Neural architecture search for tabular DNN

Alexander Yukhimchuk [1] **Alexander Khrapko** [1] **Alexander Palanevich** [1]

Abstract

Recently, deep learning methods outperformed traditional boosting-based methods on tabular data (Gorishniy et al., 2021). However, the improvement over boosting-based methods is small and not stable. The idea of this project is to make the gap wider by using techniques of neural architecture search. We use DARTS (Liu et al., 2018) to evaluate MLP models and provide the achieved metrics and architectures.

Github repo: https://github.com/Sisha3342/NASTTDNN

1. Introduction

1.1. Tabular DNNs

Due to the tremendous success of deep learning on such data domains as images, audio and texts, there has been a lot of research interest to extend this success to problems with data stored in tabular format. In these problems, data points are represented as vectors of heterogeneous features, which is typical for industrial applications and ML competitions, where neural networks have a strong non-deep competitor in the form of GBDT (Chen & Guestrin, 2016). Along with potentially higher performance, using deep learning for tabular data is appealing as it would allow constructing multi-modal pipelines for problems, where only one part of the input is tabular, and other parts include images, audio and other DL-friendly data. Such pipelines can then be trained end-to-end by gradient optimization for all modalities. For these reasons, a large number of DL solutions were recently proposed, and new models continue to emerge. Gorishniy et al. (2021) evaluate the main models for tabular DL on a diverse set of tasks to investigate their relative performance. They also demonstrate simple ResNet-like/MLP baselines and introduce FT-transformer - an adaptation of the Transformer architecture for tabular data.

[1]Yandex School of Data Analysis, Moscow, Russia.

1.2. Neural architecture search and DARTS

Discovering state-of-the-art neural network architectures requires substantial effort of human experts. Recently, there has been a growing interest in developing algorithmic solutions to automate the manual process of architecture design. But usually architecture search algorithms are computationally demanding despite their remarkable performance. An inherent cause of inefficiency for so-called sample-based approaches (White et al., 2023) is the fact that architecture search is treated as a black-box optimization problem over a discrete domain, which leads to a large number of architecture evaluations required. Liu et al. (2018) approach the problem from a different angle, and propose a method for efficient architecture search called DARTS (Differentiable ARchiTecture Search). Instead of searching over a discrete set of candidate architectures, they relax the search space to be continuous, so that the architecture can be optimized with respect to its validation set performance by gradient descent. The data efficiency of gradient-based optimization, as opposed to inefficient black-box search, allows DARTS to achieve competitive performance with the state of the art using orders of magnitude less computation resources.

1.3. Contributions

The goal of the project is to apply NAS to DNNs specified for tabular data. For this specific project we use MLP model. We summarize our contributions as follows:

- We implement MLP model for tabular data and create a search space to apply DARTS.
- We train the model on California housing (Pace & Barry, 1997), Jannis (Guyon et al., 2019) and Covertype (Blackard & Dean, 1999) datasets.
- We provide the best architectures found and compare their performance with the original paper (Gorishniy et al., 2021) results.

2. Related work

2.1. MLP architecture and search space

We formalize the "MLP" architecture in the following way (Gorishniy et al., 2021):

$$\begin{aligned} \texttt{MLP}(x) &= \texttt{Linear}(\texttt{Block}(\ldots(\texttt{Block}(x)))) \\ \texttt{Block}(x) &= \texttt{Dropout}(\texttt{Act}(\texttt{Linear}(x))) \end{aligned} \tag{1}$$

The search space we choose for the model can be described with 4 parameters:

- *n_blocks* - the number of `Block` layers in the model, which is selected in the $[1, 10]$. There is also one additional `Block` in the beginning to project model's inputs to the right dimension.
- *d_block* - hidden dimension for `Linear` layers (except the final layer which projects embeddings to the needed space). We use $[32, 64, 128, 256, 512]$ grid.
- *in_act* - activation function used in `Act` layer. We choose one from ReLU, GELU, SiLU (Hendrycks & Gimpel, 2016).
- *blocks_act* - activations function used in all `Block` layers. The same choise as for *in_act*.

2.2. Benchmark datasets and preprocessing

To evaluate models' performance we use three datasets:

- California housing (CA) (Pace & Barry, 1997). This is a regression task for 20640 samples in total with 8 numerical features each. We apply standardization (mean subtraction and scaling) to the dataset's targets and use the quantile transformation from the Scikit-learn library (Pedregosa et al., 2011) for the features.
- Jannis (JA) (Guyon et al., 2019), classification with 83733 samples, 54 numerical features and 4 classes. We use the same quantile transformation from above to the dataset's features.
- Covertype (CO) (Blackard & Dean, 1999), classification with 581012 samples, 54 numerical features and 7 classes. The same preprocessing as for the JA dataset.

We split each dataset into train-val-test in the following manner:

1. Use 4:1 split to generate trainval/test splits.
2. Apply 4:1 split to the trainval dataset to obtain train/val splits.

3. Experiments and results

For both datasets (2.2) we use AdamW optimizer (Loshchilov & Hutter, 2017) to train with learning rate 3×10^{-4} and weight decay 10^{-5}. For CA/JA datasets we apply DARTS for 100 iterations and train the best architecture found for 150 epochs with batch size 256. We use MSE loss and report both MSE/RMSE metrics. For the CO dataset we use 200 iterations for DARTS and 200 epochs for the final training with batch size 1024. Cross-Entropy is used as a loss and accuracy (ACC) as a reported metric.

3.1. Architectures found

The best architectures are represented in the Table 1. We notice that the architectures are compact (only 1-3 inner layers with hidden dimension 32). It's also expected the the architecture for the CO dataset is deeper, because the dataset itself has much more samples and features.

	CA	JA	CO
n_blocks	2	1	3
d_block	32	32	32
in_act	GELU	ReLU	ReLU
blocks_act	ReLU	ReLU	ReLU

Table 1. Best MLP architectures

3.2. Metrics

We compare metrics obtained from the models trained according to 3.1 parameters list with metrics provided by Gorishniy et al. (2021) for their MLP model. For the CA dataset we report RMSE and for JA/CO datasets we report accuracy. The results are in the Table 2. We see that for the CA dataset we get a small performance improvement. For the JA dataset out results have a slight performance decrease. But for the CO dataset the accuracy drops tremendously. Our thoughts on potential improvements and reasons for this behaviour are in the section 4.

	Ours	Original
CA $\downarrow$	**0.478**	0.499
JA $\uparrow$	0.711	**0.719**
CO $\uparrow$	0.829	**0.962**

Table 2. Metrics comparisons. Notation: $\downarrow \sim$ RMSE, $\uparrow \sim$ accuracy

4. Conclusion

During the project work we studied some popular neural networks for tabular data (Gorishniy et al., 2021). We tried to improve them with neural architecture search using DARTS (Liu et al., 2018) algorithm. For example, for California housing (Pace & Barry, 1997) dataset we get a small RMSE improvement, but for Covertype (Blackard & Dean, 1999) dataset we get a huge accuracy drop. We think that this might be fixed with a more accurate search space selection.

As we see in the Table 1, architectures don't differ much, while Covertype dataset has much more samples/features. Our guess is that increasing possible depth or blocks sizes can help to find a better architecture. Also, because the network depth might increase, it's reasonable to consider adding skip-connections to the search space (wether to add them or not will be decided by DARTS). Changing dropout rate carefully can bring some perfromance boost, too.

References

Blackard, J. A. and Dean, D. J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. Revisiting deep learning models for tabular data. In *NeurIPS*, 2021.

Guyon, I., Sun-Hosoya, L., Boullé, M., Escalante, H. J., Escalera, S., Liu, Z., Jajetic, D., Ray, B., Saeed, M., Sebag, M., Statnikov, A., Tu, W.-W., and Viegas, E. *Analysis of the AutoML Challenge Series 2015–2018*, pp. 177–219. Springer International Publishing, Cham, 2019. ISBN 978-3-030-05318-5. doi: 10.1007/978-3-030-05318-5_10. URL https://doi.org/10.1007/978-3-030-05318-5_10.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Pace, R. K. and Barry, R. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

White, C., Safari, M., Sukthanker, R., Ru, B., Elsken, T., Zela, A., Dey, D., and Hutter, F. Neural architecture search: Insights from 1000 papers. *arXiv preprint arXiv:2301.08727*, 2023.