Regularization is all you Need: Simple Neural Nets can Excel on Tabular Data

Anonymous Author(s)
Affiliation
Address
email

Abstract

Tabular datasets are the last "unconquered castle" for deep learning, with traditional ML methods like Gradient-Boosted Decision Trees still performing strongly even against recent specialized neural architectures. In this paper, we hypothesize that the key to boosting the performance of neural networks lies in rethinking the joint and simultaneous application of a large set of modern regularization techniques. As a result, we propose regularizing plain Multilayer Perceptron (MLP) networks by searching for the optimal combination/cocktail of 13 regularization techniques for each dataset using a joint optimization over the decision on which regularizers to apply and their subsidiary hyperparameters. We empirically assess the impact of these *regularization cocktails* for MLPs on a large-scale empirical study comprising 40 tabular datasets and demonstrate that (i) well-regularized plain MLPs significantly outperform recent state-of-the-art specialized neural network architectures, and (ii) they even outperform strong traditional ML methods, such as XGBoost.

1 Introduction

In contrast to the mainstream in deep learning (DL), in this paper, we focus on tabular data, a domain that we feel is understudied in DL. Nevertheless, it is of great relevance for many practical applications, such as climate science, medicine, manufacturing, finance, recommender systems, etc. During the last decade, traditional machine learning methods, such as Gradient-Boosted Decision Trees (GBDT) [5], dominated tabular data applications due to their superior performance, and the success story DL has had for raw data (e.g., images, speech, and text) stopped short of tabular data.

Even in recent years, the existing literature still gives mixed messages on the state-of-the-art status of deep learning for tabular data. While some recent neural network methods [1, 37] claim to outperform GBDT, others confirm that GBDT are still the most accurate method on tabular data [38, 22]. The extensive experiments on 40 datasets we report indeed confirm that recent neural networks [1, 37, 9] do not outperform GBDT when the hyperparameters of all methods are thoroughly tuned.

We hypothesize that the key to improving the performance of neural networks on tabular data lies in exploiting the recent DL advances on regularization techniques (reviewed in Section 3), such as data augmentation, residual blocks, model averaging (e.g. dropout, or snapshot ensembles), or on learning dynamics (e.g. look-ahead optimizer, or stochastic weight averaging). Indeed, we find that even plain Multilayer Perceptrons (MLPs) achieve state-of-the-art results when regularized by multiple modern regularization techniques applied jointly and simultaneously.

Applying multiple regularizers jointly is already a common standard for practitioners, who routinely mix regularization techniques (e.g. Dropout with early stopping and weight decay). However, the deeper question of "Which subset of regularizers gives the largest generalization performance on

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

a particular dataset among dozens of available methods?" remains unanswered, as practitioners currently combine regularizers via inefficient trial-and-error procedures. In this paper, we provide a simple, yet principled answer to that question, by posing the selection of the optimal subset of regularization techniques and their inherent hyperparameters, as a joint search for the best combination of MLP regularizers for each dataset among a pool of 13 modern regularization techniques and their subsidiary hyperparameters (Section 4).

From an empirical perspective, this paper is the first to provide compelling evidence that well-regularized neural networks (even simple MLPs!) indeed surpass the current state-of-the-art models in tabular datasets, including recent neural network architectures and GBDT (Section 6). In fact, the performance improvements are quite pronounced and highly significant. We believe this finding to potentially have far-reaching implications, and to open up a garden of delights of new applications on tabular datasets for DL.

Our contributions are as follows:

1. We demonstrate that modern DL regularizers (developed for DL applications on raw data, such as images, speech, or text) also substantially improve the performance of deep multi-layer perceptrons on tabular data.
2. We propose a simple, yet principled, paradigm for selecting the optimal subset of regularization techniques and their subsidiary hyperparameters (so-called *regularization cocktails*).
3. We demonstrate that these regularization cocktails enable even simple MLPs to outperform both recent neural network architectures, as well as traditional strong ML methods, such as GBDT, on tabular data. Specifically, we are the first to show neural networks to significantly (and substantially) outperform XGBoost in a fair, large-scale experimental study.

For reproducibility and to enable follow-up work, the code for all our experiments is publicly available at `https://github.com/ConferencePaperSubmission/RegularizationCocktail`

2 Related Work on Deep Learning for Tabular Data

Recently, various neural architectures have been proposed for improving the performance of neural networks on tabular data. TabNet [1] introduced a sequential attention mechanism for capturing salient features. Neural oblivious decision ensembles (NODE [37]) blend the concept of hierarchical decisions into neural networks. Regularization learning networks train a regularization strength on every neural weight by posing the problem as a large-scale hyperparameter tuning scheme [38]. The recent NET-DNF technique introduces a novel inductive bias in the neural structure corresponding to logical Boolean formulas in disjunctive normal forms [22]. An approach that is often mistaken as deep learning for tabular data is AutoGluon Tabular [9]. It builds ensembles of basic neural networks together with other traditional ML techniques, with its key contribution being a strong stacking approach. We emphasize that some of these publications claim to outperform Gradient Boosted Decision Trees (GDBT) [1, 37], and other papers explicitly stress that their neural networks do not outperform GBDT on tabular datasets [38, 22]. In contrast, we do not propose a new kind of neural architecture, but a novel paradigm for learning a combination of regularization methods.

3 An Overview of Regularization Methods for Deep Learning

Weight decay: The most classical approaches of regularization focused on minimizing the norms of the parameter values, e.g., either the L1 [41], the L2 [42], or a combination of L1 and L2 known as the Elastic Net [51]. A recent work fixes the malpractice of adding the decay penalty term before momentum-based adaptive learning rate steps (e.g., in common implementations of Adam [23]), by decoupling the regularization from the loss and applying it after the learning rate computation [29].

Data Augmentation: Among the augmentation regularizers, Cut-Out [8] proposes to mask a subset of input features (e.g., pixel patches for images) for ensuring that the predictions remain invariant to distortions in the input space. Along similar lines, Mix-Up [48] generates new instances as a linear span of pairs of training examples, while Cut-Mix [47] suggests super-positions of instance pairs with mutually-exclusive pixel masks. A recent technique, called Aug-Mix [16], generates instances by sampling chains of augmentation operations. On the other hand, the direction of reinforcement

learning (RL) for augmentation policies was elaborated by Auto-Augment [6], followed by a technique that speeds up the training of the RL policy [27]. Last but not least, adversarial attack strategies (e.g., FGSM [14]) generate synthetic examples with minimal perturbations, which are employed in training robust models [30].

Model Averaging: Ensembled machine learning models have been shown to reduce variance and act as regularizers [36]. A popular ensemble neural network with shared weights among its base models is Dropout [39], which was extended to a variational version with a Gaussian posterior of the model parameters [24]. As a follow-up, Mix-Out [25] extends Dropout by statistically fusing the parameters of two base models. Furthermore, so-called "snapshot ensembles" [17] can be created using models from intermediate convergence points of stochastic gradient descent with restarts [28].

Structural and Linearization: In terms of structural regularization, ResNet adds skip connections across layers [15], while the Inception model computes latent representations by aggregating diverse convolutional filter sizes [40]. A recent trend adds a dosage of *linearization* to deep models, where skip connections transfer embeddings from previous less non-linear layers [15, 18]. Along similar lines, the Shake-Shake regularization deploys skip connections in parallel convolutional blocks and aggregates the parallel representations through affine combinations [12], while Shake-Drop extends this mechanism to a larger number of CNN architectures [45].

Implicit: The last family of regularizers broadly encapsulates methods that do not directly propose novel regularization techniques but have an *implicit* regularization effect as a virtue of their 'modus operandi' [2]. For instance, Batch Normalization improves generalization by reducing the internal covariate shifts [20], while early stopping of the optimization procedure also yields a similar generalization effect [46]. On the other hand, stabilizing the convergence of the training routine is another implicit regularization, for instance by introducing learning rate scheduling schemes [28]. The recent strategy of stochastic weight averaging relies on averaging parameter values from the local optima encountered along the sequence of optimization steps [21], while another approach conducts updates in the direction of a few 'lookahead' steps [49].

4 Regularization Cocktails for Multilayer Perceptrons

4.1 Problem Definition

A training set is composed of features $\mathbf{X}^{(\text{Train})}$ and targets $\mathbf{y}^{(\text{Train})}$, while the test dataset is denoted by $\mathbf{X}^{(\text{Test})}, \mathbf{y}^{(\text{Test})}$. A parametrized function f, i.e., a neural network, approximates the targets as $\hat{\mathbf{y}} = f(\mathbf{X}; \boldsymbol{\theta})$, where the parameters $\boldsymbol{\theta}$ are trained to minimize a differentiable loss function $\mathcal{L}$ as $\arg\min_{\boldsymbol{\theta}} \quad \mathcal{L}\left(\mathbf{y}^{(\text{Train})}, f\left(\mathbf{X}^{(\text{Train})}; \boldsymbol{\theta}\right)\right)$. To generalize into minimizing $\mathcal{L}\left(\mathbf{y}^{(\text{Test})}, f(\mathbf{X}^{(\text{Test})}; \boldsymbol{\theta}\right)$, the parameters of f are controlled with a regularization technique Ω that avoids overfitting to the peculiarities of the training data. With a slight abuse of notation we denote $f\left(\mathbf{X};\ \Omega\left(\boldsymbol{\theta}; \boldsymbol{\lambda}\right)\right)$ to be the predictions of the model f whose parameters $\boldsymbol{\theta}$ are optimized under the regime of the regularization method $\Omega(\cdot; \boldsymbol{\lambda})$, where $\boldsymbol{\lambda} \in \Lambda$ represents the hyperparameters of Ω. The training data is further divided into two subsets as training and validation splits, the later denoted by $\mathbf{X}^{(\text{Val})}, \mathbf{y}^{(\text{Val})}$, such that $\boldsymbol{\lambda}$ can be tuned on the validation loss via the following hyperparameter optimization objective:

$$\begin{aligned} \boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \Lambda} \quad & \mathcal{L}\left(\mathbf{y}^{(\text{Val})}, f\left(\mathbf{X}^{(\text{Val})}; \boldsymbol{\theta}^*_{\boldsymbol{\lambda}}\right)\right), \\ \text{s.t.} \quad \boldsymbol{\theta}^*_{\boldsymbol{\lambda}} \in \arg\min_{\boldsymbol{\theta}} \quad & \mathcal{L}\left(\mathbf{y}^{(\text{Train})}, f(\mathbf{X}^{(\text{Train})}; \Omega\left(\boldsymbol{\theta}; \boldsymbol{\lambda}\right)\right). \end{aligned} \tag{1}$$

After finding the optimal (or in practice at least a well-performing) configuration $\boldsymbol{\lambda}^*$, we re-fit $\boldsymbol{\theta}$ on the entire training dataset, i.e., $\mathbf{X}^{(\text{Train})} \cup \mathbf{X}^{(\text{Val})}$ and $\mathbf{y}^{(\text{Train})} \cup \mathbf{y}^{(\text{Val})}$.

While the search for optimal hyperparameters $\boldsymbol{\lambda}$ is an active field of research in the realm of AutoML [19], still the choice of the regularizer Ω mostly remains an ad-hoc practice, where practitioners select a few combinations among popular regularizers (Dropout, L2, Batch Normalization, etc.). In contrast to prior studies, we hypothesize that the optimal regularizer is a cocktail mixture of a large set of regularization methods, all being simultaneously applied with different strengths (i.e., dataset-specific hyperparameters). Given a set of K regularizers $\left\{\left(\Omega^{(k)}\left(\cdot; \boldsymbol{\lambda}^{(k)}\right)\right\}_{k=1}^{K} := \left\{\Omega^{(1)}\left(\cdot; \boldsymbol{\lambda}^{(1)}\right), \ldots, \Omega^{(K)}\left(\cdot; \boldsymbol{\lambda}^{(K)}\right)\right\}$, each with its own hyperparameters $\boldsymbol{\lambda}^{(k)} \in \Lambda^{(k)}, \forall k \in \{1, \ldots, K\}$, the problem of finding the optimal cocktail of regularizers is:

$$\boldsymbol{\lambda}^* \in \underset{\boldsymbol{\lambda}^{(1)},\ldots,\boldsymbol{\lambda}^{(K)}}{\arg\min} \quad \mathcal{L}\left(\mathbf{y}^{(\text{Val})}, f\left(\mathbf{X}^{(\text{Val})}; \boldsymbol{\theta}^*_{\boldsymbol{\lambda}}\right)\right) \tag{2}$$
$$\text{s.t.: } \boldsymbol{\theta}^*_{\boldsymbol{\lambda}} \in \underset{\boldsymbol{\theta}}{\arg\min} \quad \mathcal{L}\left(\mathbf{y}^{(\text{Train})}, f\left(\mathbf{X}^{(\text{Train})}; \left\{\Omega^{(k)}\left(\boldsymbol{\theta}, \boldsymbol{\lambda}^{(k)}\right)\right\}_{k=1}^{K}\right)\right)$$

The intuitive interpretation of Equation 2 is searching for the optimal hyperparameters $\boldsymbol{\lambda}$ (i.e., strengths) of the cocktail's regularizers using the validation set, given that the optimal prediction model parameters $\boldsymbol{\theta}$ are trained under the regime of all the regularizers being applied jointly. We stress that, for each regularizer, the hyperparameters $\boldsymbol{\lambda}^{(k)}$ include a conditional hyperparameter controlling whether the k-th regularizer is applied at all or skipped. The best cocktail might comprise only a subset of regularizers.

4.2 Cocktail Search Space

To build our regularization cocktails we combine the 13 regularization methods listed in Table 1, which are selected among the categories of regularizers covered in Section 3. The regularization cocktail's search space with the exact ranges for the selected regularizers' hyperparameters is given in the same table. In total, the optimal cocktail is searched in a space of 19 hyperparameters.

While we can in principle use any hyperparameter optimization method, we decided to use the multi-fidelity Bayesian optimization method BOHB [10] since it achieves strong performance across a wide range of computing budgets by combining Hyperband [26] and Bayesian Optimization [32] and still has the convergence guarantees of Hyperband. Furthermore, BOHB can deal with the categorical hyperparameters we use for enabling or disabling regularization techniques and the corresponding conditional structures. In Appendix A we provide a brief description of how BOHB works. Some of the regularization methods cannot be combined, and we, therefore, introduce the following constraints to the proposed search space: (i) Shake-Shake and Shake-Drop are not simultaneously active since the latter builds on the former; (ii) Only one data augmentation technique out of Mix-Up, Cut-Mix, Cut-Out, and FGSM adversarial learning can be active at once due to a technical limitation of the base library we use [50].

5 Experimental Protocol

5.1 Experimental Setup and Datasets

We use a large collection of 40 tabular datasets (listed in Table 7 of Appendix D). This includes 31 datasets from the recent open-source OpenML AutoML Benchmark [13][1]. In addition, we added 9 popular datasets from UCI [3] and Kaggle that contain roughly 100K+ instances. Our resulting benchmark of 40 datasets includes tabular datasets that represent diverse classification problems, containing between 452 and 416 188 data points, and between 4 and 2 001 features, varying in terms of the number of numerical and categorical features. The datasets are retrieved from the OpenML repository [44] and split as 60% training, 20% validation, and 20% testing sets. The data is standardized to have zero mean and unit variance where the statistics for the standardization are calculated on the training split.

We ran all experiments on a CPU cluster, each node of which contains two Intel Xeon E5-2630v4 at 2.2GHz with 20 CPU cores and a total memory of 128GB. We chose the PyTorch library [34] as a deep learning framework and extended the AutoDL-framework Auto-Pytorch [31, 50] with our implementations for the regularizers of Table 1.

To optimally utilize resources, we ran BOHB with 10 workers in parallel, where each worker had access to 2 CPU cores and 12GB of memory, executing one configuration at a time. Taking into account the dimensions D of the considered configuration spaces, we ran BOHB for at most 4 days, or at most $40 \times D$ hyperparameter configurations, whichever came first. During the training phase, each configuration was run for 105 epochs, in accordance with the cosine learning rate annealing with restarts (described in the following subsection). For the sake of studying the effect on more datasets,

[1] The remaining 8 datasets from that benchmark were too large to run effectively on our cluster.

Group	Regularizer	Hyperparameter	Type	Range	Conditionality
Implicit	BN	BN-active	Boolean	$\{\text{True}, \text{False}\}$	–
	SWA	SWA-active	Boolean	$\{\text{True}, \text{False}\}$	-
	LA	LA-active	Boolean	$\{\text{True}, \text{False}\}$	–
		Step size	Continuous	$[0.5, 0.8]$	LA-active
		Num. steps	Integer	$[1, 5]$	LA-active
W. Decay	WD	WD-active	Boolean	$\{\text{True}, \text{False}\}$	–
		Decay factor	Continuous	$[10^{-5}, 0.1]$	WD-active
M. Averaging	DO	DO-active	Boolean	$\{\text{True}, \text{False}\}$	–
		Dropout shape	Nominal	$\{\text{funnel}, \text{long funnel}, \text{diamond}, \text{hexagon}, \text{brick}, \text{triangle}, \text{stairs}\}$	DO-active
		Drop rate	Continuous	$[0.0, 0.8]$	DO-active
	SE	SE-active	Boolean	$\{\text{True}, \text{False}\}$	-
Structural	SC	SC-active	Boolean	$\{\text{True}, \text{False}\}$	–
		MB choice	Nominal	$\{\text{SS}, \text{SD}, \text{Standard}\}$	SC-active
	SD	Max. probability	Continuous	$[0.0, 1.0]$	SC-active $\wedge$ MB choice = SD
	SS	-	-	-	SC-active $\wedge$ MB choice = SS
Augmentation	–	Augment	Nominal	$\{\text{MU}, \text{CM}, \text{CO}, \text{AT}, \text{None}\}$	–
	MU	Mix. magnitude	Continuous	$[0.0, 1.0]$	Augment $=$ MU
	CM	Probability	Continuous	$[0.0, 1.0]$	Augment $=$ CM
	CO	Probability	Continuous	$[0.0, 1.0]$	Augment $=$ CO
		Patch ratio	Continuous	$[0.0, 1.0]$	Augment $=$ CO
	AT	-	-	-	Augment $=$ AT

Table 1: The configuration space for the regularization cocktail regarding the **explicit regularization hyperparameters** of the methods and the conditional constraints enabling or disabling them. (BN: Batch Normalization, SWA: Stochastic Weight Averaging, LA: Lookahead Optimizer, WD: Weight Decay, DO: Dropout, SE: Snapshot Ensembles, SC: Skip Connection, MB: Multi-branch choice, SD: Shake-Drop, SS: Shake-Shake, MU: Mix-Up, CM: Cut-Mix, CO: Cut-Out, and AT: FGSM Adversarial Learning)

we only evaluated a single train-val-test split. After the training phase is completed, we report the results of the best hyperparameter configuration found, retrained on the joint train and validation set.

5.2 Fixed Architecture and Optimization Hyperparameters

In order to focus exclusively on investigating the effect of regularization we fix the neural architecture to a simple multilayer perceptron (MLP) and also fix some hyperparameters of the general training procedure. These fixed hyperparameter values, as specified in Table 3 of Appendix B.1, have been tuned for maximizing the performance of an unregularized neural network on our dataset collection (see Table 7 in Appendix D). We use a 9-layer feed-forward neural network with 512 units for each layer, a choice motivated by previous work [33].

Moreover, we set a low learning rate of 10^{-3} after performing a grid search for finding the best value across datasets. We use AdamW [29], which implements decoupled weight decay, and cosine annealing with restarts [28] as a learning rate scheduler. Using a learning rate scheduler with restarts helps in our case because we keep a fixed initial learning rate. For the restarts, we use an initial budget of 15 epochs, with a budget multiplier of 2, following published practices [50]. Additionally, since our benchmark includes imbalanced datasets, we use a weighted version of categorical cross-entropy and balanced accuracy [4] as the evaluation metric.

5.3 Research Hypotheses and Associated Experiments

Hypothesis 1: Regularization cocktails outperform state-of-the-art deep learning architectures on tabular datasets.

Experiment 1: We compare our well-regularized MLPs against the recently proposed deep learning architectures Node [37] and TabNet [1]. We also compare against AutoGluon Tabular [9]

and add an unregularized version of our MLP for reference, as well as a version of our MLP regularized with Dropout (where the dropout hyperparameters are tuned on every dataset).

Hypothesis 2: Regularization cocktails outperform Gradient-Boosted Decision Trees, as the most commonly used traditional ML method for tabular data.

Experiment 2: We compare against state-of-the-art classifiers for tabular data. In particular, we compare against Gradient Boosted Decision Trees (GBDT), the de-facto state-of-the-art in tabular datasets. We use two different implementations of GBDT: an implementation from scikit-learn [35] and optimized by Auto-sklearn [11], and the popular XGBoost [5].

5.4 Experimental Setup for the Baselines

All baselines use the same train, validation, and test splits, the same seed, and the same HPO resources and constraints as for our automatically constructed regularization cocktails (4 days on 20 CPU cores with 128GB of memory). After finding the best incumbent configuration, the baselines are refitted on the union of the training and validation sets and evaluated on the test set. The baselines consist of two recent neural architectures, AutoGluon Tabular with neural networks, and two implementations of GBDT, as follows:

TabNet: This library does not provide an HPO algorithm by default; therefore, we also used BOHB for this search space, with the hyperparameter value ranges recommended by the authors [1].

Node: This library does not offer an HPO algorithm by default. We performed a grid search among the hyperparameter value ranges as proposed by the authors [37]; however, we faced multiple memory and runtime issues in running the code. To overcome these issues we used the default hyperparameters the authors used in their public implementation[2].

AutoGluon Tabular: This library constructs stacked ensembles with bagging among diverse neural network architectures having various kinds of regularization [9]. The training of the stacking ensemble of neural networks and its hyperparameter tuning are integrated into the library. While AutoGluon Tabular by default uses a broad range of traditional ML techniques, here, in order to study it as a "pure" deep learning method, we restrict it to only use neural networks as base learners.

ASK-GBDT: The GBDT implementation of scikit-learn offered by Auto-sklearn [11] uses SMAC for HPO, and we used the default hyperparameter search space given by the library.

XGBoost: The original library [5] does not incorporate an HPO algorithm by default, so we used BOHB for its HPO. We defined a search space for XGBoost's hyperparameters following the best practices by the community; we describe this in the Appendix B.2.

For in-depth details about the different baseline configurations with the exact hyperparameter search spaces, please refer to Appendix B.2.

6 Experimental Results

Table 2 presents the comparative results of our MLPs regularized with the proposed regularization cocktails against seven baselines: two state-of-the-art architectures, AutoGluon Tabular with neural networks, two Gradient-Boosted Decision Tree (GBDT) implementations, as well as two reference MLPs (unregularized and regularized only with Dropout). It is worth re-emphasizing that the hyperparameters of all the presented baselines (except the unregularized MLP, which has no hyperparameters) are carefully tuned on a validation set as detailed in Section 5 and the appendices referenced therein. The table entries represent the test sets' balanced accuracies achieved over the described large-scale collection of 40 datasets. Figure 1 visualizes these results, showing substantial improvements for our method.

To assess the statistical significance, we analyze the ranks of the classification accuracies across the 40 datasets. We use the Critical Difference (CD) diagram of the ranks based on the Wilcoxon significance test, a standard metric for comparing classifiers across multiple datasets [7]. The overall empirical comparison of the elaborated methods is given in Figure 2. The analysis of neural network

[2] `https://github.com/Qwicen/node/blob/master/`

Dataset	#Ins./#Feat.	MLP	MLP+D	XGB.	ASK-G.	TabN.	Node	AutoGL.	MLP+C
anneal	898 / 39	84.131	86.916	85.416	**90.000**	84.248	20.000	80.000	89.270
kr-vs-kp	3196 / 37	99.701	**99.850**	**99.850**	**99.850**	93.250	97.264	99.687	**99.850**
arrhythmia	452 / 280	37.991	38.704	48.779	46.850	43.562	N/A	48.934	**61.461**
mfeat.	2000 / 217	97.750	**98.000**	**98.000**	97.500	97.250	97.250	**98.000**	**98.000**
credit-g	1000 / 21	69.405	68.095	68.929	71.191	61.190	73.095	69.643	**74.643**
vehicle	846 / 19	83.766	82.603	74.973	80.165	79.654	75.541	**83.793**	82.576
kc1	2109 / 22	70.274	72.980	66.846	63.353	52.517	55.803	67.270	**74.381**
adult	48842 / 15	76.893	78.520	79.824	79.830	77.155	78.168	80.557	**82.443**
walking.	149332 / 5	60.997	63.754	61.616	62.764	56.801	N/A	60.800	**63.923**
phoneme	5404 / 6	87.514	**88.387**	87.972	88.341	86.824	82.720	83.943	86.619
skin-seg.	245057 / 4	99.971	99.962	99.968	99.967	99.961	N/A	**99.973**	99.953
ldpa	164860 / 8	62.831	67.035	**99.008**	68.947	54.815	N/A	53.023	68.107
nomao	34465 / 119	95.917	96.232	96.872	**97.217**	95.425	96.217	96.420	96.826
cnae	1080 / 857	87.500	90.741	94.907	93.519	89.352	**96.759**	92.593	95.833
blood.	748 / 5	67.836	**68.421**	62.281	64.985	64.327	50.000	67.251	67.617
bank.	45211 / 17	78.076	83.145	72.658	72.283	70.639	74.607	79.483	**85.993**
connect.	67557 / 43	73.627	76.345	72.374	72.645	72.045	N/A	75.622	**80.073**
shuttle	58000 / 10	99.475	99.892	98.563	98.571	88.017	42.805	83.433	**99.948**
higgs	98050 / 29	67.752	66.873	72.944	72.926	72.036	N/A	**73.798**	73.546
australian	690 / 15	86.268	86.268	**89.717**	88.589	85.278	83.468	88.248	87.088
car	1728 / 7	97.442	99.690	92.376	**100.000**	98.701	46.119	99.675	99.587
segment	2310 / 20	**94.805**	94.589	93.723	93.074	91.775	90.043	91.991	93.723
fashion.	70000 / 785	90.464	90.507	91.243	90.457	89.793	N/A	91.336	**91.950**
jungle.	44819 / 7	97.061	97.237	87.325	83.070	73.425	N/A	93.017	**97.471**
numerai	96320 / 22	50.262	50.301	52.363	52.421	51.599	52.364	51.706	**52.668**
devnagari	92000 / 1025	96.125	97.000	93.310	77.897	94.179	N/A	97.734	**98.370**
helena	65196 / 28	16.836	23.983	21.994	21.144	19.032	N/A	27.115	**27.701**
jannis	83733 / 55	51.505	55.118	55.225	55.593	56.214	N/A	58.526	**65.287**
volkert	58310 / 181	65.081	66.996	64.170	63.428	59.409	N/A	70.195	**71.667**
miniboone	130064 / 51	90.639	94.099	94.024	94.137	62.173	N/A	**94.978**	94.015
apsfailure	76000 / 171	87.759	91.194	88.825	91.797	51.444	N/A	88.890	**92.535**
christine	5418 / 1637	70.941	70.756	**74.815**	74.447	69.649	73.247	74.170	74.262
dilbert	10000 / 2001	96.930	96.733	**99.106**	98.704	97.608	N/A	98.758	99.049
fabert	8237 / 801	63.707	64.814	70.098	**70.120**	62.277	66.097	68.142	69.183
jasmine	2984 / 145	78.048	76.211	**80.546**	78.878	76.690	80.053	80.046	79.217
sylvine	5124 / 21	93.070	93.363	**95.509**	95.119	83.595	93.852	93.753	94.045
dionis	416188 / 61	91.905	92.724	91.222	74.620	83.960	N/A	**94.127**	94.010
aloi	108000 / 129	92.331	93.852	95.338	13.534	93.589	N/A	**97.423**	97.175
ccfraud	284807 / 31	50.000	50.000	90.303	92.514	85.705	N/A	91.831	**92.531**
clickpred.	399482 / 12	63.125	**64.367**	58.361	58.201	50.163	N/A	54.410	64.280
Wins/Loses/Ties	**MLP+C** vs ...	35/5/0	30/8/2	26/12/2	29/11/0	38/2/0	19/2/0	30/9/1	-
Wilcoxon p-value	**MLP+C** vs ...	5.3×10^{-7}	8.9×10^{-6}	6×10^{-4}	2.8×10^{-4}	4.5×10^{-8}	8.2×10^{-8}	4×10^{-5}	-

Table 2: Comparison of well-regularized MLPs vs. rivals in terms of balanced accuracy. N/A values indicate a failure due to exceeding the cluster's memory (24GB per process) or runtime limits (4 days). The acronyms stand for MLP+D: MLP with Dropout, XGB.: XGBoost, ASK-G.: GBDT by Auto-sklearn, AutoGL.: Autogluon, TabN.: TabNet and MLP+C: our MLP regularized by cocktails.

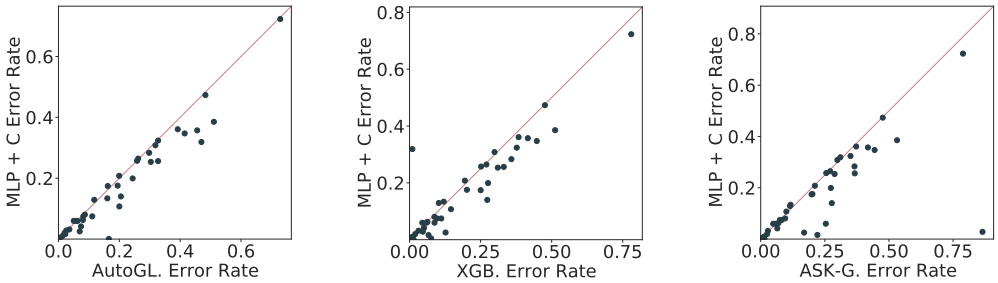
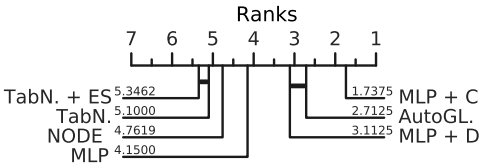


Figure 1: Comparison of our proposed dataset-specific cocktail (MLP+C) against the top three baselines. Each dot in the plot represents a dataset, the y-axis our method's errors and the x-axis the baselines' errors.

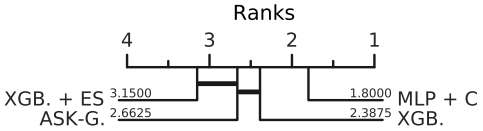
baselines in Subplot 2a reveals a clear statistical significance of the regularization cocktails against the other methods. Apart from AutoGluon, the other neural architectures are not competitive even against a MLP regularized only with Dropout and optimized with our standard, fixed training pipeline of Adam with cosine annealing. To be even fairer to the weaker baselines (TabNet and Node) we tried boosting them by adding early stopping (indicated with "+ES"), but their rank did not improve. Overall, the large-scale experimental analysis deduces that Hypothesis 1 in Section 5.3 is validated and that **well-regularized simple deep MLPs outperform specialized neural architectures**.

Next, we analyze the empirical significance of our well-regularized MLPs against the GBDT implementations in Figure 2b. The results show that our MLPs outperform both GBDT variants (XGBoost and auto-sklearn) with a statistically significant margin. We added early stopping ("+ES") to XGBoost, but it did not improve its performance. Among the GBDT implementations, XGBoost has a non-significant margin over auto-sklearn. We conclude that **well-regularized simple deep MLPs outperform GBDT**, which validates Hypothesis 2 in Section 5.3.

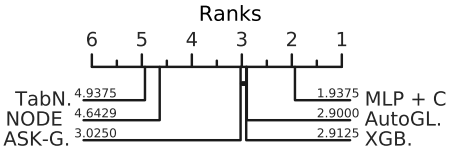
The final cumulative comparison in Figure 2c provides a further result: none of the specialized previous deep learning methods (TabNet, NODE, AutoGluon Tabular) outperforms GBDT significantly. To the best of our awareness, this paper is therefore the first to demonstrate that neural networks beat GBDT with a statistically significant margin over a large-scale experimental protocol that conducts a thorough hyperparameter optimization for all methods.



(a) CD of MLP+C vs. neural networks



(b) CD of MLP+C vs. GBDT



(c) CD of MLP+C vs. all baselines

Figure 2: **Critical difference diagrams** with a Wilcoxon significance analysis on 40 datasets. Connected ranks via a bold bar indicate that performances are not significantly different ($p > 0.05$).

Lastly, Figure 3 provides a further analysis on the most prominent regularizers of the MLP cocktails, based on the frequency of regularization methods that our HPO procedure selected for each dataset's cocktail. In the left plot, we show the frequent individual regularizers, while in the right plot the frequencies are grouped by types of regularizers. The grouping reveals that a cocktail for each dataset often has at least one ingredient from every regularization family (detailed in Section 3), highlighting the need for jointly applying diverse regularization methods.

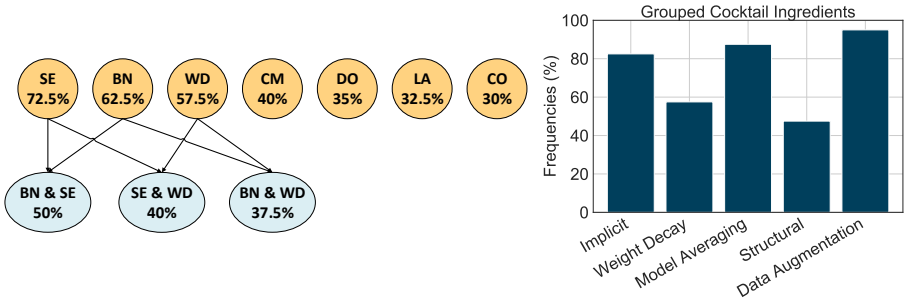


Figure 3: **Left**: Cocktail ingredients occurring in at least 30% of the datasets. **Right**: Clustered histogram (union of member occurrences) with the acronyms from Table 1. Implicit: {BN, LA, SWA}, M. Averaging: {DO, SE}, Structural: {SC, SS, SD}, D. Augmentation: {MU, CM, CO, AT}.

7 Conclusion

Summary. Focusing on the important domain of tabular datasets, this paper studied improvements to deep learning (DL) by better regularization techniques. We presented *regularization cocktails*, per-dataset-optimized combinations of many regularization techniques, and demonstrated that these improve the performance of even simple neural networks enough to substantially and significantly surpass XGBoost, the current state of the art method for tabular datasets. We conducted a large-scale experiment involving 13 regularization methods and 40 datasets and empirically showed that (i) modern DL regularization methods developed in the context of raw data (e.g., vision, speech, text) substantially improve the performance of deep neural networks on tabular data; (ii) regularization cocktails significantly outperform recent neural networks architectures, and most importantly iii) regularization cocktails outperform GBDT on tabular datasets.

Limitations. Compared to traditional machine learning methods, such as XGBoost, fitting deep neural networks is slow, and our regularization cocktails require per-dataset hyperparameter optimization on top. Therefore, in many data science applications, practitioners may currently still prefer the cheaper, albeit less accurate, traditional methods. To comprehensively study basic principles, we have also chosen an empirical evaluation that has many limitations. We only studied classification, not regression. We only used somewhat balanced datasets (the ratio of the minority class and the majority class is above 0.05). We did not study the regime of extremely few data points (our smallest data set contained 452 data points, our largest 416 188 data points). We also did not study datasets with extreme outliers, missing labels, semi-supervised data, streaming data, and many more modalities in which tabular data arises. An important point worth noticing is that the recent neural network architectures (Section 5.4) could also benefit from our regularization cocktails, however, integrating the regularizers into the baseline libraries requires considerable coding efforts.

Future Work. This work opens up the door for a wealth of exciting follow-up research. Firstly, the per-dataset optimization of regularization cocktails may be substantially sped up by using meta-learning across datasets [43]. Secondly, as we have used a fixed neural architecture, our method's performance may be further improved by using joint architecture and hyperparameter optimization. Thirdly, regularization cocktails should also be tested under all the data modalities under "Limitations" above. In addition, it is interesting to validate the gain of integrating our well-regularized MLPs into modern AutoML libraries, by combining them with enhanced feature preprocessing and ensembling.

Take-away. *Even simple neural networks can achieve competitive classification accuracies on tabular datasets when they are well regularized, using dataset-specific regularization cocktails found via standard hyperparameter optimization.*

8 Societal Implications

Enabling neural networks to advance the state-of-the-art on tabular datasets may open up a garden of delights in many crucial applications, such as climate science, medicine, manufacturing, and recommender systems. In addition, the proposed networks can serve as a backbone for applications of data science for social good, such as the realm of fair machine learning where the associated data are naturally in a tabular form. However, there are also potential negative consequences of advancing deep learning for tabular data. Our neural networks are slower to train and tune than most traditional ML methods, which could increase carbon emissions and thus potentially cause environmental harm. However, we see this risk to be low, since our networks are orders of magnitude cheaper to train (even on CPUs) than standard deep neural networks for raw data. Nevertheless, in order to limit this risk further, we encourage research into energy-efficient deep learning on tabular data.

References

[1] S. Arik and T. Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI Conference on Artificial Intelligence*, 2021.

[2] S. Arora, N. Cohen, W. Hu, and Y. Luo. Implicit regularization in deep matrix factorization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7413–7424. Curran Associates, Inc., 2019.

[3] A. Asuncion and D. Newman. Uci machine learning repository, 2007.

[4] K. Henning Brodersen, C., and E. Klaas Stephan J. M Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124. IEEE, 2010.

[5] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[6] E. Cubuk, B. Zoph, D. Mane, and V.Vasudevanand Q. Le. Autoaugment: Learning augmentation strategies from data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[7] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.

[8] T. Devries and G. Taylor. Improved regularization of convolutional neural networks with cutout. *ArXiv*, abs/1708.04552, 2017.

[9] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola. Autogluon-tabular: Robust and accurate automl for structured data. *CoRR*, abs/2003.06505, 2020.

[10] S. Falkner, A.Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scalae. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pages 1436–1445, July 2018.

[11] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, page 2755–2763. MIT Press, 2015.

[12] X. Gastaldi. Shake-shake regularization of 3-branch residual networks. In *5th International Conference on Learning Representations, ICLR*. OpenReview.net, 2017.

[13] P. Gijsbers, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren. An open source automl benchmark. *arXiv preprint arXiv:1907.00909 [cs.LG]*, 2019. Accepted at AutoML Workshop at ICML 2019.

[14] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[16] D. Hendrycks, N. Mu, E. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. Augmix: A simple method to improve robustness and uncertainty under data shift. In *International Conference on Learning Representations*, 2020.

[17] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. Hopcroft, and K. Weinberger. Snapshot Ensembles: Train 1, Get M for Free. *International Conference on Learning Representations*, November 2017.

[18] G. Huang, Z. Liu, L. van der Maaten, and K. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[19] F. Hutter, L. Kotthoff, and J. Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019. In press, available at http://automl.org/book.

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456. PMLR, 07–09 Jul 2015.

[21] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. Wilson. Averaging weights leads to wider optima and better generalization. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI*, pages 876–885. AUAI Press, 2018.

[22] L. Katzir, G. Elidan, and R. El-Yaniv. Net-{dnf}: Effective deep modeling of tabular data. In *International Conference on Learning Representations*, 2021.

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[24] D. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2575–2583. MIT Press, 2015.

[25] C. Lee, K. Cho, and W. Kang. Mixout: Effective regularization to finetune large-scale pretrained language models. In *International Conference on Learning Representations*, 2020.

[26] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1):6765–6816, January 2017.

[27] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim. Fast autoaugment. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 6665–6675. Curran Associates, Inc., 2019.

[28] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR) 2017 Conference Track*, April 2017.

[29] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[30] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[31] H. Mendoza, A. Klein, M. Feurer, J. Tobias Springenberg, M. Urban, M. Burkart, M. Dippel, M. Lindauer, and F. Hutter. Towards automatically-tuned deep neural networks. In F. Hutter, L. Kotthoff, and J. Vanschoren, editors, *AutoML: Methods, Sytems, Challenges*, chapter 7, pages 141–156. Springer, December 2019.

[32] J. Mockus. Application of bayesian approach to numerical methods of global and stochastic optimization. *J. Glob. Optim.*, 4(4):347–365, 1994.

[33] A. Emin Orhan and X. Pitkow. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175*, 2017.

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[35] 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.

[36] R. Polikar. Ensemble Learning. In C. Zhang and Y. Ma, editors, *Ensemble Machine Learning: Methods and Applications*, pages 1–34. Springer US, 2012.

[37] S. Popov, S. Morozov, and A. Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *International Conference on Learning Representations*, 2020.

[38] I. Shavitt and E. Segal. Regularization learning networks: Deep learning for tabular datasets. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, page 1386–1396. Curran Associates Inc., 2018.

[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[40] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[41] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[42] A. Tikhonov. On the stability of inverse problems. In *Doklady Akademii Nauk SSSR*, 1943.

[43] J. Vanschoren. Meta-learning. In F. Hutter, L. Kotthoff, and J. Vanschoren, editors, *Automated Machine Learning - Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning, pages 35–61. Springer, 2019.

[44] J. Vanschoren, J. Van Rijn, B. Bischl, and L. Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

[45] Y. Yamada, M. Iwamura, and K. Kise. Shakedrop regularization, 2018.

[46] Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, August 2007.

[47] S. Yun, D. Han, S. Joon, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision (ICCV)*, 2019.

[48] H. Zhang, M. Cisse, Y. Dauphin, and D. Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[49] M. Zhang, J. Lucas, J. Ba, and G. Hinton. Lookahead optimizer: k steps forward, 1 step back. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*, pages 9593–9604, 2019.

[50] L. Zimmer, M. Lindauer, and F. Hutter. Auto-pytorch tabular: Multi-fidelity metalearning for efficient and robust autodl. *IEEE TPAMI*, 2021. IEEE Early Access.

[51] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 8.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 1 and Section 5.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and Section 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Yes, we report the statistical significance of the performance difference between our method and the baselines in Section 6
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Section1.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]