# Supplementary - ABLATOR: Robust Horizontal-Scaling of Machine Learning Ablation Experiments

**Iordanis Fostiropoulos**[1] **Laurent Itti**[1]

[1]University of Southern California, Los Angeles California

| Framework | HPO | Configuration | Training | Tuning | Analysis |
|---|---|---|---|---|---|
| Ray | ✓ | ✗ | ✗ | ✓ | ✗ |
| Lighting | ✗ | ✗ | ✓ | ✗ | ✗ |
| Optuna | ✓ | ✗ | ✗ | ✗ | ✓ |
| Hydra | ✗ | ✓ | ✗ | ✗ | ✗ |
| **ABLATOR** | **✓** | **✓** | **✓** | **✓** | **✓** |

Table 1: Popular Machine Learning frameworks are used for different experiment stages such as hyperparameter selection (‘HPO’), removing boilerplate code for configuring experiments (‘Configuration’), removing boiler plate code for running experiments at scale (‘Tuning’) and performing analysis on the hyperparameter selection (‘Analysis’). `ABLATOR` is the only framework that addresses the entire life-cycle of an ablation experiment.

## A Comparison

In table 1 we compare `ABLATOR` with popular machine learning frameworks Ray [26], Optuna[2], Lighting[9], and Hydra[34] as they can be used at different stages of an experiment. Practitioners would need to write different code during prototyping i.e. with ‘Lighting’, and make changes to their code for scaling the experiment with i.e. ‘Ray’. Current practices are error-prone and cumbersome as we discuss in RQ2 of the main text and appendix C.1. `ABLATOR` is the only framework that can address all stages of the experiment and thus fully supports automation.

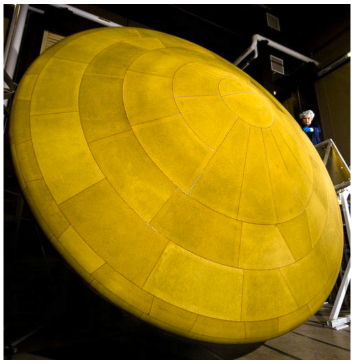


Figure 1: Ablators are materials that are depleted during operation [19]. An experimental `ABLATOR` should not interfere with the experimental result.

Tools like AutoPytorch [38], AutoGluon[8], AutoSkLearn[11], H2O AutoML[22], SMAC3[23] support limited use-cases. The number of ablating models is limited by design [22, 8], or execution is limited to statistical models [11], when neural networks are supported [23] there is no way to manage GPU resources [38]. None of the tools support experiment persistence that is important for Machine Learning experiments, that can take significant amount of time and the execution can be interrupted. `ABLATOR` is the only tool that can address complex use-cases, such as training of Reinforcement Learning algorithms, removes boilerplate code for training and scaling of experiments and allows practitioners to focus on prototyping.

## B Experimental Setup

For our experiments, the search space is defined in fig. 3. For all our results we set a cut-off for performance where we exclude trials which perform close to random. Our motivation is based on

```python
class MyModelWrapper(ModelWrapper):

    def config_parser(self, config: RunConfig):
        config.model_config.d_out = self.dataset.d_out
        return config

config = RunConfig.load("config.yaml")

model = MyModelWrapper(model_class=Transformer)
# Prototyping
trainer = ProtoTrainer(model=model,run_config=config)
# Distributed Execution
trainer = ParallelTrainer(model=model,run_config=config)
trainer.launch()
```

```python
class Transformer(nn.Module):
    def __init__(self, config: ModelConfig):
        super().__init__()
        self.residual = config.residual

    def forward(self, x: Tensor) -> Tensor:
        for layer in self.layers:
            x_prime = layer(x)
            if self.residual:
                x_prime = x_prime + x
            x = x_prime
        return x
```

Figure 2: ABLATOR illustration of the implementation used for our experiments. On the **left** is the code specific to the ablation experiment where we use a `ProtoTrainer` with built-in training mechanisms. While the `ParallelTrainer` is used to scale the `model` to a large cluster of nodes. On the right it is the PyTorch model from the official implementation of FT-T [14] that uses the configuration from figure 2 in the main text. It required minimal changes to `Transformer` to evaluate our hypothesis. We provide analysis in RQ1 of the main text.

the post-randomization exclusion strategy following similar work in clinical trials [10]. We set the exclusion performance cut-off as the performance of the model to the dataset on a pre-defined fixed value. We determined the value empirically where we train a model and evaluate the performance in the validation set after a few training steps i.e. close to random. The 'Cut-Off Perf.' for each dataset are in table 2 and are applied to the accuracy score for the classification data set and RMSE for the regression data set ('Regr.'). The cut-off is only applicable for analysis on the mean performance and figures, where the nonconvergent solutions are excluded. The hyperparameters of our experiment are in fig. 3. We base the choice of our hyperparameters on the defaults of [14].

```yaml
# Train Configurations
train_config.optimizer_config.name: ["adam", "adamw","adabelief","radam","sgd"]
train_config.optimizer_config.lr: [0.01,0.0001]
train_config.epochs: [5,20]
# Dataset Specific
train_config.dataset: ["year","yahoo","helena","covtype","epsilon","jannis","adult","aloi","higgs_small","microsoft","
    california_housing"]
train_config.normalization: ["standard", "quantile"]
train_config.cat_nan_policy: ["new", "most_frequent"]
train_config.cat_policy: ["ohe", "indices", "counter"]
train_config.cat_min_frequency: [0,0.2]
train_config.dataset_seed: [0, 100]
# Model Configurations
model_config.token_bias: [True, False]
model_config.n_layers: [1, 10]
model_config.d_token: [8, 128]
model_config.n_heads: [1, 12]
model_config.d_ffn_factor: [1, 5]
model_config.attention_dropout: [0, 0.3]
model_config.ffn_dropout: [0, 0.3]
model_config.residual_dropout: [0,0.3]
model_config.prenormalization: [True, False]
model_config.initialization: ["xavier", "kaiming"]
model_config.activation: ["relu","gelu","geglu","reglu","leaky_relu","sigmoid"]
model_config.residual: [True, False]
model_config.mask_type: ["mix","global","full","random"]
model_config.random_mask_alpha: [0.5, 1]
```

Figure 3: We present the hyper-parameters we use for our experiment in the `yaml` format provided by ABLATOR. Numerical attributes are sampled from an interval and can be discrete i.e. `model_config.n_layers` or floating point i.e. `model_config.ffn_dropout`. The configuration is parsed with strict type-checking by ABLATOR configuration library. We discuss details of each hyper-parameter in appendix B

### B.1 Train Configuration

In this section we list the parameters that we ablate that are specific to the training procedure and preprocessing of the dataset.

**Hyper-Parameters**; We vary the number of epochs (`epochs`), optimization algorithm (`optimizer_config.name`) (‘Adam’ [20], ‘AdamW’ [25], ‘AdaB’ [37], ‘RAdam’ [24] and ‘SGD’ [30]) and learning rate (`lr`).

**Normalization** (`normalization`); For pre-processing of raw data set features, we use transformations provided by scikit-learn [28]. We evaluate a ‘quantile’[1] transformation, where the features are discretized and mapped to a normal distribution and ‘standard’ [2] where the features are standardized by their mean and variance from the train set.

**Imputation policy** (`cat_nan_policy`); For imputing missing values, we use two strategies, imputing by defining a new categorical token or replacing with the most frequent value.

**Categorical Policy**; We use different methods to provide the categorical features to the model, such as providing them by their indices, one-hot encoding them or using a ‘LeaveOneOut’[3] encoding.

**Rare Categorical Policy** (`cat_min_frequency`); For rare categorical values, such that they appear with a probability less than `cat_min_frequency` of the train data, we replace them with a special category.

**Random Seed** (`random_seed`); we randomly sample a seed that, in turn, is used for the initialization of the training. Using different seed allows us to test identical configurations under different initial training conditions.

### B.2 Model Configuration

In this section we list the parameters that we ablate that are specific to the model definition.

**Token Bias** (`token_bias`) is the learnable bias term added to the token embedding computed from the features that are used as input to the model.

**Model Capacity**; We control the number of parameters to the model with the number of layers (`n_layers`), dimensionality of the token embedding `d_token`, number of heads (`n_heads`), and the expansion factor of the hidden dimension (`d_fnn_factor`).

**Dropout**; We experiment with dropout on the residual connection (`residual_dropout`), the feed-forward network (FFN) output before the residual connection (`fnn_dropout`) and the output of the attention mechanism (`attention_dropout`).

**Prenormalization** (`prenormalization`) refers to applying layer normalization before and after the residual connection.

**Weight Initialization** (`initialization`); we ablate two different weight initialization algorithms for Tablator; `xavier` [13] and `kaiming` [16].

**Activation** (`activation`); we ablate 6 different intermediate activation functions: RELU [1], GELU[18], GEGLU [31], REGLU [31], Leaky RELU [33] and Sigmoid.

**Residual** (`residual`); we ablate whether the residual connections [17] on a transformer model improve performance, where we remove them when `False`.

#### B.2.1 Attention Mask.

We experiment with 4 types of masks. ‘Full’, ‘Global’, ‘Random’ and ‘Mix’ that are inspired by BigBird [35] and evaluated on Tablator. We define $\alpha$ as a hyperparameter that corresponds to the random probability of a token attending another token. A new random mask is calculated on each training iteration with the same $\alpha$. A random mask is similar to dropout applied in the context of the attention mask. Global attention evaluates whether attention is effective for ‘Tablator’. Illustrations of masks are in fig. 4 and results on the effectiveness of mask are in appendix C.2.

---

[1] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.quantile_transform.html

[2] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[3] https://contrib.scikit-learn.org/category_encoders/leaveoneout.html

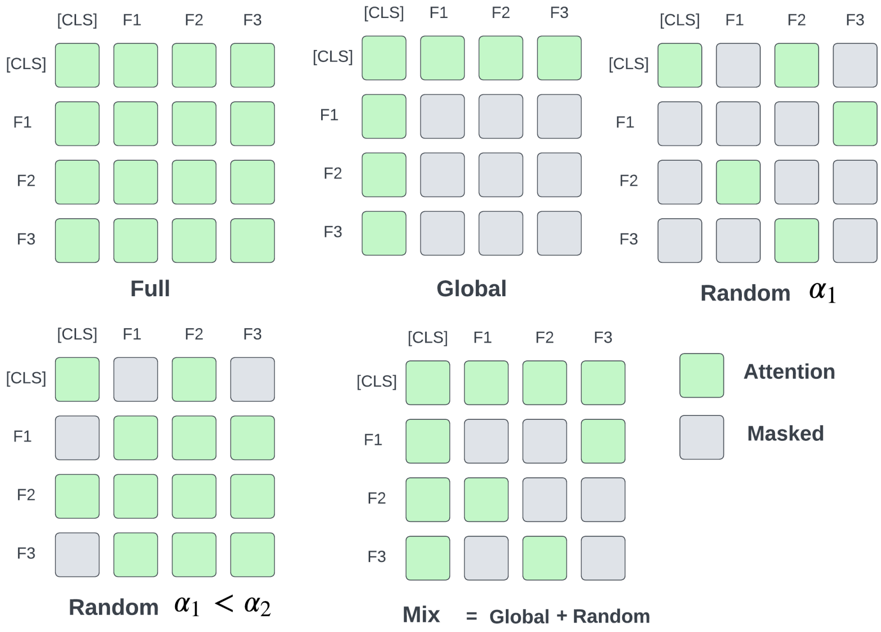


Figure 4: We use 4 type of masks for our experiments with Tablator; 'Full', 'Global', 'Random' and 'Mix'. `F1` through `F3` correspond to the hidden embedding for features 1 through 3. While `[CLS]` is a special token auxiliary token to a downstream task that prepended to the features; a technique used for Transformer model [32]. We present the attention matrix where for each row the features are *attending* to the column features. 'Masked' tokens are prevented from attending or being attended to, and thereby their context is not used in computing of the attention mechanism in Transformer model. Mask sparsity can improve performance in some cases [35].

### B.3 Dataset and Code

We use the code and the preprocessed dataset provided by FT-Transformers [14] official implementation [4] and released under the MIT copyright license[5]. The datasets with their corresponding attributes are listed on table 2.

### B.4 Experiment Execution

For our experiments of **RQ3** in the main text we use a GPU cluster of 3 nodes. We use a total of 24 x A100 GPU and we set a cutoff time limit for the execution of our experiments of 24 hours. Each node has 680 GB of memory; 96 virtual CPUs. For the experiments of **RQ2** we use a single node and train on a subset of datasets; 'CO' and 'MI' as they were the largest for classification and regression, respectively.

## C Additional Results

In this section, we present additional results to the Research Questions of the main text.

### C.1 RQ2

We summarize the recommendations on the most common errors in ML research in table 4. In table 3 we present detailed results on sampling strategy bias.

---

[4] https://github.com/Yura52/tabular-dl-revisiting-models

[5] https://opensource.org/license/mit/

| Name | Cont. | Cat. | Train | Val. | N. Classes | Cut-Off |
|---|---|---|---|---|---|---|
| | Features | | Size | | | Perf. |
| Year (‘YE’) [5] | 90 | - | 370972 | 92743 | Regr. | 10.5 |
| Yahoo (‘YA’) [7] | 699 | - | 473134 | 71083 | | 0.95 |
| Microsoft (‘MI’) [29] | 136 | - | 723412 | 235259 | | 0.82 |
| Housing (‘CA’) [27] | 8 | - | 13209 | 3303 | | 1.1 |
| Aloi (‘AL’) [12] | 128 | - | 69120 | 17280 | 1000 | 0.08 |
| Helena (‘HE’) [15] | 27 | - | 41724 | 10432 | 100 | 0.08 |
| CovType (‘CO’) [6] | 54 | - | 371847 | 92962 | 7 | 0.55 |
| Jannis (‘JA’) [15] | 54 | - | 53588 | 13398 | 4 | 0.55 |
| Epsilon (‘EP’) [14] | 2000 | - | 320000 | 80000 | 2 | 0.55 |
| Higgs (‘HI’)[3] | 28 | - | 62752 | 15688 | 2 | 0.55 |
| Adult (‘AD’) [21] | 6 | 8 | 26048 | 6513 | 2 | 0.77 |

Table 2: The dataset used in our work are identical to FT-Transformers [14]. We set a threshold on the performance for nonconvergent models that correspond to the difficulty of the dataset as well as the task type. i.e. AD as it was an easy dataset, most nonconvergent solutions obtained a high accuracy score. We detail the methodology for identifying nonconvergent solutions in appendix B. The datasets have variable size, number of features, and number of classes which make it a comprehensive benchmark.

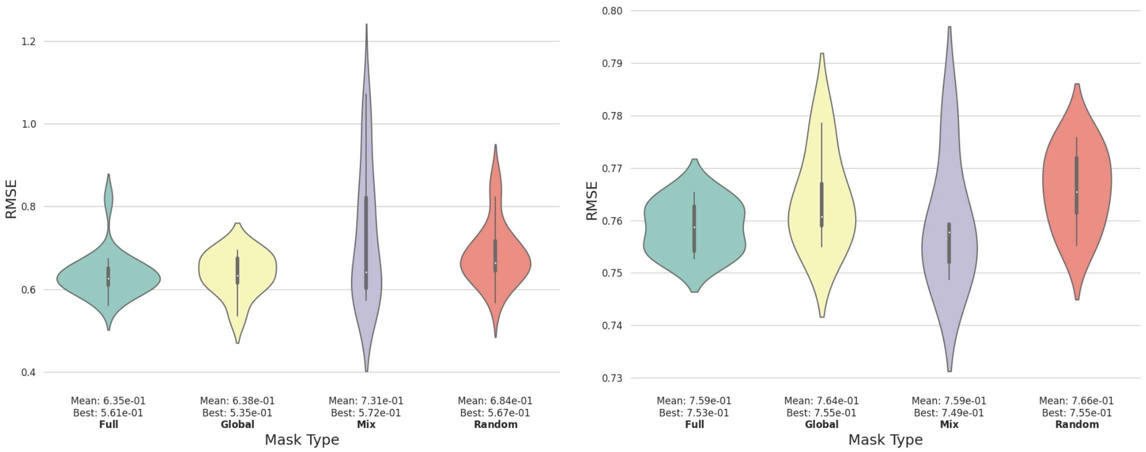
| Optim. ($\mathcal{O}$) | Method | Best | Mean | $P(\mathcal{O})$ |
|---|---|---|---|---|
| AdaB [37] | Random | **0.83** | **0.82** | 0.13 |
| | TPE | 0.78 | 0.73 | 0.07 |
| Adam [20] | Random | **0.89** | **0.79** | 0.22 |
| | TPE | 0.87 | 0.78 | 0.06 |
| AdamW [25] | Random | **0.81** | 0.74 | 0.20 |
| | TPE | 0.79 | **0.76** | 0.05 |
| RAdam [24] | Random | **0.83** | **0.75** | 0.16 |
| | TPE | 0.71 | 0.69 | 0.06 |
| SGD[30] | Random | **0.93** | **0.81** | 0.29 |
| | TPE | 0.92 | 0.76 | **0.76** |

Table 3: Evaluation of sampling bias of configuration selection strategy between TPE [4] and Random. TPE allocates significantly more budget to SGD (0.76 of the trials) and as a result of under-sampling other methods, the mean and best performance appears lower compared to random. Statistical tests such as ANOVA are not applicable with TPE.

| Source | Recommendation |
|---|---|
| Sampling Strategy† | The strategy used to select the parameters should be taken into consideration during analysis. The use of an HPO strategy can lead to misleading results. |
| Survival Bias† | Correlation of the type of errors with the evaluated method can identify non-random errors that can be introduced in the analysis. |
| Resources Utilization‡ | Use of a heuristic to allocate resources such that there are no unitilized resources and there are sufficient resources for a trial can prevent memory errors and experiment speed. Analysis of the memory utilization with the method hyper-parameters can help identify a heuristic. |
| Budget Allocation‡ | Statistical power of a study decreases with respect to the number of ablating components but increases with the number of trials. Selecting a budget should correspond to the number of ablating components. |
| Method Discrepancies$*$ | Use minimal differences between ablating components as to avoid user and numerical errors from implementation differences [36] |
| Versioning Errors$*$ | Create a self-contained module with all experiment artifacts. Version control of a module can help track the results corresponding to their implementation and configuration. |

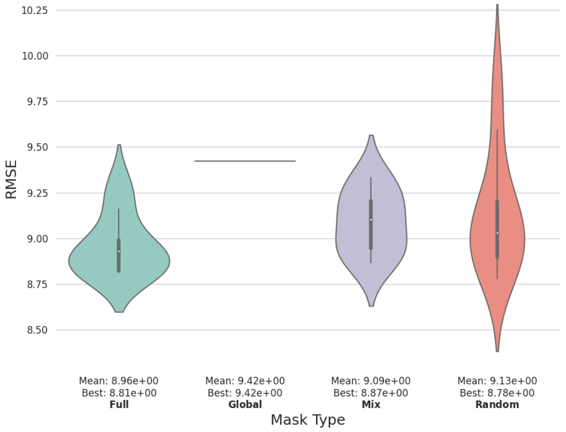
Table 4: We categorize the types of errors as Analysis †, Execution ‡ and Implemention$*$ errors.

## C.2 RQ3



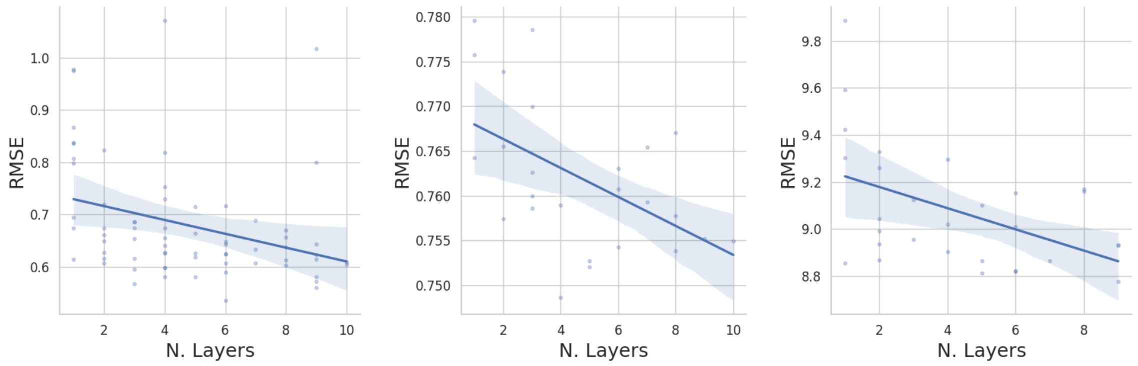
(a) Housing (‘CA’)

(b) Microsoft (‘MI’)



(c) Year (‘YE’)

Figure 5: We compare the effect of different mask-types on the root-mean-squared error (RMSE) of Tablator for dataset that the task is regression, where **lower** values are better. ‘Full’ mask performs consistently better. There is no clear benefit of other mask types where the results are mixed. Global Attention performs well for some dataset ‘CA’ but poor for other ‘YE’. The result can have us conclude that the inter-feature correlation for the specific dataset introduce noise for ‘CA’ but is necessary for ‘YE’.

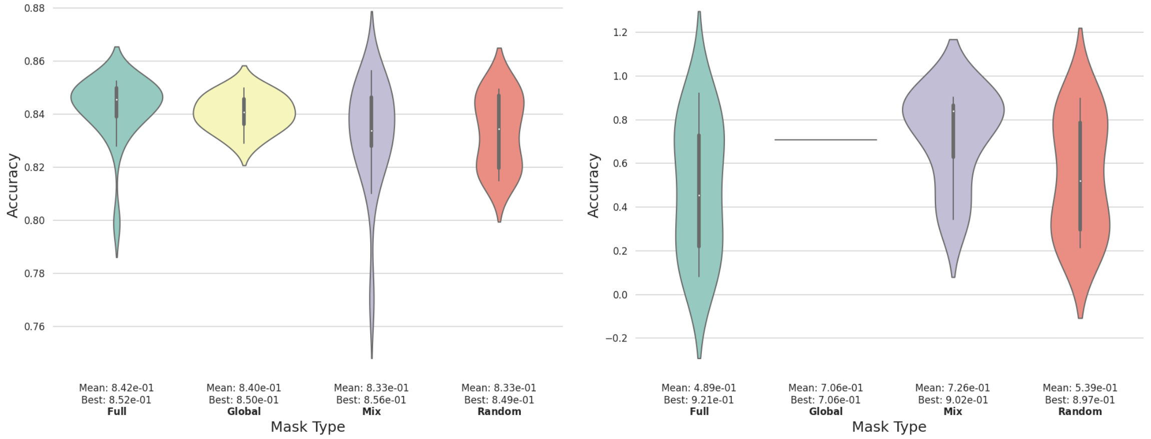


(a) Housing (‘CA’)

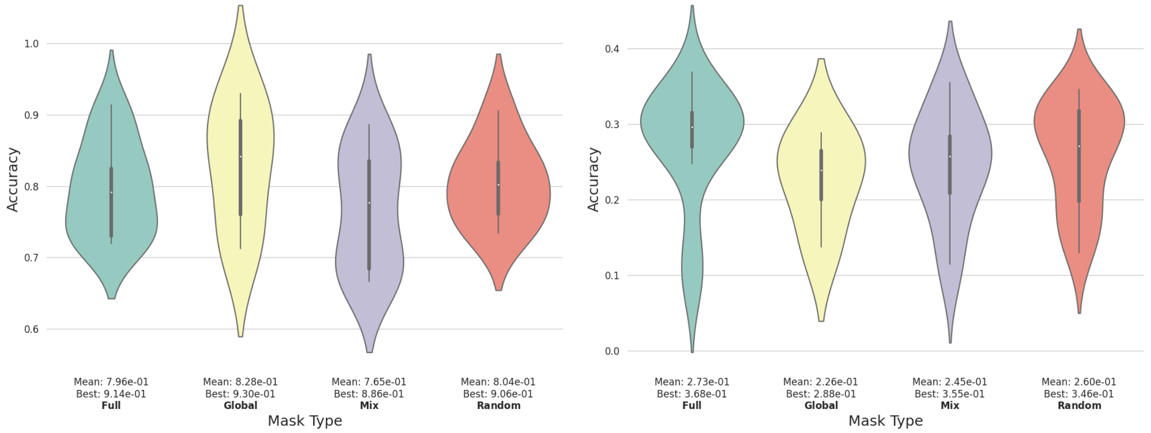
(b) Microsoft (‘MI’)

(c) Year (‘YE’)

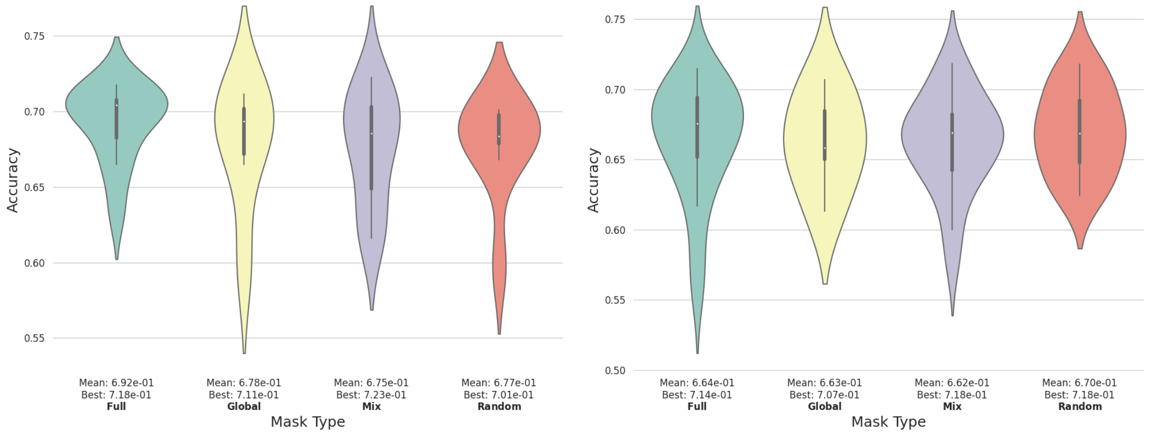
Figure 6: We evaluate the effect of a larger model on the root mean squared error (RMSE) of Tablator for the dataset that the task is regression, where **lower** values are better. The larger dataset (‘MI’) benefit from the added capacity where the slope is steeper when compared to the smaller dataset (‘CA’).



(a) Adult (‘AD’) (b) Aloi (‘AL’)

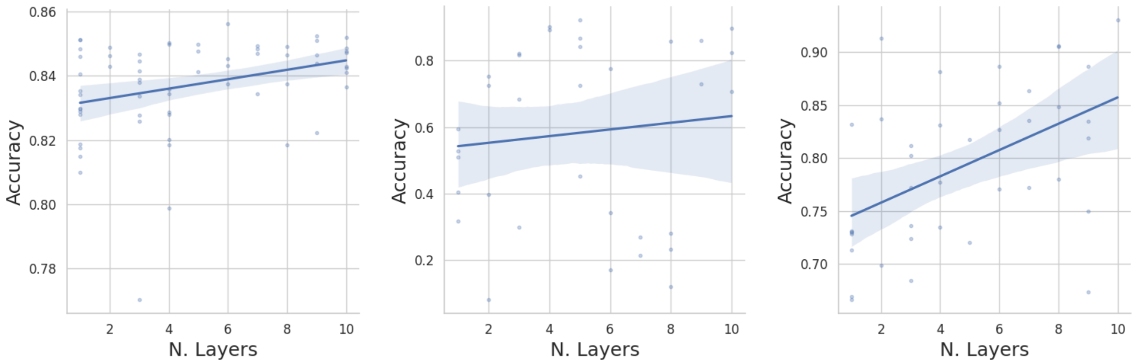


(c) CovType (‘CO’) (d) Helena (‘HE’)

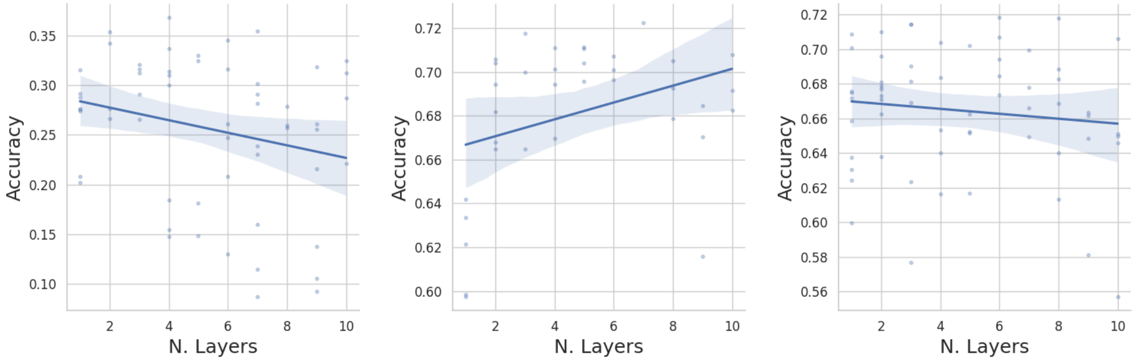


(e) Higgs (‘HI’) (f) Jannis (‘JA’)

Figure 7: We compare the effect of different mask-types on the accuracy score of Tablator for the dataset that the task is classification, where **higher** values are better. Our results are similar to fig. 5 where ‘Full’ mask performs consistently better and without a clear benefit of other mask types. The improved performance of ‘Global’ on ‘HI’, ’CO’ can lead us to conclude that there can be noise introduced from inter-feature attention but it is necessary for ‘HE’ and ‘AL’.



(a) Adult ('AD') (b) Aloi ('AL') (c) CovType ('CO')



(d) Helena ('HE') (e) Higgs ('HI') (f) Jannis ('JA')

Figure 8: We evaluate the effect of a larger model on the accuracy score of Tablator for the dataset that the task is classifcation, where **higher** values are better. Our results are similar to fig. 6 where the larger dataset i.e. ('CO'), benefit more from the added capacity where the slope is steeper compared to the smaller dataset i.e. ('JA').

## References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *CoRR*, abs/1907.10902, 2019.

[3] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):4308, 2014.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[5] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. 2011.

[6] Jock A Blackard and Denis J Dean. 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.

[7] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge*, pages 1–24. PMLR, 2011.

[8] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[9] William Falcon et al. Pytorch lightning. *GitHub repository*, 3, 2019.

[10] Dean Fergusson, Shawn D Aaron, Gordon Guyatt, and Paul Hébert. Post-randomisation exclusions: the intention to treat principle and excluding patients from analysis. *Bmj*, 325(7365):652–654, 2002.

[11] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28 (2015)*, pages 2962–2970, 2015.

[12] Jan-Mark Geusebroek, Gertjan J Burghouts, and Arnold WM Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61:103–112, 2005.

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[14] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *CoRR*, abs/2106.11959, 2021.

[15] Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, et al. Analysis of the automl challenge series. *Automated Machine Learning*, 177, 2019.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[17] 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.

[18] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[19] Albert Huang. Low density ablators, Aug 2017.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Ron Kohavi et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207, 1996.

[22] Erin LeDell and Sebastien Poirier. H2O AutoML: Scalable automatic machine learning. *7th ICML Workshop on Automated Machine Learning (AutoML)*, July 2020.

[23] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

[24] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[26] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging AI applications. *CoRR*, abs/1712.05889, 2017.

[27] R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] Tao Qin and Tie-Yan Liu. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*, 2013.

[30] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[31] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[33] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[34] Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019.

[35] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

[36] Donglin Zhuang, Xingyao Zhang, Shuaiwen Song, and Sara Hooker. Randomness in neural network training: Characterizing the impact of tooling. *Proceedings of Machine Learning and Systems*, 4:316–336, 2022.

[37] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, 33:18795–18806, 2020.

[38] Lucas Zimmer, Marius Lindauer, and Frank Hutter. Auto-pytorch tabular: Multi-fidelity metalearning for efficient and robust autodl. *arXiv preprint arXiv:2006.13799*, 2020.