Learning to Rank Learning Curves

— Supplementary Material —

Martin Wistuba [1] **Tejaswini Pedapati** [1]

1. Ablation Study: Learning Curve Component

One of the most important design decisions of LCRankNet was the learning curve component. We experimented with several models that would generate the best learning curve embedding. In the following discussion of different models, we assume that the learning curve represents the validation accuracy over time and thus higher values are better.

To begin with, passing the entire learning curve directly to be concatenated with architecture embedding would result in large number of predictors thereby overfitting. It was also clear that the best or last value of the learning curve alone (assuming that the learning curve is constantly improving, i.e. it increases monotonically, it is the same) is a very good predictor. After all, it is the only one used by methods like Hyperband and Successive Halving. In our example, this would be achieved through a simple global max pooling layer. Any further information regarding the development of the learning curve (improvement since epoch 1, 1st and 2nd order gradients, etc.) would, however, be disregarded. Feature engineering would be one way to create such features but convolutions allow to automatically learn which of these features are helpful. In Figure 1 we take two different models with max pooling layer into account. The version with global max pooling layer reduces the number of predictors to one per filter, while the version with strides reduces the number to 4 per filter. In our experiments, we did not notice a big difference between these two versions, but a significant improvement over the version that only uses the best value.

Furthermore, we tried in vain to apply LSTMs to this problem. But both, LSTMs directly using the learning curve and using the output of the convolution did not achieve better results than if the learning curve had not been taken into account at all ("architecture only").

[1]IBM Research. Correspondence to: Martin Wistuba <martin.wistuba@ibm.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

2. Joint Architecture and Hyperparameter Optimization

As briefly discussed in the main paper, we want to show that LCRankNet

- can be combined with optimization methods such as Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011), Regularized Evolution (RE) (Real et al., 2019) and Reinforcement Learning (RL) (Zoph & Le, 2017),
- can be applied to different search spaces (convolutional vs. fully connected neural networks) with and without hyperparameters, and
- work with different machine learning tasks (classification vs. regression).

For this reason we are carrying out an additional experiment on a publicly available tabular regression benchmark (Klein & Hutter, 2019). This benchmark was created by performing a grid search with a total of 62,208 different architecture and hyperparameter settings for four different tabular regression datasets: slice localization, protein structure, parkinsons telemonitoring, and naval propulsion. Each dataset is split into 60% train, 20% validation, and 20% test. Each architecture has two fully connected layers. Settings vary with respect to the initial learning rate (0.0005, 0.001, 0.005, 0.01, 0.05, 0.1), the batch size (8, 16, 32, 64), the learning rate schedule (cosine or fixed), the activation per layer (ReLU or tanh), the number of units per layer (16, 32, 64, 128, 256, 512), and dropout per layer (0.0, 0.3, 0.6).

We are again following the leave-one-dataset-out cross-validation protocol: We optimize the architecture and hyperparameter setting for one dataset and transfer the knowledge from the others. 100 settings per dataset are randomly selected as additional data for LCRankNet.

2.1. Accelerating Hyperparameter Optimization

The goal is to minimize the mean squared error (MSE) by choosing the right architecture and hyperparameter settings. In this experiment we compare the optimization methods TPE, RE and RL with a version with early termination using

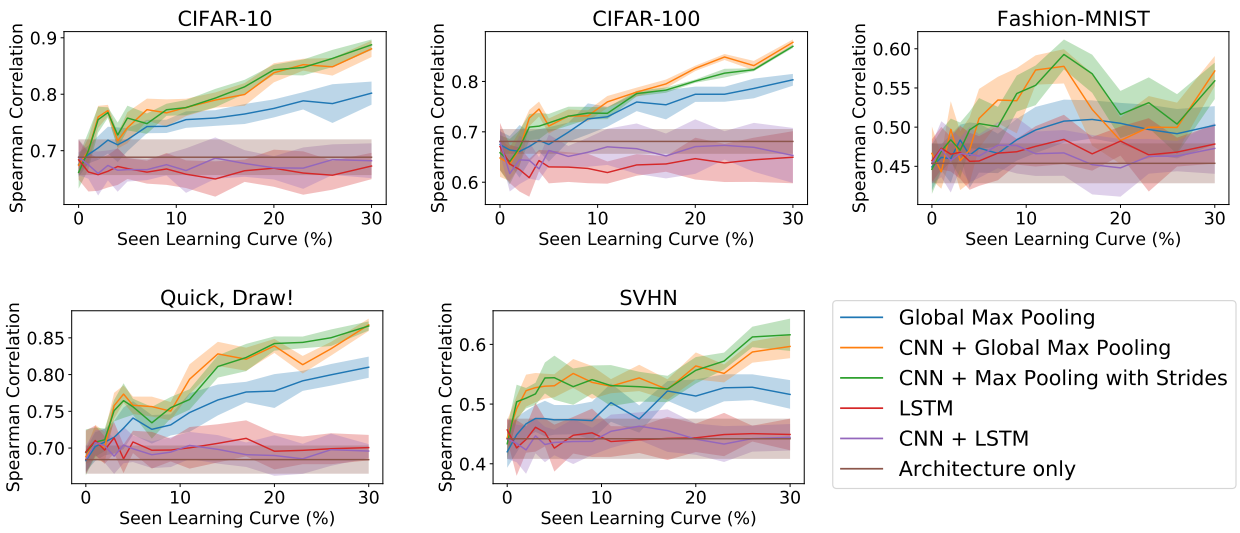

Figure 1. Analysis of various different learning curve components in comparison to not using the learning curve at all.

LCRankNet. All experiments are repeated ten times. Each optimization method can take up to 100 different settings into account. We report results with respect to test regret following Klein & Hutter (2019). This number indicates the difference between test MSE actually achieved and best possible test MSE. Finding the best setting will result in a regret of 0. The setting chosen by an optimization method is based on the validation MSE. It is therefore possible that the test regret may increase again. This can be seen as an overfitting on the validation set. In Figure 3 to 5 we report the results up to the search time required for the shortest of all repetitions. We find that early termination generally improves the method. Only in one case are they not better, but not worse either.

2.2. Technical Details

In contrast to random search, Successive Halving or Hyperband, the intermediate performance is not sufficient for other optimization methods and the final performance of a model is required. To take this into account, a simple change to LCRankNet is required. An additional output layer is added that predicts the final performance in additional to the ranking score. We continue to use the ranking score only to decide whether to terminate a run early or not. The predicted final performance is only used in the event of early termination and is only used to provide feedback to the optimization method. If a run is not terminated early, the actual final performance is used. To ensure that the predicted final performance is in a reasonable range, we define a lower and upper bound. We are now explaining them in the context of MSE so lower values are better and vice versa. The lower bound is defined by the mean MSE observed for previous runs. The motivation for this decision is that each terminated run should be below average. We define the best observed MSE of the partial learning curve as an upper bound. If the upper and lower bounds conflict, we prefer the upper bound because it is based directly on observed data.

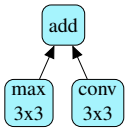

Figure 2. One example block used in an architecture from the NASNet search space.

3. Architecture Representation

In this work we consider two different search spaces. The NASNet search space that takes CNNs (Zoph et al., 2018) into account and a search space for FCNNs (Klein & Hutter, 2019). In the NASNet search space, an architecture is completely described by two cells, each consisting of several blocks. The entire architecture is defined by selecting the design of all blocks. A block is designed by selecting two operations and their corresponding inputs and how these two operations are combined. A sample block is shown in Figure 2. We follow Luo et al. (2018) and model every combination of input and operation through three embeddings. The first embedding specifies the input choice, the second the type of operation (convolution, maximum pooling, etc.), and the third the kernel size. In this way, an architecture with two cells, each with five blocks, is clearly described by

a sequence of 60 decisions.

We describe the FCNNs in a very similar way. For each layer we learn an embedding for the activation function, the dropout rate and the number of units. The batch size, the learning rate (after log transformation) and the learning rate schedule (after one-hot encoding) are taken into account as other hyperparameters.

References

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pp. 2546–2554, 2011. URL http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimiza

Klein, A. and Hutter, F. Tabular benchmarks for joint architecture and hyperparameter optimization. *CoRR*, abs/1905.04970, 2019. URL http://arxiv.org/abs/1905.04970.

Luo, R., Tian, F., Qin, T., Chen, E., and Liu, T. Neural architecture optimization. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 7827–7838, 2018. URL http://papers.nips.cc/paper/8007-neural-architecture-optimization.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 4780–4789, 2019. doi: 10.1609/aaai.v33i01.33014780. URL https://doi.org/10.1609/aaai.v33i01.33014780.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL https://openreview.net/forum?id=r1Ue8Hcxg.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 8697–8710,

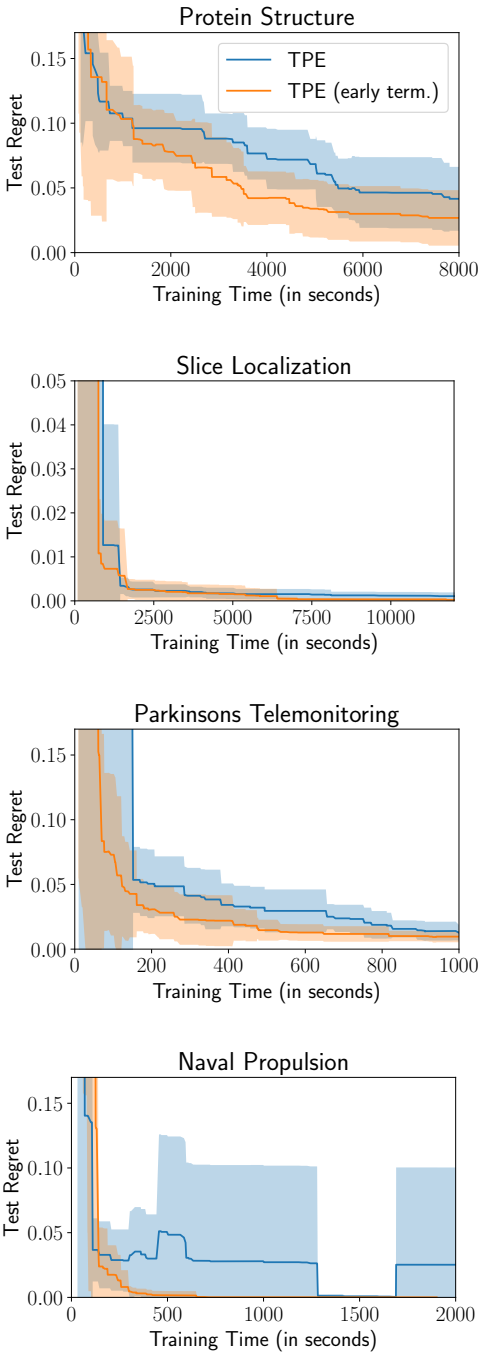

Figure 3. TPE benefits from early termination on all datasets.

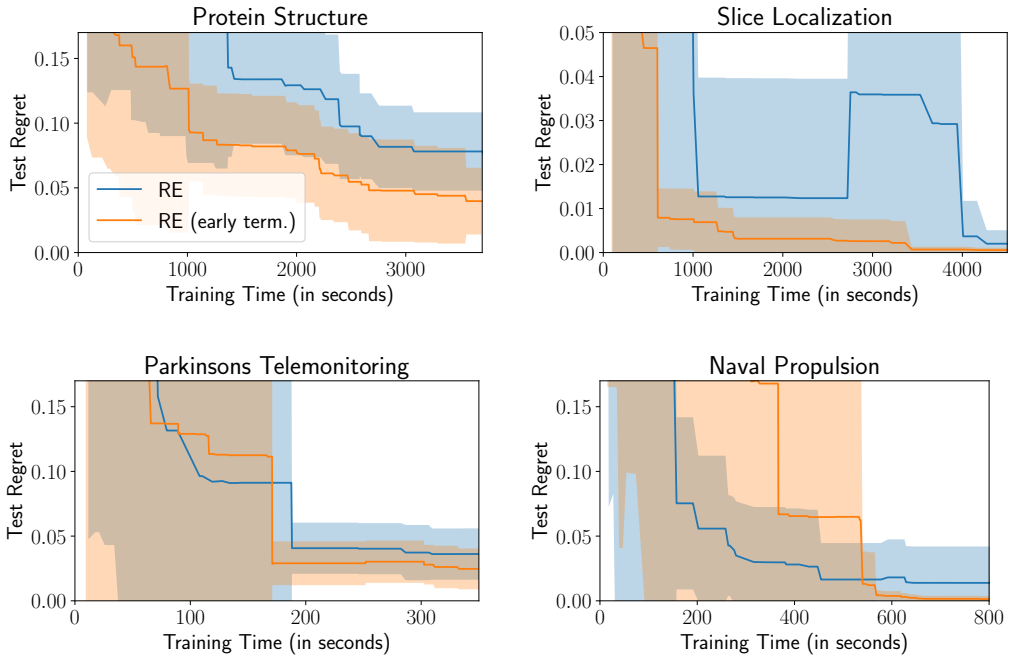

Figure 4. Regularized Evolution benefits from early termination on all datasets.

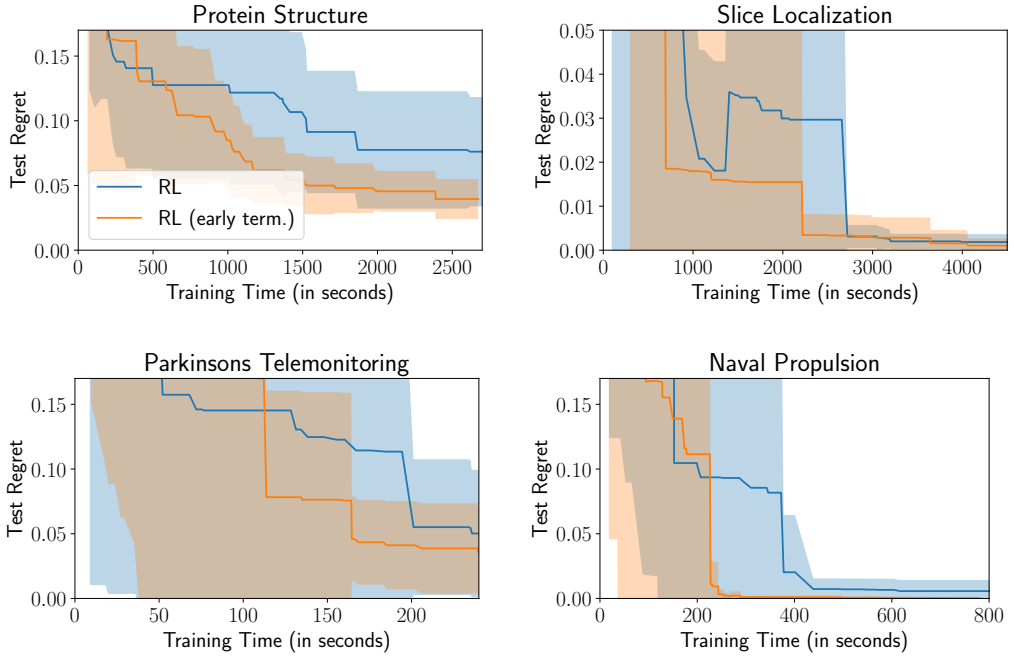

Figure 5. Reinforcement Learning benefits from early termination on all datasets.

2018. doi: 10.1109/CVPR.2018.00907. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Zoph_Learning_Transferable_Architectures_CVPR_2018_paper.html.