. From the observed parameters, the TPE picks the candidates that are more likely in the first group. The expected improvement per each candidate is defined as the following formula.

$$EI = \frac{l(x)}{g(x)} \tag{1}$$

where EI is the expected improvement per iteration, l(x) is a probability being in the first group and g(x) is a probability being in the second group.

Because of the prior distribution, the TPE does not have to search every simple parameter. It skips testing the parameters that have likely low performance, and hence, the decision of the TPE converges in the early iterations. This means that the TPE will not waste its time to test too deep networks, leading to decreased computation time. In addition, the TPE memorizes all previous results as the prior distribution and can make a decision to choose the best parameter to test next. We set the number of iterations of the TPE to 10. Multiple tests were conducted and the accuracy of estimating the the best number of layers for the VGG Net and the computation time were measured.

3 Results

Despite the fact that there were 9 choices of network depth and only 10 iterations were done, our TPE algorithm was able to estimate the best architecture sets that gives the highest accuracy among given sets. In Cifar-10 dataset, the accuracy of estimating the best sets (Sets 5 and 6) was 83.33% (For Cifar-10, a total of 6 different tests were conducted with iteration of 10). In CIFAR-100 dataset, the accuracy of estimating the best sets was 60.00% (For Cifar-100, a total of 5 different tests were conducted with iteration of 10). Not only were we able to estimate the best architecture sets with reasonable accuracy, we were also able to reduce the computation time by 70%. While the grid search took approximately 20 hours, the TPE algorithm took only 6 hours for CIFAR-10 and 5 hours for CIFAR-100. The TPE with CIFAR-100 took 1 hour less because the optimal set was shallower than the optimal set required for CIFAR-10. However, the limitation of the number of iterations causes the TPE to mis-estimate the architecture set in some cases. In those mis-estimated cases, the TPE did not test the ground truth sets, leading the TPE to build a wrong prior distribution. To overcome the mis-estimation, the number of iterations has to be increased, which will be done as a future work. The direct comparison between grid search and the TPE algorithm in the number of iterations and computation time is shown in *Table 1*.

Table 1: The direct comparison between traditional Gird Search and our TPE algorithm for Cifar-10 and Cifar-100 datasets. Our TPE algorithm requires significantly less iterations to estimate the best set than the Grid Search.

Algorithms	Iterations	Time	Trials	Estimated Sets	Accuracy
Grid Search	27	20 hrs	-	-	-
TPE (Cifar-10)	**10**	**6 hrs**	6	**5,5,6, 6,6**,8	83.33%
TPE (Cifar-100)	**10**	**5 hrs**	5	**1,1,1,** 2,2	60.00%

4 Conclusion

We validated the effectiveness of the TPE algorithm for estimating the optimal depth of VGG Net styled CNNs. Our result showed that the TPE algorithm requires significantly shorter computation time relative to the grid search while retaining estimating the optimal depth of CNNs architecture. We anticipate that the proposed algorithms perform even better with the core architecture that requires deeper networks such as ResNet where its depth could go as deep as 152 layers.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*, 2016.

[4] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

[9] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[10] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[11] Simone Scardapane and Dianhui Wang. Randomness in neural networks: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(2), 2017.

[12] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.