Bayesian Optimization Combined with Incremental Evaluation for Neural Network Architecture Optimization

Martin Wistuba

IBM Research
Dublin, Ireland
martin.wistuba@ibm.com

Abstract. The choice of hyperparameters and the selection of algorithms is a crucial part in machine learning. Bayesian optimization methods and successive halving have been applied successfully to optimize hyperparameters automatically. Therefore, we propose to combine both methods by estimating the initial population of incremental evaluation, our variation of successive halving, by means of Bayesian optimization. We apply the proposed methodology to the challenging problem of optimizing neural network architectures automatically and investigate how state of the art hyperparameter optimization methods perform for this task. In our evaluation of these automatic methods, we are able to achieve human expert performance on the MNIST data set but we are not able to achieve similar good results for the CIFAR-10 data set. However, the automated methods find shallow convolutional neural networks that outperform human crafted shallow neural networks with respect to classification error and training time.

1 Introduction

Deep neural networks are powerful models that are able to extract features from complex data such as images, audio and text automatically. They enabled to improve the state of the art on many different domains such as image classification [10,18,30,39], object detection [8,27], speech recognition [4,23] and machine translation [35]. Rather than manually extracting features from the data, the humans' task changed to create neural network architectures and components which are able to learn how to extract useful features. However, the creation of neural networks requires a lot of domain expertise. Considering the domain of image classification only, hundreds of researchers have created several components and architectures before reaching top performances on all the well-known benchmark data sets. Therefore, an automatic way to find the optimal neural network architecture would be extremely useful.

Snoek et al. proposed to apply Bayesian optimization to tackle the hyperparameter optimization problem automatically [31]. In numerous occasions Bayesian optimization provided better hyperparameter configurations than human data scientists. Snoek et al. pushed the state of the art in deep learning to

new levels on the CIFAR-10 data set in 2012 [31] and again in 2015 for CIFAR-10 and CIFAR-100 [32]. However, a fixed, good performing neural network architecture was chosen and only its hyperparameters were optimized.

Since deep learning is becoming more and more important in various domains, it is not utterly surprising that there are attempts to automate the search for suitable neural network architectures [1, 26, 46]. Reinforcement learning and evolutionary algorithms are currently the dominating methods and are able to achieve state of the art results.

In this paper, we conduct a study to investigate whether the state of the art for hyperparameter optimization can be applied to the task of neural network architecture optimization. We propose incremental evaluation, a variation of successive halving [13], with two different initialization methods: Bayesian optimization and random search. We compare Bayesian optimization, incremental evaluation and random search [3]. We evaluate all approaches on the two popular image classification benchmark data sets MNIST and CIFAR-10.

2 Related Work

Neural network architecture optimization got recently a lot of attention thanks to new automatic optimizers which achieve human level performances. Zoph and Le proposed to use a reinforcement learning method to learn optimal network architectures [46]. They propose to learn a controller based on a recurrent neural network which generates a sequence that encodes a neural network architecture. Real et al. propose the use of evolutionary algorithms [26]. In contrast to the typical operations of evolutionary algorithms, they are not using cross-over but rely on mutations only. Both approaches achieve impressive results on CIFAR-10 but at the cost of high computation time. Zoph and Le use 800 GPUs in parallel and evalute 12,800 different architectures before they reach their best performing neural network architecture. Real et al. use 250 parallel workers, each of them for more than 256 hours.

One of the current state of the art methods for hyperparameter optimization is Bayesian optimization [31]. Various extensions are proposed to consider different important aspects such as new surrogate models [11, 32], initializations [7,42,43], pruning [41] and learning from samples [15], learning curves [6,38] as well as related tasks [2, 28, 37, 45], or addressing particular problems [22, 44]. Recently, successive halving [13] have been proposed for the same task and was further extended to Hyperband [20]. Klein et al. propose a similar extension to Hyperband. Using a special Bayesian neural network for learning curve prediction, they determine the initial population after few random trials [16].

Swersky et al. propose to use a specific kernel for conditional parameter spaces and applies it in Bayesian optimization for neural network optimization [36]. Mendoza et al. also apply Bayesian optimization for this task but replace the surrogate model with a random forest [22].

3 Problem Definition

We define the problem of finding the optimal neural network architecture as a black-box optimization problem. Considering an arbitrary machine learning task, the black-box function f is defined as

$$f \ : \ \mathcal{X} \rightarrow \mathbb{R} \ , \tag{1}$$

which maps a network architecture $\mathbf{x} \in \mathcal{X}$ to its loss on the validation data set D_{valid}. Evaluating the function f at value $\mathbf{x}$ involves training the neural network encoded by $\mathbf{x}$ on the training data set D_{train} and evaluating it on the validation data set D_{valid}. Hence, the network architecture $\mathbf{x}^*$ which minimizes f,

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{X}} f\left(\mathbf{x}\right) \ \ , \tag{2}$$

is the optimal neural network architecture for our machine learning task.

In the following, we will assume that we want to find the optimal network architecture for an image classification task. Hence, we want to find the convolutional neural network architecture which minimizes the classification error.

4 Combining Bayesian Optimization and Incremental Evaluation

Bayesian optimization is an optimization method for black-box functions. It consists of two components, a surrogate model and an acquisition function. The surrogate model tries to approximate the function to be minimized and additionally provides uncertainty estimations about its predictions. The acquisition function uses the prediction of the surrogate model in order to estimate the expected utility of a possible candidate. The candidate which maximizes the expected utility is selected to be evaluated. In our experiments, we use a Gaussian process with Matérn $5/2$ kernel or a Random Forest [11] combined with the expected improvement acquisition function [14]. Kernel hyperparameters are optimized by maximizing the log-likelihood [25]. For more information and related work, we refer the interested reader to the recent review by Shahriari et al. [29].

Successive halving [13] is a simple, however, effective bandit-based approach to optimize hyperparameters for algorithms with iterative learning procedures. It starts with a set of random hyperparameter configurations. Every iteration has the same budget that is uniformly distributed among the remaining candidates in order to continue the learning process. At the end of an iteration the population is reduced by a factor r, keeping only the candidates with the smallest, current loss. Since the budget per iteration is fix, the budget keeps increasing per remaining candidate by a factor of r^{-1}.

We propose incremental evaluation which is motivated by our observations when training neural networks. We observed that it is difficult to distinguish between good and very good architectures after a small training time because

deeper architectures need more time than shallow ones before providing good predictions. Therefore, in contrast to succcessive halving, we propose to spend more budget in the beginning and less later before deciding which candidates to discard. This algorithm is outlined in Algorithm 1. Given a total budget B, we choose a set of candidates based on an initialization strategy using an initial budget b_{init} for each candidate. Then, we keep training all remaining candidates each with a budget of b before reducing the set of candidates by a factor of r. Thus, the budget spent per iteration reduces exponentially. We propose two different initialization strategies: random search and Bayesian optimization. Random search picks the network depth at random and then chooses the hyperparameters of all layers at random. Each candidate network is trained for b_{init} epochs. The initialization with Bayesian optimization is simply running Bayesian optimization but limiting the number of epochs to b_{init}. We set $b_{\text{init}} = 2$, $b = 1$ and the reduction factor to $r = 0.25$.

Algorithm 1 Incremental Evaluation

Input: Budget B, reduction factor r, budget per candidate for the initialization b_{init} and later phase b, current loss of candidate i l_i. The function "initPopulation" denotes the selection strategy for the initial population. Its first argument is the population size and the second the budget spend for each candidate.

Output: Best neural network architecture found.

$P_0 \leftarrow \text{initPopulation}\left(\lfloor B \left(b_{\text{init}} + b\left(\frac{1}{1-r} - r\right)\right)^{-1} \rfloor, b_{\text{init}}\right)$
for $k = 0, 1, \ldots, \lceil \log_{r^{-1}}(|P_0|) \rceil - 1$ **do**
  Let σ_k be a bijection on P_k such that $l_{\sigma_k(1)} \leq \ldots \leq l_{\sigma_k(|P_k|)}$.
  $P_{k+1} \leftarrow \left\{i \in P_k \mid l_{\sigma_k(i)} \leq l_{\sigma_k(\lfloor r|P_k| \rfloor)}\right\}$.
  for $i \in P_{k+1}$ **do**
    Continue training i for a budget of b.
return Remaining element in $P_{\lceil \log_{r^{-1}}(|P_0|) \rceil}$.

5 Experiments

5.1 Search Space

We defined the search space $\mathcal{X}$ of neural network architectures as follows. Each network has between one and five convolutional layers. For each convolutional layer the search algorithm needs to choose a quadratic filter with dimensions between three and ten and the number of filters within the interval $[2, 128]$. The activation functions of the convolutional layers are rectified linear units [24]. Each convolutional layer is followed by a dropout layer [33] with dropout rates between 0 and 0.9. There is an optional batch normalization layer [12] after each convolutional layer. The stride is fixed to one and we use zero padding to ensure the dimensionality of the feature maps is not reduced. Finally, there are optional max pooling layers after each convolutional layers. Again, zero padding is applied but for the max pooling layer after the first and last convolutional

layer in the neural network. All networks are flattened after the convolutional part and a fully connected layer with number of nodes equal to the number of classes is added. We follow the lead by Zoph and Le [46] and do not include the hyperparameters of the optimizer into our search. We use the momentum optimizer with a learning rate of 0.1, weight decay of 10^{-4}, momentum of 0.9 and Nesterov momentum [34] with a batch size of 50. The reason for this decision is that we want to focus on finding the optimal architecture and avoid to increase the search space even more. However, learning these hyperparameters is possible and we will consider it for future work. Each network is trained for up to 20 epochs, early stopping is applied to stop a learning process early in case the loss on the validation set did not improve for two epochs.

Bayesian optimization expects as input a vector of constant size, even if the network depth changes. We used following encoding, similar to the one proposed by Diaz et al. [5]. We force all entries for layers that are unused to zero and add another entry to the vector that encodes the depth of the network.

We conduct experiments on the famous benchmark data sets MNIST [19] and CIFAR-10 [17]. Each optimizer has one hour for the MNIST data set and six hours for the CIFAR-10 data set. We use the original train and test split and create our own train/validation split using the original train split. Our train split is used in order to train the network and the validation split to evaluate the networks performance. We report the mean test error of five repetitions. The test error is estimated with the neural network that performed best on our validation split. We do not use any data augmentation techniques.

5.2 Neural Network Architecture Optimizers

In this work we compare six different neural network architecture optimizers. Random search [3] is selecting neural network architectures at random. It seems to be a simple baseline, however, random search has provided good results, especially in the domain of neural networks where the number of hyperparameters is large but many of them are insensitive to smaller changes. Furthermore, we investigate the performance of the methods we discussed in Section 4: Bayesian optimization with Gaussian process [31] or Random Forest [11], iterative evaluation in combination with random search [13] or Bayesian optimization.

5.3 Neural Network Architecture Search for MNIST

MNIST is a 20 years old digit recognition benchmark data set which has been used by hundreds of researchers to evaluate their image classification methods. The data set contains 60,000 28x28 grayscale images of the 10 digits 0-9, along with a test set of 10,000 images. The results are reported in Table 1. We see no significant differences between all methods. It seems that one GPU hour is already enough to find top performing networks and the solution is probably close to the optimum within our search space. To set our results into context, the best reported result on MNIST is an error of 0.21% [40]. This result is achieved using a neural network architecture which is not an element of our search space. During

the automatic search we found an architecture with only two convolutional layers which is capable of achieving an classification error of 0.59%. This convolutional neural network can be trained in only 150 seconds. This result is comparable to the fast-learning shallow convolutional neural network proposed by McDonnell and Vladusich which achieves in four minutes an error of 0.5% [21]. However, our training time is significantly shorter with a slightly smaller accuracy.

Table 1. Classification error in percent. The network architecture optimization methods can achieve state of the art performance for MNIST. No reasonable solution for CIFAR-10 have been found due to search space and computation time constraints.

Method	MNIST	CIFAR-10
Random Search (RS)	0.896 ± 0.149	29.016 ± 3.600
Bayesian Optimization with Gaussian Process (BO-GP)	0.792 ± 0.153	28.160 ± 2.021
Bayesian Optimization with Random Forest (BO-RF)	0.714 ± 0.050	30.430 ± 4.117
Incremental Evaluation (IE) + RS	0.648 ± 0.124	26.556 ± 2.335
IE + BO-GP	0.730 ± 0.073	27.964 ± 2.332
IE + BO-RF	0.650 ± 0.127	25.650 ± 3.657
Fast-Learning Shallow CNN [21]	0.37	24.14
Best Human Score	0.21 [40]	3.47 [9]

5.4 Neural Network Architecture Search for CIFAR-10

The CIFAR-10 data set is another popular image classification benchmark data set which contains 50,000 32x32 color training images and 10,000 test images. The results achieved by our investigated methods are reported in Table 1. The best result achieved by a human is as low as 3.47% [9], clearly better than the results we are able to achieve with the automatic methods. Actually, results below 10% error are very common in recently published papers for this data set. We identified two reasons for this result. First, our search space is very restrictive, only allowing networks with a depth up to 5 convolutional layers and without skip connections [10]. Second, we limited the search duration to only 6 GPU hours. Since we learn only shallow networks, we draw a comparison to the work of McDonnell and Vladusich [21] again. Their best network achieved an error of 24.14% in less than an hour while our best network achieves an error of 22.81% within less than 25 minutes training time. Thus, our training time is not only by a factor of two shorter but it also provides a smaller classification error.

6 Conclusions

We investigated how well state of the art black-box optimization methods used for hyperparameter optimization perform for the task of neural network architecture search. We can show that they are capable of reaching human level

performance on the MNIST data set. The results on CIFAR-10 are currently less promising but we identified possible problems which we are going to investigate in the future. Comparing the shallow networks found during our search to human crafted shallow neural networks, we see that the automatic search is able to find neural network architectures which need less training time for achieving the same or even better performance.

We proposed incremental evaluation, a variation of succcessive halving, and various initialization methods for it. While this idea is theoretically sound, our experiments do not show any significant improvements.

As future work, we consider to increase the search space significantly in order to create deep architectures. Furthermore, we will increase the search time and use multiple GPUs in parallel. Finally, we want to test our variation of successive halving for the task of hyperparameter optimization in order to see whether a significant improvement for this task can be achieved.

References

1. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. CoRR abs/1611.02167 (2016), http://arxiv.org/abs/1611.02167
2. Bardenet, R., Brendel, M., Kégl, B., Sebag, M.: Collaborative hyperparameter tuning. In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013. pp. 199–207 (2013), http://jmlr.org/proceedings/papers/v28/bardenet13.html
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, 281–305 (2012), http://dl.acm.org/citation.cfm?id=2188395
4. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Trans. Audio, Speech & Language Processing 20(1), 30–42 (2012), https://doi.org/10.1109/TASL.2011.2134090
5. Diaz, G.I., Fokoue, A., Nannicini, G., Samulowitz, H.: An effective algorithm for hyperparameter optimization of neural networks. CoRR abs/1705.08520 (2017), http://arxiv.org/abs/1705.08520
6. Domhan, T., Springenberg, J.T., Hutter, F.: Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. pp. 3460–3468 (2015), http://ijcai.org/Abstract/15/487
7. Feurer, M., Springenberg, J.T., Hutter, F.: Initializing bayesian hyperparameter optimization via meta-learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. pp. 1128–1135 (2015), http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/10029
8. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014. pp. 580–587 (2014), https://doi.org/10.1109/CVPR.2014.81

9. Graham, B.: Fractional max-pooling. CoRR abs/1412.6071 (2014), http://arxiv.org/abs/1412.6071
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 770–778 (2016), https://doi.org/10.1109/CVPR.2016.90
11. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Learning and Intelligent Optimization - 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers. pp. 507–523 (2011), https://doi.org/10.1007/978-3-642-25566-3_40
12. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. pp. 448–456 (2015), http://jmlr.org/proceedings/papers/v37/ioffe15.html
13. Jamieson, K.G., Talwalkar, A.: Non-stochastic best arm identification and hyperparameter optimization. In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016. pp. 240–248 (2016), http://jmlr.org/proceedings/papers/v51/jamieson16.html
14. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Global Optimization 13(4), 455–492 (1998), https://doi.org/10.1023/A:1008306431147
15. Klein, A., Falkner, S., Bartels, S., Hennig, P., Hutter, F.: Fast bayesian optimization of machine learning hyperparameters on large datasets. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA. pp. 528–536 (2017), http://proceedings.mlr.press/v54/klein17a.html
16. Klein, A., Falkner, S., Springenberg, J.T., Hutter, F.: Learning curve prediction with Bayesian neural networks. In: Proceedings of the International Conference on Learning Representations (ICLR) (2017)
17. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep. (2009)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM 60(6), 84–90 (2017), http://doi.acm.org/10.1145/3065386
19. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. pp. 2278–2324 (1998)
20. Li, L., Jamieson, K.G., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. CoRR abs/1603.06560 (2016), http://arxiv.org/abs/1603.06560
21. McDonnell, M.D., Vladusich, T.: Enhanced image classification with a fast-learning shallow convolutional neural network. In: 2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015. pp. 1–7 (2015), https://doi.org/10.1109/IJCNN.2015.7280796
22. Mendoza, H., Klein, A., Feurer, M., Springenberg, J.T., Hutter, F.: Towards automatically-tuned neural networks. In: Proceedings of the 2016 Workshop on Automatic Machine Learning, AutoML 2016, co-located with 33rd International Conference on Machine Learning (ICML 2016), New York City, NY, USA, June 24, 2016. pp. 58–65 (2016), http://jmlr.org/proceedings/papers/v64/mendoza_towards_2016.html

23. Mohamed, A., Dahl, G.E., Hinton, G.E.: Acoustic modeling using deep belief networks. IEEE Trans. Audio, Speech & Language Processing 20(1), 14–22 (2012), https://doi.org/10.1109/TASL.2011.2109382
24. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel. pp. 807–814 (2010), http://www.icml2010.org/papers/432.pdf
25. Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning. Adaptive computation and machine learning, MIT Press (2006), http://www.worldcat.org/oclc/61285753
26. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Le, Q.V., Kurakin, A.: Large-scale evolution of image classifiers. CoRR abs/1703.01041 (2017), http://arxiv.org/abs/1703.01041
27. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. 39(6), 1137–1149 (2017), https://doi.org/10.1109/TPAMI.2016.2577031
28. Schilling, N., Wistuba, M., Drumond, L., Schmidt-Thieme, L.: Hyperparameter optimization with factorized multilayer perceptrons. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part II. pp. 87–103 (2015), https://doi.org/10.1007/978-3-319-23525-7_6
29. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N.: Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE 104(1), 148–175 (2016), https://doi.org/10.1109/JPROC.2015.2494218
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014), http://arxiv.org/abs/1409.1556
31. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. pp. 2960–2968 (2012), http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms
32. Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M.M.A., Prabhat, Adams, R.P.: Scalable bayesian optimization using deep neural networks. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. pp. 2171–2180 (2015), http://jmlr.org/proceedings/papers/v37/snoek15.html
33. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014), http://dl.acm.org/citation.cfm?id=2670313
34. Sutskever, I., Martens, J., Dahl, G.E., Hinton, G.E.: On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013. pp. 1139–1147 (2013), http://jmlr.org/proceedings/papers/v28/sutskever13.html
35. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada. pp. 3104–3112 (2014), http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks

36. Swersky, K., Duvenaud, D., Snoek, J., Hutter, F., Osborne, M.A.: Raiders of the lost architecture: Kernels for bayesian optimization in conditional parameter spaces (2014)
37. Swersky, K., Snoek, J., Adams, R.P.: Multi-task bayesian optimization. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. pp. 2004–2012 (2013), http://papers.nips.cc/paper/5086-multi-task-bayesian-optimization
38. Swersky, K., Snoek, J., Adams, R.P.: Freeze-thaw bayesian optimization. CoRR abs/1406.3896 (2014), http://arxiv.org/abs/1406.3896
39. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. pp. 1–9 (2015), https://doi.org/10.1109/CVPR.2015.7298594
40. Wan, L., Zeiler, M.D., Zhang, S., LeCun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013. pp. 1058–1066 (2013), http://jmlr.org/proceedings/papers/v28/wan13.html
41. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Hyperparameter search space pruning - A new component for sequential model-based hyperparameter optimization. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part II. pp. 104–119 (2015), https://doi.org/10.1007/978-3-319-23525-7_7
42. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Learning data set similarities for hyperparameter optimization initializations. In: Proceedings of the 2015 International Workshop on Meta-Learning and Algorithm Selection co-located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2015 (ECMLPKDD 2015), Porto, Portugal, September 7th, 2015. pp. 15–26 (2015), http://ceur-ws.org/Vol-1455/paper-04.pdf
43. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Learning hyperparameter optimization initializations. In: 2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Campus des Cordeliers, Paris, France, October 19-21, 2015. pp. 1–10 (2015), https://doi.org/10.1109/DSAA.2015.7344817
44. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Automatic frankensteining: Creating complex ensembles autonomously. In: Proceedings of the 2017 SIAM International Conference on Data Mining. pp. 741–749. Society for Industrial and Applied Mathematics (2017)
45. Yogatama, D., Mann, G.: Efficient transfer learning method for automatic hyperparameter tuning. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014. pp. 1077–1085 (2014), http://jmlr.org/proceedings/papers/v33/yogatama14.html
46. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. CoRR abs/1611.01578 (2016), http://arxiv.org/abs/1611.01578