ON HYPERPARAMETER TUNING ON THE TEST SET

Anonymous authors
Paper under double-blind review

ABSTRACT

"*Don't tune hyperparameters on the test set*" is often stated in machine learning textbooks. Violating it is considered a cardinal sin that produces misleadingly optimistic results, corrupts benchmark integrity, and thus can even be interpreted as scientific fraud. Yet evidence suggests that test set hyperparameter tuning does occur in practice, making it all the more important to understand its actual consequences. So how bad is it, really? In this work we question this dogma and put it to an empirical test. We systematically study the magnitude of the performance inflation caused by tuning the hyperparameters on the test set for MNIST-1D, CIFAR-10, and three tasks from the GLUE benchmark. Our experiments show that while the effect is real and significant, it is frequently small relative to other sources of noise. In many cases, we find that tuning on the test set recovers exactly the same model as when tuning on the validation set. Most importantly, we find that the rankings of models remain essentially preserved after tuning on the test set and therefore that consistent test-set tuning may not invalidate benchmarks or model selection. Our results call for a more nuanced view of tuning hyperparameters on the test set, stimulating researchers to openly report test tuning.

1 INTRODUCTION

Overfitting is an inflated generalization estimate, possibly distorting model selection rankings, which may lead to deploying the wrong model. We investigate overfitting of hyperparameters: configurations that determine model training behavior. Specifically, we investigate overfitting by tuning hyperparameters on the test set, *i.e.*, the same data set used for reporting generalization performance.

Machine learning textbooks prohibit hyperparameter tuning on the test set. From a theoretical perspective, a hyperparameter is no different than any other model parameter that is set at training time. Thus, an extreme view can see tuning hyperparameters on the test set as academic fraud in the current benchmarking paradigm (Hardt, 2026). For example, tuning model parameter initialization on the test set (Picard, 2021) will give excellent accuracy; independent of other claims for empirical gain in machine learning scholarship (Lipton & Steinhardt, 2019; Sculley et al., 2018).

In practice, however, several implementations already tune hyperparameters on the test set, both in computer vision (PyTorch Contributors, 2024; Wightman, 2019; Touvron et al., 2021a; Bao et al., 2022b; He et al., 2022a) and in natural language processing (Devlin et al., 2019; Google Research, 2018; Liu et al., 2019; Facebook AI Research, 2019). Accordingly, it can be argued that a hyperparameter is *not* like an individual model parameter, because hyperparameters are usually regularized by their global effect on a model, where *e.g.,* the learning rate is typically shared over all model parameters. Moreover, in practice, not an infinite number of hyperparameter configurations are evaluated, but only a handful (Hastie et al., 2009). Thus, there is tension between theory and practice, and here we investigate how bad hyperparameter overfitting really is.

In this paper, we give the first systematic empirical evaluation of overfitting by tuning hyperparameters on the test set. We carefully construct a controlled evaluation benchmark of repeated hyperparameter tuning on a validation set, compared to tuning on the test set. We evaluate on the controlled MNIST-1D (Greydanus & Kobak, 2024), image classifcation on CIFAR-10 (Krizhevsky, 2009) and three natural language understanding tasks from the GLUE benchmark (Wang et al., 2019) (RTE, MRPC and CoLA). We test multiple model families, and train $36{,}000$ model variants. We investigate how easily models overfit, how severe the overfitting is, and if the overfitting leads to differences in model selection.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Our results show that overfitting by tuning on the test set is significant on every benchmark we evaluate. Yet, the same hyperparameter configuration is often best on both the validation set as on the test set and the accuracy gain by overfitting is mild when compared to other sources of randomness such as data splits. Model selection rankings barely change when all models are tuned on test data, and we did not observe that the best model is not selected.

Our results call for a nuanced view on tuning hyperparameters on the test set. Overfitting is real, and comparing validation-tuned models vs test-tuned models benefits test-tuned models. However, model rankings rarely change when all models under comparison are tuned on the same data, making the effect of test-tuning relatively benign. Our paper gives empirical evidence to normalize hyperparameter tuning on the test set, and we encourage authors not to hide or be reluctant to admit test-tuning, but openly report it.

2 Related Work

Adaptive overfitting and test set reuse. Adaptive data analysis is formalized (Dwork et al., 2015; Makarova et al., 2021) as a setting in which a holdout dataset is queried repeatedly by adaptively chosen hypotheses. Theory suggests that repeated queries may lead to overfitting of the hidden test set, even though this may be not intentional (e.g. through publication bias or human actions).

This suggests that long-term reuse of benchmark test sets should lead to measurable overfitting. However, empirical evidence paints a more nuanced picture. New test sets for CIFAR-10 and ImageNet, constructed to match the original data collection process, show that accuracy drops of 3–15% on CIFAR-10 and 11–14% on ImageNet are largely explained by a distribution shift instead of adaptive overfitting (Recht et al., 2018; 2019). Importantly, model rankings are preserved, with an approximately linear relationship between original and replicated accuracy. Evidence at a larger scale points in the same direction (Liao et al., 2021; Roelofs et al., 2019).

One explanation for this gap between theory and practice is that the effective number of distinct hypotheses explored during model development is smaller than it appears. Models trained on the same benchmark often produce highly similar predictions (Mania et al., 2019). This reduces the ability of adaptive procedures to exploit test-set noise, even when the test set is reused multiple times. Recent work has begun to formalize this risk for hyperparameter tuning (Schröder et al., 2025; Schneider et al., 2025; Makarova et al., 2021), but no study has isolated its effect when tuning is done directly on the test set of a widely used benchmark. We follow these works in a stronger setting: we give the tuning procedure direct access to the test set.

Test set hyperparameter tuning is common practice. Despite this reassuring empirical picture, tuning and final reporting are usually performed on the same labels, because official test labels are often withheld: ImageNet uses the `val` split as a proxy test set (Russakovsky et al., 2015), and GLUE (Wang et al., 2019) the development split. The official PyTorch example (PyTorch Contributors, 2024) evaluates on `val` every epoch and keeps the best checkpoint, and others present hyperparameter search as an ablation on that same data (Wightman, 2019; Liu et al., 2022). The DeiT (Touvron et al., 2021a;b), BEiT (Bao et al., 2022b;a) and MAE (He et al., 2022a;b) implementations go further, calling `val` the "test" set in their own print statements and metric keys. On GLUE, BERT (Devlin et al., 2019; Google Research, 2018) performs an exhaustive search choosing "the model that performs best on the development set" and, for the smaller tasks, "several random restarts" selecting "the best model on the Dev set", while RoBERTa (Liu et al., 2019; Facebook AI Research, 2019) keeps only the checkpoint that maximises validation accuracy and reports on that same split.

Hyperparameter sensitivity. The extent to which hyperparameter tuning can cause adaptive overfitting depends on how strongly hyperparameters influence accuracy. Sensitivity analysis methods quantify these effects. Taylor et al. (2021) identify learning rate decay and batch size as consistently important. For image classification, a functional ANOVA study of 12 hyperparameters for Residual Neural Networks across 10 datasets (Sharma et al., 2019) shows that individual hyperparameters and their interactions have a substantial effect on accuracy. A complementary analysis over 10,000 CNN configurations (Wang et al., 2024) ranks the most influential factors and highlights the number of convolutional layers, learning rate, dropout rate, optimizer and number of epochs.

Hyperparameter tuning leads to significant improvements. Empirical evidence shows that hyperparameter tuning is a major driver of model performance (Weerts et al., 2020). For example, tuning only learning rate, weight decay and batch size on a validation split results in a baseline that outperforms several specialised methods (Brigato et al., 2021), which suggests that reported gains may partly reflect more thorough hyperparameter optimisation rather than algorithmic innovation. The process is also heavily shaped by human skill: practitioner expertise and the trial-and-error budget significantly dictate final outcomes (Anand et al., 2020). The same property that makes tuning so effective, however, is what creates room for overfitting. Even adaptive search strategies such as Bayesian optimisation, which use prior evaluations to guide where to query next, can substantially overfit hyperparameters to a validation set (Makarova et al., 2021).

Variance in benchmarks. Benchmark accuracy varies due to data sampling, model initialization, training dynamics and hyperparameter optimization. If this variability is not accounted for, comparisons between models can be biased because observed differences may reflect noise rather than systematic effects (Bouthillier et al., 2021). To address this, models can be compared using the probability of outperforming. Let A and B be the performances of two models to be compared. Then $P(A > B)$ can be estimated over repeated runs, where all sources of randomness should be varied. This provides a more robust measure than single-run accuracy and allows uncertainty to be quantified explicitly.

Empirical evidence shows that variability can be substantial. A sweep over up to 10^4 random seeds on CIFAR-10 and ImageNet (Picard, 2021) finds that, despite low average variance, it is easy to obtain outlier seeds with significantly better or worse accuracy. Differences of performance in literature can therefore arise purely from randomness or adaptive overfitting. Here, we question if these benchmark rankings are still valid when hyperparameters are tuned on the test set.

3 METHODOLOGY

We study adaptive overfitting induced by hyperparameter selection under repeated evaluation on a fixed dataset split. Formally, we define adaptive overfitting as the accuracy advantage obtained when hyperparameters are selected using test-set feedback rather than an independent validation set, following Dwork et al. (2015). Given a training pipeline, a hyperparameter search space and a dataset, we estimate adaptive overfitting and draw statistical conclusions.

Problem Setup and Selection Strategies We focus on a classification setting and let D_{train}, D_{val} and D_{test} denote the training, validation and test datasets, respectively. The hyperparameter space for N hyperparameters is Λ, where a configuration $\lambda \in \Lambda$ is an N-dimensional vector, and $A(\lambda, D)$ represents the accuracy of a model trained on D_{train} with hyperparameters λ and evaluated on dataset D. Hyperparameters are selected by sampling T candidate configurations $\Lambda_T = \{\lambda_1, \ldots, \lambda_T\} \subset \Lambda$ via random sampling. We compare two hyperparameter selection strategies: *validation-tuning* selects $\lambda^*_{\text{val}} = \arg\max_{\lambda \in \Lambda_T} A(\lambda, D_{\text{val}})$, while *test-tuning* selects $\lambda^*_{\text{test}} = \arg\max_{\lambda \in \Lambda_T} A(\lambda, D_{\text{test}})$, where we carefully pair the validation-tuned and the test-tuned runs so that the only source of variation is in the evaluation dataset D_{val} vs D_{test}.

Adaptive Overfitting Gap. We quantify adaptive overfitting through the *adaptive overfitting gap*,

$$\Delta_{\text{AO}} \;=\; A(\lambda^*_{\text{test}}, D_{\text{test}}) \;-\; A(\lambda^*_{\text{val}}, D_{\text{test}}), \tag{1}$$

the excess test accuracy attributable to hyperparameter selection with direct knowledge of D_{test}. Because λ^*_{test} maximises $A(\lambda, D_{\text{test}})$ over the same pool Λ_T from which λ^*_{val} is drawn, $\Delta_{\text{AO}} \geq 0$ in every paired run, with equality when both rules pick the same configuration (or two configurations that score identically on D_{test}). Following Bouthillier et al. (2021), we additionally report

$$\hat{P}(\Delta_{\text{AO}} > 0) \;:=\; P(\Delta_{\text{AO}} > 0) + \tfrac{1}{2}\, P(\Delta_{\text{AO}} = 0), \tag{2}$$

the empirical win probability of test-tuning, where ties contribute 0.5. We estimate it together with a $95\,\%$ percentile-bootstrap confidence interval over the K paired runs, and call adaptive overfitting *significant* when the lower bound of this Confidence Interval (CI) exceeds 0.5.

Paired Run Design. Each of the K paired (val-tuned, test-tuned) training runs executes the following pipeline:

1. sample an independent bootstrap split $(D_{\text{train}}, D_{\text{val}}, D_{\text{test}})$ (Appendix I describes the splitting scheme),
2. draw T independent hyperparameter configurations Λ_T via random sampling,
3. for each configuration $\lambda \in \Lambda_T$, train a model on D_{train} (from random initialisation for the image benchmarks, and by fine-tuning a pretrained checkpoint for the GLUE tasks) and record $A(\lambda, D_{\text{val}})$ and $A(\lambda, D_{\text{test}})$,
4. select λ^*_{val} and λ^*_{test} from the same pool of T trained models and compute Δ_{AO}.

Because of the paired run design, both selection strategies operate on the same bootstrap split, so any difference between λ^*_{val} and λ^*_{test} isolates the effect of the selection rule alone. We use $K = 30$ paired runs per model, the minimum needed for the Noether-bound paired-comparison guarantee (Bouthillier et al., 2021; Noether, 1987).

Following Bouthillier et al. (2021), every experiment varies the major sources of randomness under our control (data splits, hyperparameter samples, initialisation seeds) so that the empirical distribution of Δ_{AO} reflects the full pipeline variance rather than a single favorable configuration.

4 Experiments

We investigate overfitting by tuning hyperparameters directly on the test set, i.e., using the same dataset for hyperparameter selection and accuracy estimation. We first investigate model sensitivity to overfitting and how severe the overfitting could be. Then, we evaluate if tuning on the test set leads to different model selection compared to tuning on the validation set.

Datasets. We evaluate on two image classification benchmarks and three natural language understanding tasks. The controlled benchmark is MNIST-1D (Greydanus & Kobak, 2024), a 1D-signal variant of MNIST designed for controlled experiments in machine learning. We use a 4,000-example source set bootstrapped per training run, and test a family of eight fully-connected networks spanning $1.3\,\text{k}$ to $2{\times}10^5$ parameters (Appendix D), trained with Adam for 8,000 steps. We optimize learning rate, learning-rate decay rate (set to zero with probability 0.3), weight decay, and the random seed used to initialize the network weights.

For CIFAR-10 we combine the training and test sets into a single 60,000-example pool, from which each training run draws an independent bootstrap split. We use 8 architectures from the CIFAR-10 leaderboard (Appendix D), covering ResNets (He et al., 2016), Shake-Shake (Gastaldi, 2017), VGG (Simonyan & Zisserman, 2015), MobileNetV2 (Sandler et al., 2018) and ResNet-9 (Page, 2019). We tune learning rate, weight decay, momentum, number of training steps and the random seed used to initialise the network weights, using Nesterov SGD (Sutskever et al., 2013) with cosine-annealed learning rate.

We additionally evaluate three classification datasets from the GLUE benchmark (Wang et al., 2019): RTE (textual entailment), MRPC (paraphrase detection) and CoLA (grammatical acceptability). The official GLUE test splits are unlabeled and withheld on the leaderboard server, so we combine each dataset's train and validation sets into their own single pool (RTE: $2{,}767$, MRPC: $4{,}076$ and CoLA: $9{,}594$ examples), from which each training run draws an independent bootstrap split. To save space we report MRPC in the main text and RTE and CoLA in Appendix A since all three behave alike.

We fine-tune eight pretrained transformers spanning 11.8M to 141.9M parameters (Appendix D) with AdamW and a linear warmup-decay schedule, tuning the fine-tuning hyperparameters: learning rate, weight decay, warmup ratio, number of epochs and the initialization seed. We score all three by accuracy, so that one metric applies across all five benchmarks.

Experimental settings. We use $K = 30$ paired training runs per architecture, where we re-sample the train, val, and test sets differently for each of the K runs. The hyperparameter search budget per run is $T = 30$, which is in line with the number of configurations typically evaluated in deep learning architecture training (Hastie et al., 2009). We select the best accuracy over the T randomly sampled

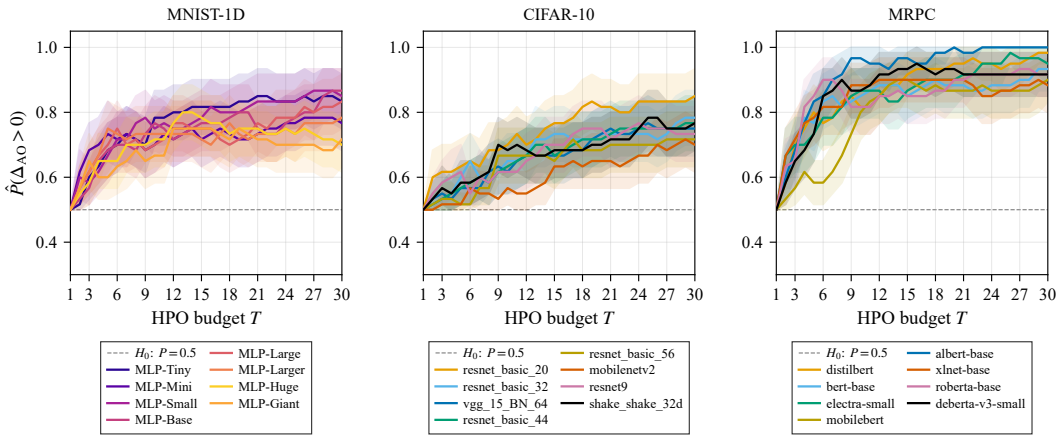


Figure 1: Overfitting sensitivity: the tie-broken win probability of test-tuning, $\hat{P}(\Delta_{\mathrm{AO}} > 0)$, as a function of the tuning budget T. Each line is one architecture, with its 95 % percentile-bootstrap CI shaded. A line whose CI lower bound clears the dashed threshold at 0.5 shows statistically significant adaptive overfitting at that budget. MNIST-1D and MRPC clear it at around T=3, CIFAR-10 at around T=13. RTE and CoLA are in Appendix A.

different hyperparameter configurations. All randomness derives from a single master seed. Hyperparameter search spaces are in Appendix C and additional implementation details in Appendix B. In total this yields $8 \times 30 \times 30 = 7{,}200$ trained models per benchmark, or 36,000 across the five benchmarks. The full set of 36,000 trained models and 72,000 validation/test set evaluations used approximately 8,000 GPU-hours.

As a sanity check to validate if overfitting is possible, we compute an upper bound by training and tuning models directly on the test data. Every architecture nearly memorizes its own test fold, achieving exactly 100% accuracy on both image benchmarks, and between 98.9% and 100% accuracy on the GLUE tasks, whose test folds are smaller (Appendix F).

4.1 Exp 1: Overfitting sensitivity and overfitting severity

For tuning hyperparameters on the test set, we investigate overfitting sensitivity: how easy it is to overfit, and overfitting severity: how much a model overfits.

Overfitting sensitivity. To quantify how easily a model overfits, we compare the hyperparameter search budget T against the empirical win probability $\hat{P}(\Delta_{\mathrm{AO}} > 0)$ and its 95 % bootstrap CI. The rationale is that a model that overfits easily reaches significance, i.e. $\mathrm{CI}_{\mathrm{lower}} > 0.5$, at a smaller budget T.

Results in Figure 1 show that all models reach significance easily, needing only around T=3 different hyperparameter configurations on MNIST-1D and MRPC, and around T=13 on CIFAR-10. RTE and CoLA behave like MRPC (Appendix A). Note that $\hat{P}(\Delta_{\mathrm{AO}} > 0)$ is not strictly increasing with T, because a larger budget also benefits validation-tuning, which can in turn lower $\hat{P}(\Delta_{\mathrm{AO}} > 0)$.

A substantial fraction of runs have $\Delta_{\mathrm{AO}} = 0$: validation- and test-tuning select the same configuration, so test-tuning gains nothing. Table 1 shows this fraction is high on the image benchmarks (41.2% on MNIST-1D, 48.3% on CIFAR-10) and lower on MRPC (12.9%), which is also where the gap is largest relative to run-to-run noise. On every benchmark this effect is overwhelmingly caused by both selection strategies picking the same hyperparameter configurations instead of accidental ties between distinct configurations. When one configuration wins on both splits, test-tuning has nothing to gain.

Overfitting severity. To evaluate how much a model overfits, we compare the mean Δ_{AO} over the K=30 paired runs against the natural run-to-run variability of the same architecture, measured as the standard deviation of the validation-tuned test accuracy across the same K=30 runs. If Δ_{AO} is

Benchmark	$\Delta_{\mathrm{AO}} = 0$	Percentage	Identical configuration
MNIST-1D	99/240	41.2%	98.0%
CIFAR-10	116/240	48.3%	98.3%
MRPC	31/240	12.9%	90.3%

Table 1: Runs with $\Delta_{\mathrm{AO}} = 0$ pooled over the eight architectures of each benchmark ($K = 30$ runs each), as a count and as a percentage of all runs. The last column is the share of runs with $\Delta_{\mathrm{AO}} = 0$ in which validation-tuning and test-tuning selected the identical configuration. The remainder are distinct configurations that happen to score identically on the test split. RTE and CoLA are in Appendix A.

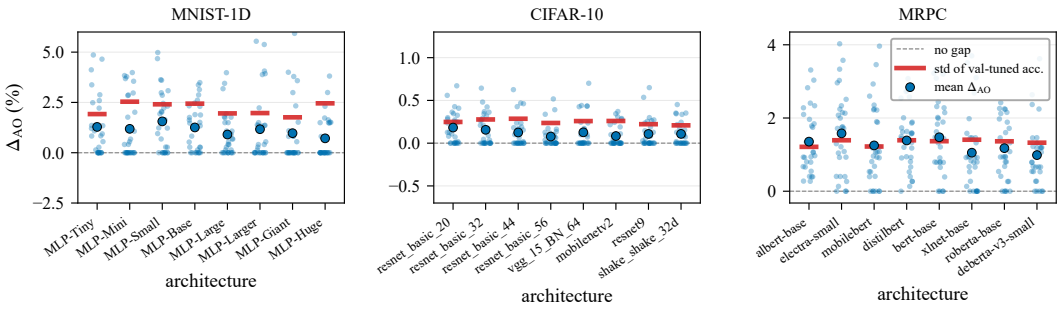


Figure 2: Overfitting severity: per-run adaptive overfitting gap. Each translucent dot is one of the $K = 30$ paired runs of an architecture; the filled blue circle is the per-architecture mean Δ_{AO}. The red horizontal bar depicts one standard deviation of that architecture's val-tuned test accuracy, representing run-to-run noise. On nearly every architecture the mean Δ_{AO} sits below the standard deviation, showing that the gap is small relative to natural run-to-run variability. RTE and CoLA are in Appendix A.

small relative to this experimental noise, the apparent overfitting is no larger than what re-running the pipeline under different data splits and initializations would already produce.

Results in Figure 2 show that on 20 of the 24 architecture–benchmark pairs, the mean gap is smaller than the run-to-run standard deviation of the validation-tuned models. The margin is widest on the image benchmarks, where the gap is roughly half the standard deviation and never exceeds 0.74 of it, and narrowest on MRPC, where four of the eight architectures sit marginally above it, up to 1.13. We conclude that overfitting severity, in our setup, is comparable to the natural experimental variability in the experimental design.

Robustness checks. Two additional experiments test whether these findings depend on the search budget, the split sizes, or the particular test fold.

First, we reran MLP-Giant on MNIST-1D with $T = 100$ trials per run (a $3\times$ larger budget) and larger fixed splits of 10,000/10,000/10,000 drawn from a 70,000-sample MNIST-1D set, denoted MLP-Giant$_{T=100}$. Both findings hold at this budget: $\hat{P}(\Delta_{\mathrm{AO}} > 0) = 0.73$ at T=100 with the CI lower bound well above 0.5, and 16 of 30 runs (53%) still have $\Delta_{\mathrm{AO}} = 0$, essentially unchanged from the 17/30 (57%) in the main setup, with the two rules selecting the same configuration in all 16. The large fraction of runs with $\Delta_{\mathrm{AO}} = 0$ is therefore structural rather than caused by the small search budget (Appendix G).

Secondly, we re-evaluated the selected CIFAR-10 configurations on CIFAR-10.1 (Recht et al., 2019), a 2,000-image replication of CIFAR-10 drawn from the same source distribution. On this independent set the val-tuned and test-tuned configurations score within run-to-run noise of each other, so the gap measured on the bootstrap fold does not carry over. Tuning directly on CIFAR-10.1 beats both by roughly 0.4%, showing what a procedure with access to it could gain (Appendix H).

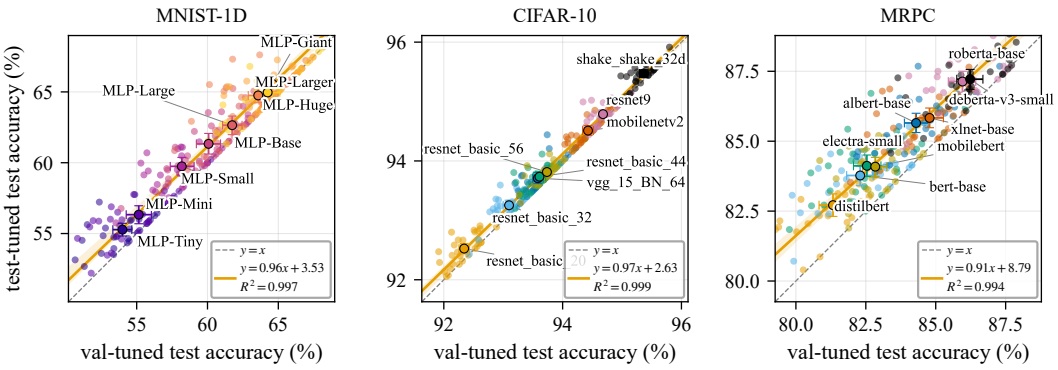


Figure 3: Validation-tuned vs. test-tuned mean test accuracy. Markers are per-architecture means with 95 % bootstrap confidence intervals; dots are the individual paired runs. The dashed grey line is the identity $y = x$ and the orange line is the ordinary least squares fit through the eight architecture means, with its 95 % pairs-bootstrap confidence interval shaded. On every benchmark the means lie close to a line nearly parallel to the identity, so test-tuning acts as a near-uniform upward shift that preserves architecture rankings. RTE and CoLA are in Appendix A.

4.2 Exp 2: How much does overfitting influence model selection?

Here, we investigate if overfitting leads to different model selection when compared to tuning hyperparameters on a validation set. We evaluate model selection by ranking models, as commonly done on a leaderboard.

For each experiment, we fit test-tuned on validation-tuned test accuracy by ordinary least squares on the per-architecture means, and report Spearman and Kendall rank correlations on those same means. All confidence intervals are based on 100,000 bootstrap resamples. For the linear fit we use a standard pairs bootstrap, resampling the eight architecture mean pairs with replacement. The rank correlations resample the K runs within each architecture.

Figure 3 shows the ordinary least squares fit between validation-tuned and test-tuned test accuracies, with per-architecture rankings in Table 2. The slope is close to one (0.91–0.97) and the fit is tight ($R^2 \geq 0.99$). Test-tuning acts as a near-uniform upward shift and does not reorder the ranking. The architecture ranking is identical under both selection rules on MNIST-1D and CIFAR-10 (Spearman $\rho = 1.00$, Kendall $\tau = 1.00$). On MRPC a single adjacent pair exchanges places ($\rho = 0.976$, $\tau = 0.929$), between two architectures differing by 0.30% against a run-to-run standard deviation of 1.34%, so they are statistically indistinguishable under either rule. RTE and CoLA show the same pattern (Appendix A). For all benchmarks and all models, adaptive overfitting does better in the *absolute* accuracies, but not in the *relative* comparison: a leaderboard built from val-tuned numbers selects the same winner as a leaderboard built from test-tuned numbers.

5 Discussion and Limitations

The experiments tell a consistent story across five benchmarks, two modalities and three model families. It is easy to overfit by tuning on the test set, but the severity stays within the range of natural experimental variability such as random dataset splits and training randomness. The mean gap is roughly half the run-to-run standard deviation on the image benchmarks and between 0.7 and 1.0 times it on the GLUE tasks. Test-tuning yields a consistent absolute improvement without reordering models: rankings are identical under both selection rules on MNIST-1D, CIFAR-10 and RTE, and on MRPC and CoLA the only change is a single adjacent inversion between architectures separated by far less than run-to-run noise. The top-ranked model is unchanged everywhere.

A key observation that explains why the magnitude stays small is the fraction of runs with an adaptive overfitting gap of 0, where validation and test tuning select exactly the same configuration and test-tuning gains nothing. This fraction is high on the image benchmarks (41.2% on MNIST-1D, 48.3% on CIFAR-10) and lower on the GLUE tasks (12.9%–25.8%), which are also where the gap

Val. Rank	Model	Validation-tuned (%)	Test-tuned (%)	Δ_{AO}	Val. Std. (%)	ΔRank
		MNIST-1D				
1	MLP-Giant	64.59	65.56	+0.97	1.76	0
2	MLP-Huge	64.25	64.96	+0.72	2.46	0
3	MLP-Larger	63.58	64.75	+1.17	1.97	0
4	MLP-Large	61.74	62.65	+0.91	1.96	0
5	MLP-Base	60.08	61.33	+1.26	2.44	0
6	MLP-Small	58.19	59.74	+1.56	2.41	0
7	MLP-Mini	55.15	56.33	+1.19	2.54	0
8	MLP-Tiny	53.99	55.28	+1.28	1.92	0
		CIFAR-10				
1	shake_shake_32d	95.36	95.47	+0.11	0.21	0
2	resnet9	94.68	94.79	+0.11	0.22	0
3	mobilenetv2	94.43	94.51	+0.08	0.26	0
4	resnet_basic_56	93.74	93.81	+0.08	0.24	0
5	resnet_basic_44	93.61	93.73	+0.12	0.29	0
6	vgg_15_BN_64	93.58	93.71	+0.13	0.26	0
7	resnet_basic_32	93.10	93.25	+0.16	0.28	0
8	resnet_basic_20	92.34	92.52	+0.18	0.25	0
		MRPC				
1	deberta-v3-small	86.23	87.22	+0.99	1.33	0
2	roberta-base	85.96	87.13	+1.18	1.36	0
3	xlnet-base	84.78	85.83	+1.05	1.41	0
4	albert-base	84.30	85.65	+1.35	1.21	0
5	mobilebert	82.84	84.09	+1.25	1.22	−1
6	electra-small	82.54	84.12	+1.58	1.39	+1
7	bert-base	82.30	83.77	+1.47	1.37	0
8	distilbert	81.33	82.71	+1.39	1.39	0

Table 2: Model rankings ordered by validation-tuned accuracy, with the adaptive overfitting difference and the run-to-run standard deviation of validation-tuned accuracy. Several models within a benchmark are close together and have near-similar accuracy. On MNIST-1D and CIFAR-10 no model changes rank when tuned on the test set rather than the validation set. On MRPC exactly one adjacent pair exchanges places, between models separated by far less than the run-to-run standard deviation, so they are statistically indistinguishable under either rule. The top-ranked model is unchanged. RTE and CoLA are in Appendix A.

is largest relative to run-to-run noise. The cause is structural: within a run, one configuration tends to win on both the validation and test split, and this dominance weakens on the small GLUE tasks, where the evaluation splits are small and fine-tuning is noisy, so the two rankings agree less often. We confirmed this by rerunning MLP-Giant on MNIST-1D with $T = 100$ trials and larger validation and test sets: the fraction of runs with $\Delta_{\mathrm{AO}} = 0$ stayed nearly unchanged at 53%. This aligns with Mania et al. (2019), who attribute low adaptive overfitting to high prediction similarity across models. Our finding is specific to a single hyperparameter search but reflects the same: when one configuration dominates, adaptive selection has little to exploit.

The results agree with the empirical literature. Recht et al. (2018; 2019) and Roelofs et al. (2019) both find that adaptive overfitting on popular benchmarks is small and does not change model rankings. Our work extends this by directly measuring adaptive overfitting for specific experiments and architectures in a controlled setting, compared to previous work which infers it from leaderboard differences. Our findings make a pattern clear: a substantial fraction of runs produce a zero gap, the rest a small positive one. Compared to the worst-case theoretical bounds from adaptive data analysis (Dwork et al., 2015), the observed effect is smaller, which is expected given that those bounds are conservative: they hold against an adversarial analyst that can adapt each query to prior test feedback, whereas random hyperparameter search draws queries independently and produces highly correlated trials, so the effective number of independent test set queries is much smaller than T.

In practice, this means that reusing the test set for hyperparameter tuning under random search at this budget produces a measurable but very small accuracy inflation, on image classification and natural language understanding alike.

5.1 LIMITATIONS

Random search is a weak adaptive querier. The methodology uses random search, which evaluates each hyperparameter configuration independently. In practice, methods such as Bayesian optimization, hyperband Li et al. (2018), or manual iterative tuning reuse results from earlier evaluations, and could produce larger adaptive overfitting gaps.

Small hyperparameter budget. Each run uses $T = 30$ samples, which is in line with search budgets typically used in practice Hastie et al. (2009), but is still small in absolute terms. Additionally, the batch size is kept fixed and not tuned. A follow-up on MNIST-1D with $T = 100$ random-search trials showed no meaningful difference, but behavior at much larger compute budgets is not explored.

Limited domain coverage. The larger GLUE tasks, other modalities, and regression are not tested. Liao et al. (2021) suggest that adaptive overfitting may be more severe for regression.

Per-procedure rather than cumulative. Each Δ_{AO} estimate covers one hyperparameter search. In practice, benchmark inflation builds up across many groups reusing the same test set over time. Increasing T within our experiments only adds more queries from the same procedure and does not capture the adaptive selection across the deep learning community (architecture choices, publication filtering, etc.) that drives the community-level inflation studied by Recht et al. (2019); Roelofs et al. (2019).

Wide confidence intervals. $K{=}30$ paired runs is the minimum recommended by Bouthillier et al. (2021) via the Noether bound (Noether, 1987), so the bootstrap intervals around $\hat{P}(\Delta_{\mathrm{AO}} > 0)$ are wide, around 0.10–0.17. The weakest architectures have lower bounds near 0.60, so their individual results are not robust, but the overall pattern of small positive gaps does not depend on them.

Only visual and textual models evaluated. We only evaluate common deep learning architectures, limited to the image and text domain. We did not explore other machine learning algorithms.

6 CONCLUSION

We presented a paired-run methodology for directly measuring adaptive overfitting from hyperparameter tuning, applied to a controlled MNIST-1D family of 8 MLPs, a realistic suite of 8 CIFAR-10 architectures, and three small GLUE tasks each fine-tuning 8 pretrained transformers, for $36{,}000$ trained models in total. We investigate how easy and severe test-tuning is and show statistically significant adaptive overfitting on all 40 architecture–benchmark combinations. The effect was small in absolute terms, and within the range of the natural experimental variation. A large fraction of runs had no overfitting gap, where validation and test tuning selected the same configuration: 41.2% of runs on MNIST-1D and 48.3% on CIFAR-10, and a smaller but still substantial 12.9%–25.8% on the GLUE tasks. This zero-gap phenomenon persisted under a $3\times$ larger search budget and larger splits, indicating that increasing the search budget does not substantially change the overlap between validation and test selections at this scale. We also investigate whether test-tuning changes which model would be selected, where it behaved as a near-uniform upward shift: the linear fit through the per-architecture means had slope close to one (0.91–1.02) and $R^2 \geq 0.99$ on every benchmark, and rank correlations were exactly 1 on MNIST-1D, CIFAR-10 and RTE. On MRPC and CoLA a single adjacent pair exchanged places, in both cases between architectures separated by far less than run-to-run noise, and the top-ranked model was unchanged everywhere. Taken together, the experiments suggest that for honest random-search hyperparameter tuning, the practical risk of test-set reuse is much smaller than worst-case theory predicts, and comparing accuracies under test-tuning does not change which model a leaderboard would select. These empirical results allow researchers to discuss, mention, and defend tuning hyperparameters on the test set, leading to a more open scientific discourse.

AI USE STATEMENT

In this work, we used generative AI tools for coding support, such as refactoring and debugging. We have not used generative AI tools for generating research ideas, designing experiments, producing results, or creating data, and other required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to improve the clarity of the writing and to correct grammatical errors. We have reviewed all AI-assisted work. AI-assisted code was manually verified and tested for correctness, and all AI-suggested edits to the writing were checked against the original intent and meaning before acceptance. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies a methodological issue in machine learning evaluation, namely how strongly tuning hyperparameters on the test set inflates reported accuracy. It does not introduce new models or datasets, and we do not anticipate direct negative societal consequences from its release. If anything, a clearer empirical picture of when test-set tuning matters and when it does not should help researchers and reviewers assess published results more accurately, reducing the risk that benchmark numbers are taken at face value without questioning the selection protocol behind them. All experiments are run on standard public datasets (MNIST-1D, CIFAR-10 and three GLUE tasks) on shared academic cluster hardware, and the released code and analysis scripts are intended to make the findings independently verifiable.

REPRODUCIBILITY STATEMENT

All code, experiment configurations, and analysis scripts required to reproduce our results are available at `https://anonymous.4open.science/r/hyperparameter-tuning-on-test-set/README.md`. All randomness derives from a single master seed: NumPy's `SeedSequence` splits it into independent sub-seeds, one per architecture, per run, and per source of randomness within a run (bootstrap split, hyperparameter sampling – including the per-trial initialisation seed – and other training stochasticity). No two sub-seeds coincide, so the full experiment is reproducible end-to-end from that single seed. The paired-run protocol, the two selection rules, and the definition of Δ_{AO} are given in Section 3, and the experimental settings, budgets, and compute environment in Section 4. Dataset construction and the out-of-bootstrap splitting scheme are detailed in Appendix I, the architectures and their reference implementations in Appendix D, the hyperparameter search spaces in Appendix C, and the training setup in Appendix B. Appendix E reports the exact bootstrap settings and full confidence intervals behind the ranking analysis, and Appendix J the empirical winner distributions used to verify that the search ranges are well-shaped. The supporting analyses are likewise fully specified: the memorisation upper bound in Appendix F, the larger-budget MLP-Giant$_{T=100}$ rerun in Appendix G, and the CIFAR-10.1 transfer evaluation in Appendix H.

REFERENCES

Kanav Anand, Ziqi Wang, Marco Loog, and Jan van Gemert. Black magic in deep learning: How human skill impacts network training. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2020. URL `https://arxiv.org/abs/2008.05981`.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *International Conference on Learning Representations (ICLR 2022)*, 2022a. URL `https://openreview.net/forum?id=p-BhZSz59o4`.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. `https://github.com/microsoft/unilm/tree/master/beit`, 2022b. Accessed: 2026-05-06.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Dmitriy Serdyuk, Tal Arbel, Chris

Pal, Gaël Varoquaux, and Pascal Vincent. Accounting for variance in machine learning benchmarks. In *Proceedings of Machine Learning and Systems 3 (MLSys 2021)*, 2021. URL https://proceedings.mlsys.org/paper_files/paper/2021/hash/0184b0cd3cfb185989f858a1d9f5c1eb-Abstract.html.

Lorenzo Brigato, Björn Barz, Luca Iocchi, and Joachim Denzler. Tune it or don't use it: Benchmarking data-efficient image classification. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 1071–1080, 2021. doi: 10.1109/ICCVW54120.2021.00125.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations (ICLR)*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping, 2020. URL https://arxiv.org/abs/2002.06305.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pp. 2350–2358, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/bad5f33780c42f2588878a9d07405083-Abstract.html.

Facebook AI Research. fairseq: Fine-tuning RoBERTa on GLUE tasks. https://github.com/facebookresearch/fairseq/blob/main/examples/roberta/README.glue.md, 2019. Accessed 2026.

Xavier Gastaldi. Shake-shake regularization of 3-branch residual networks. In *International Conference on Learning Representations Workshop (ICLR 2017)*, 2017. URL https://openreview.net/forum?id=HkO-PCmYl.

Google Research. BERT: TensorFlow code and pre-trained models. https://github.com/google-research/bert, 2018. Accessed 2026.

Samuel James Greydanus and Dmitry Kobak. Scaling down deep learning with MNIST-1D. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16404–16415. PMLR, 2024. URL https://proceedings.mlr.press/v235/greydanus24a.html.

Moritz Hardt. *The Emerging Science of Machine Learning Benchmarks*. Princeton University Press, 2026. URL https://mlbenchmarks.org.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, New York, NY, 2 edition, 2009. ISBN 978-0-387-84857-0. doi: 10.1007/978-0-387-84858-7.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders: A PyTorch implementation. https://github.com/facebookresearch/mae, 2022a. Accessed: 2026-05-06.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15979–15988, 2022b. doi: 10.1109/CVPR52688.2022.01553.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations (ICLR)*, 2023.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR)*, 2020.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. URL https://jmlr.org/papers/v18/16-558.html.

Thomas I. Liao, Rohan Taori, Deborah Raji, and Ludwig Schmidt. Are we learning yet? a meta review of evaluation failures across machine learning. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1 (NeurIPS Datasets and Benchmarks 2021), Round 2*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/757b505cfd34c64c85ca5b5690ee5293-Abstract-round2.html.

Zachary C. Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship. *Communications of the ACM*, 62(6):45–53, 2019. doi: 10.1145/3317287.3328534. URL https://dl.acm.org/doi/10.1145/3317287.3328534.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. ConvNeXt official PyTorch implementation. https://github.com/facebookresearch/ConvNeXt, 2022. Accessed: 2026-05-06.

Anastasiia Makarova, Huibin Shen, Valerio Perrone, Aaron Klein, Jean Baptiste Faddoul, Andreas Krause, Matthias Seeger, and Cédric Archambeau. Overfitting in Bayesian optimization: An empirical study and early-stopping solution. In *ICLR 2021 Workshop on Neural Architecture Search*, 2021. URL https://openreview.net/forum?id=SKUHN5b3PmG.

Horia Mania, John Miller, Ludwig Schmidt, Moritz Hardt, and Benjamin Recht. Model similarity mitigates test set overuse. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pp. 9993–10002, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/48237d9f2dea8c74c2a72126cf63d933-Paper.pdf.

Gottfried E. Noether. Sample size determination for some common nonparametric tests. *Journal of the American Statistical Association*, 82(398):645–647, 1987. doi: 10.1080/01621459.1987.10478478. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1987.10478478.

David Page. How to train your ResNet 8: Bag of tricks. https://myrtle.ai/learn/how-to-train-your-resnet-8-bag-of-tricks/, 2019. Blog post, Myrtle.ai.

David Picard. Torch.manual_seed(3407) is All You Need: On the Influence of Random Seeds in Deep Learning Architectures for Computer Vision. *CoRR*, abs/2109.08203, 2021. doi: 10.48550/arXiv.2109.08203. URL https://arxiv.org/abs/2109.08203.

PyTorch Contributors. PyTorch examples: ImageNet training in PyTorch. https://github.com/pytorch/examples/tree/main/imagenet, 2024. Accessed: 2026-04-21.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do CIFAR-10 classifiers generalize to cifar-10? *CoRR*, abs/1806.00451, 2018. URL http://arxiv.org/abs/1806.00451.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5389–5400. PMLR, 2019. URL https://proceedings.mlr.press/v97/recht19a.html.

Rebecca Roelofs, Sara Fridovich-Keil, John Miller, Vaishaal Shankar, Moritz Hardt, Benjamin Recht, and Ludwig Schmidt. A meta-analysis of overfitting in machine learning. In *Advances in Neural Information Processing Systems 32*, 2019. URL https://proceedings.neurips.cc/paper/2019/file/ee39e503b6bedf0c98c388b7e8589aca-Paper.pdf. 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. URL https://link.springer.com/article/10.1007/s11263-015-0816-y.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018. doi: 10.1109/CVPR.2018.00474.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing, NeurIPS*, 2019.

Lennart Schneider, Bernd Bischl, and Matthias Feurer. Overtuning in hyperparameter optimization. In *Proceedings of the Fourth International Conference on Automated Machine Learning*, volume 293 of *Proceedings of Machine Learning Research*, pp. 17/1–43. PMLR, 2025. URL https://proceedings.mlr.press/v293/schneider25a.html.

Sietse Schröder, Mitra Baratchi, and Jan N. van Rijn. Overfitting in combined algorithm selection and hyperparameter optimization. In *Advances in Intelligent Data Analysis XXIII (IDA 2025)*, volume 15669 of *Lecture Notes in Computer Science*, pp. 181–194, Cham, 2025. Springer. doi: 10.1007/978-3-031-91398-3_14. URL https://link.springer.com/chapter/10.1007/978-3-031-91398-3_14.

D. Sculley, Jasper Snoek, Alex Wiltschko, and Ali Rahimi. Winner's curse? On pace, progress, and empirical rigor. In *ICLR 2018 Workshop on Reproducibility in Machine Learning*, 2018. URL https://openreview.net/forum?id=rJWF0Fywf.

Abhinav Sharma, Jan N. van Rijn, Frank Hutter, and Andreas Müller. Hyperparameter importance for image classification by residual neural networks. In *Discovery Science (DS 2019)*, volume 11828 of *Lecture Notes in Computer Science*, pp. 112–126, Cham, 2019. Springer. doi: 10.1007/978-3-030-33778-0_10. URL https://link.springer.com/chapter/10.1007/978-3-030-33778-0_10.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR 2015)*, 2015. URL https://arxiv.org/abs/1409.1556.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2158–2170, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.195. URL https://aclanthology.org/2020.acl-main.195/.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/sutskever13.html.

Ross Taylor, Varun Ojha, Ivan Martino, and Giuseppe Nicosia. Sensitivity analysis for deep learning: Ranking hyper-parameter influence. In *33rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2021. doi: 10.1109/ICTAI52525.2021.00083. URL https://ieeexplore.ieee.org/document/9643336/.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. DeiT: Data-efficient image transformers. https://github.com/facebookresearch/deit, 2021a. Accessed: 2026-05-06.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 2021b. URL https://proceedings.mlr.press/v139/touvron21a.html.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*, 2019.

Ruinan Wang, Ian Nabney, and Mohammad Golbabaee. Efficient hyperparameter importance assessment for CNNs. In *Neural Information Processing (ICONIP 2024)*, volume 2282 of *Communications in Computer and Information Science*, pp. 16–31, Singapore, 2024. Springer. doi: 10.1007/978-981-96-6948-6_2. URL https://link.springer.com/chapter/10.1007/978-981-96-6948-6_2.

Hilde J. P. Weerts, Andreas C. Mueller, and Joaquin Vanschoren. Importance of tuning hyperparameters of machine learning algorithms, 2020. URL https://arxiv.org/abs/2007.07588.

Ross Wightman. timm: PyTorch image models. https://github.com/huggingface/pytorch-image-models, 2019. Accessed: 2026-05-06.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

A ADDITIONAL GLUE DATASETS: RTE AND COLA

RTE and CoLA follow exactly the same protocol as MRPC in the main text.

Overfitting sensitivity. Both datasets reach significance at around $T{=}3$ (Figure 4), matching MNIST-1D and MRPC and well below CIFAR-10's $T{=}13$. Adaptive overfitting is significant for all 16 architecture–dataset pairs.

Runs with no gap. Table 3 gives the fraction of runs with $\Delta_{\mathrm{AO}} = 0$. As in the main text, the overwhelming majority of these happen because both selection rules pick the identical configuration rather than through accidental ties.

Benchmark	$\Delta_{\mathrm{AO}} = 0$	Percentage	Identical configuration
RTE	$62/240$	25.8%	95.2%
CoLA	$48/240$	20.0%	95.8%

Table 3: Runs with $\Delta_{\mathrm{AO}} = 0$ on RTE and CoLA, pooled over the eight architectures of each dataset ($K = 30$ runs each). Columns are as in Table 1.

Overfitting severity. On 15 of the 16 architecture–dataset pairs the mean gap is smaller than the run-to-run standard deviation of the validation-tuned models (Figure 5). The single exception is distilbert on RTE, at 1.25 times the standard deviation, which is the largest ratio observed anywhere in the paper. On CoLA no architecture exceeds 0.89.

Model selection. The linear fit is tight on both datasets (Figure 6): slope 0.916 with $R^2 = 0.997$ on RTE, and slope 1.020 with $R^2 = 0.992$ on CoLA. Rankings are identical under both selection rules on RTE (Spearman $\rho = 1.00$, Kendall $\tau = 1.00$). CoLA has a single adjacent inversion ($\rho = 0.976$, $\tau = 0.929$) between `xlnet-base` and `distilbert`, whose validation-tuned means differ by 0.14% against a run-to-run standard deviation of 0.88%, so they are statistically indistinguishable under either rule. The top-ranked model is unchanged on both datasets. Per-architecture results are in Table 4.

Val. Rank	Model	Validation-tuned (%)	Test-tuned (%)	Δ_{AO}	Val. Std. (%)	ΔRank
		RTE				
1	`roberta-base`	73.62	74.88	+1.26	1.88	0
2	`deberta-v3-small`	73.08	74.65	+1.57	1.75	0
3	`xlnet-base`	68.53	70.45	+1.92	2.50	0
4	`albert-base`	68.35	70.03	+1.68	1.92	0
5	`bert-base`	67.85	69.60	+1.75	2.43	0
6	`mobilebert`	66.24	68.12	+1.88	2.73	0
7	`electra-small`	64.97	66.52	+1.55	2.61	0
8	`distilbert`	59.06	61.87	+2.81	2.25	0
		CoLA				
1	`deberta-v3-small`	86.22	86.78	+0.56	0.93	0
2	`roberta-base`	84.97	85.54	+0.57	0.76	0
3	`electra-small`	83.38	84.23	+0.85	0.96	0
4	`bert-base`	82.90	83.61	+0.71	0.81	0
5	`mobilebert`	82.38	83.02	+0.64	0.91	0
6	`albert-base`	81.96	82.66	+0.70	0.97	0
7	`xlnet-base`	80.36	80.55	+0.20	0.91	−1
8	`distilbert`	80.22	80.91	+0.69	0.81	+1

Table 4: Model rankings on RTE and CoLA, ordered by validation-tuned accuracy. Columns are as in Table 2. On RTE no model changes rank; on CoLA one adjacent pair exchanges places, between models separated by far less than the run-to-run standard deviation. The top-ranked model is unchanged on both.

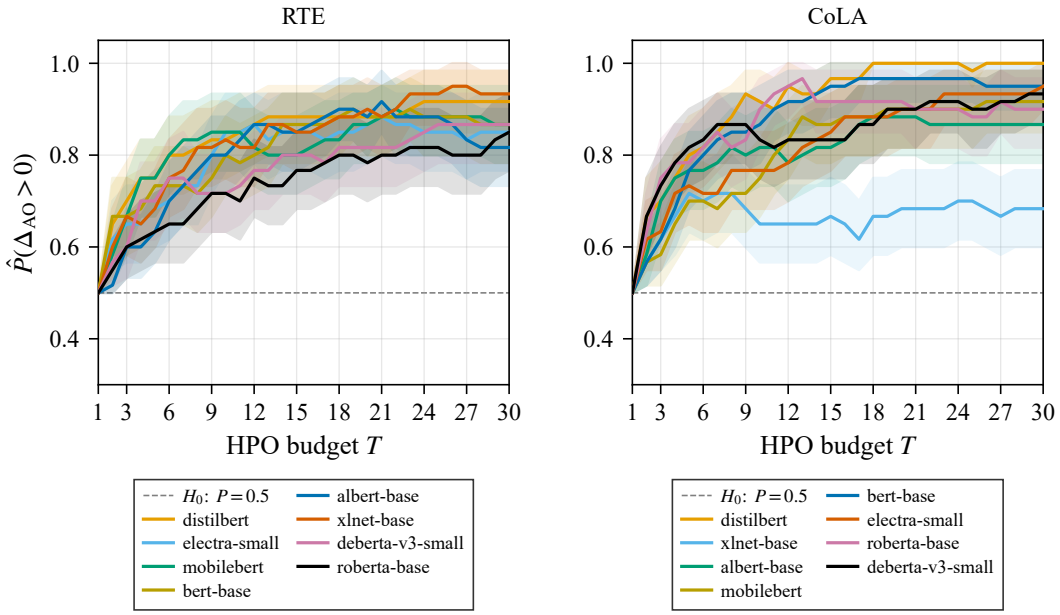


Figure 4: Overfitting sensitivity on RTE and CoLA: the tie-broken win probability of test-tuning, $\hat{P}(\Delta_{\mathrm{AO}} > 0)$, as a function of the tuning budget T. Each line is one architecture, with its 95 % percentile-bootstrap CI shaded. A line whose CI lower bound clears the dashed threshold at 0.5 shows statistically significant adaptive overfitting at that budget. Both datasets clear it at around T=3. Compare Figure 1.

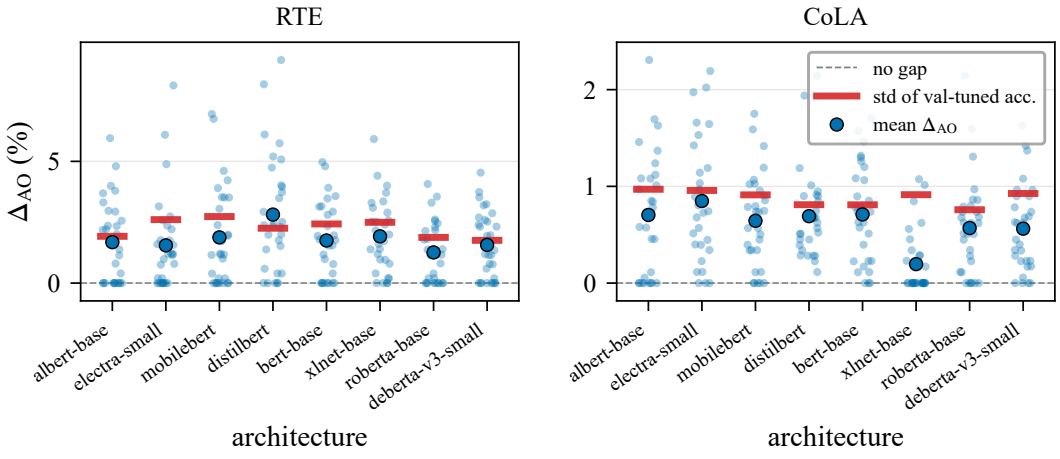


Figure 5: Overfitting severity on RTE and CoLA: per-run adaptive overfitting gap. Each translucent dot is one of the $K = 30$ paired runs of an architecture; the filled blue circle is the per-architecture mean Δ_{AO}. The red horizontal bar depicts one standard deviation of that architecture's val-tuned test accuracy, representing run-to-run noise. Only distilbert on RTE has a mean gap above its standard deviation. Compare Figure 2.

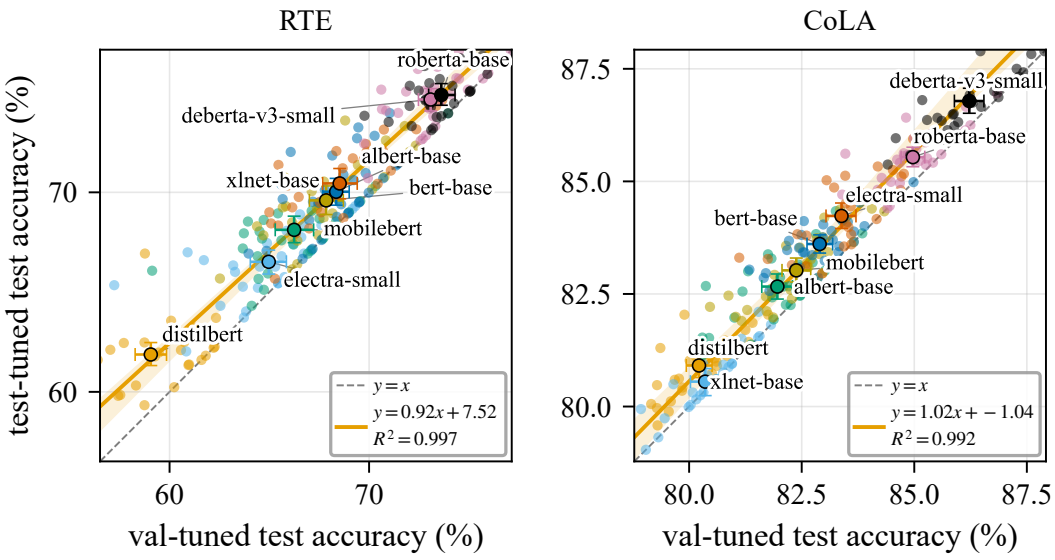


Figure 6: Validation-tuned vs. test-tuned mean test accuracy on RTE and CoLA. Markers are per-architecture means with 95 % bootstrap confidence intervals; dots are the individual paired runs. The dashed grey line is the identity $y = x$ and the orange line is the ordinary least squares fit through the eight architecture means, with its 95 % pairs-bootstrap confidence interval shaded. On both datasets the means lie close to a line nearly parallel to the identity. Compare Figure 3.

B Implementation Details

Batch size. The two image experiments use a training batch size of 128 and an evaluation batch size of 512. The GLUE experiment uses a training batch size of 32, following the published fine-tuning recipes of the model families we evaluate, and an evaluation batch size of 128.

Data augmentation (CIFAR-10). During training, each mini-batch is augmented with random cropping (32×32 with 4-pixel padding) and random horizontal flipping, followed by per-channel normalisation to zero mean and unit variance. At evaluation time, only the normalisation is applied.

Mixed-precision training (CIFAR-10). All CIFAR-10 models are trained with automatic mixed precision (`torch.amp`) using a gradient scaler to maintain numerical stability.

Training budget (CIFAR-10). The number of training steps is itself a tuned hyperparameter, sampled log-uniformly from $[75{,}000,\ 141{,}000]$. At batch size 128, the bootstrapped 60,000-example training split yields ≈ 470 mini-batches per epoch, so this range corresponds to roughly 160–300 epochs. This sits within the typical training-length range reported for most of the architectures we use, providing enough budget for each configuration to converge while keeping experiments computationally manageable. Since our goal is to measure adaptive overfitting rather than to reproduce state-of-the-art accuracies, what matters is that the hyperparameter choices yield meaningful accuracy differences between configurations.

Fine-tuning setup (GLUE). Each GLUE trial loads a fresh pretrained checkpoint with a randomly initialised classification head and fine-tunes all parameters; no layers are frozen. Inputs are tokenised with each model's own tokenizer, truncated and padded to a maximum sequence length of 128. Training uses AdamW with a linear warmup-then-decay schedule and gradient clipping at a maximum norm of 1.0. The number of epochs, the warmup ratio, the learning rate and the weight decay are all tuned (Table 7); the batch size is held fixed at 32. The initialisation seed is sampled per trial and applied *before* the model is constructed, so the classification head's initialisation and the data ordering are both part of the trial's seed — fine-tuning on small GLUE tasks is known to be highly sensitive to this choice Dodge et al. (2020).

Numerical precision (GLUE). All GLUE models are trained with automatic mixed precision and a gradient scaler, with one exception: MobileBERT's activations overflow in fp16, so it is trained in full fp32. Checkpoints are loaded with an explicit fp32 master dtype, because some checkpoints (e.g. DeBERTa-v3-small) declare a half-precision `torch_dtype` and would otherwise load in fp16, which is incompatible with gradient scaling. With fp32 master weights, autocast still casts activations to fp16 in the forward pass while gradients remain fp32.

C HYPERPARAMETER SEARCH SPACES

Table 5: Hyperparameter search space for MNIST-1D.

Hyperparameter	Distribution	Range
Learning rate	Log-Uniform	$[10^{-4}, 10^{-1}]$
Learning-rate decay rate	Log-Uniform†	$[10^{-5}, 10^{-2}]$
Weight decay	Log-Uniform	$[10^{-6}, 10^{-2}]$
Initialization seed	Uniform integer	$[0,\ 2^{31}-1]$

† Set to zero with probability 0.3 to explicitly sample the no-decay setting.

Table 6: Hyperparameter search space for CIFAR-10.

Hyperparameter	Distribution	Range
Learning rate	Log-Uniform	$[5\times10^{-3},\ 0.5]$
Weight decay	Log-Uniform	$[10^{-5},\ 10^{-2}]$
SGD momentum	Uniform	$[0.8,\ 0.99]$
Training steps	Log-Uniform	[75,000, 141,000]
Initialization seed	Uniform integer	$[0,\ 2^{31}-1]$

Table 7: Hyperparameter search space for the GLUE tasks, shared across all three tasks and all eight architectures. The ranges bracket the published fine-tuning grids of the model families in the registry: learning rates of 2–5 $\times\ 10^{-5}$ (BERT, XLNet), 1–3 $\times\ 10^{-5}$ (RoBERTa), 1–5 $\times\ 10^{-5}$ (ALBERT) and 10^{-4} (ELECTRA-base); weight decays of 0 (HuggingFace `run_glue` default), 0.01 (BERT, DeBERTa) and 0.1 (RoBERTa); warmup ratios of 6% (RoBERTa) and 10% (BERT); and epoch counts of 3 (BERT) through 10 (RoBERTa).

Hyperparameter	Distribution	Range
Learning rate	Log-Uniform	$[5\times10^{-6},\ 10^{-4}]$
Weight decay	Log-Uniform	$[10^{-5},\ 10^{-1}]$
Warmup ratio	Uniform	$[0.0,\ 0.2]$
Number of epochs	Uniform integer	$[2,\ 10]$
Initialization seed	Uniform integer	$[0,\ 2^{31}-1]$

D ARCHITECTURES

D.1 MNIST-1D

Table 8: MLP family evaluated on MNIST-1D. All networks use ReLU activations and residual connections between hidden layers.

Architecture	Hidden sizes	Parameters
MLP-Tiny	[25]	1,285
MLP-Mini	[50]	2,560
MLP-Small	[100]	5,110
MLP-Base	[100, 100]	15,210
MLP-Large	[100, 100, 100]	25,310
MLP-Larger	[200, 200, 200]	90,610
MLP-Giant	[200, 200, 200, 200]	130,810
MLP-Huge	[300, 300, 300]	195,910

D.2 CIFAR-10

Table 9: Architectures evaluated on CIFAR-10. Each model is ported from its reference implementation without architectural modifications.

#	Model	Paper	Repository
1	ResNet-20	He et al. (2016)	`github.com/hysts/pytorch_image_classification`
2	ResNet-32	He et al. (2016)	`github.com/hysts/pytorch_image_classification`
3	ResNet-44	He et al. (2016)	`github.com/hysts/pytorch_image_classification`
4	ResNet-56	He et al. (2016)	`github.com/hysts/pytorch_image_classification`
5	Shake-Shake (32d)	Gastaldi (2017)	`github.com/hysts/pytorch_image_classification`
6	VGG-15 BN	Simonyan & Zisserman (2015)	`github.com/hysts/pytorch_image_classification`
7	MobileNetV2	Sandler et al. (2018)	`github.com/hysts/pytorch_image_classification`
8	ResNet-9	Page (2019)	`github.com/KellerJordan/cifar10-airbench`

D.3 GLUE

Table 10: Pretrained transformers fine-tuned on each GLUE task. All checkpoints are public Hugging Face models, used without architectural modification; each trial loads the checkpoint afresh and attaches a randomly initialised classification head. The same eight models are evaluated on all three tasks.

#	Model	Paper	Parameters	Hugging Face checkpoint
1	ALBERT-base-v2	Lan et al. (2020)	11.8M	`albert/albert-base-v2`
2	ELECTRA-small	Clark et al. (2020)	14.0M	`google/electra-small-discriminator`
3	MobileBERT	Sun et al. (2020)	25.3M	`google/mobilebert-uncased`
4	DistilBERT-base	Sanh et al. (2019)	66.4M	`distilbert/distilbert-base-uncased`
5	BERT-base	Devlin et al. (2019)	109.5M	`google-bert/bert-base-uncased`
6	XLNet-base	Yang et al. (2019)	110.0M	`xlnet/xlnet-base-cased`
7	RoBERTa-base	Liu et al. (2019)	124.6M	`FacebookAI/roberta-base`
8	DeBERTa-v3-small	He et al. (2023)	141.9M	`microsoft/deberta-v3-small`

E RANKING PRESERVATION DETAILS

For each benchmark we fit test-tuned on validation-tuned test accuracy by ordinary least squares on the per-architecture means, and report Spearman and Kendall rank correlations on those same means. All confidence intervals are based on bootstrap resampling. For the linear fit we use a standard pairs bootstrap with 100,000 resamples, drawing the eight architecture mean pairs with

replacement and refitting the line at each iteration. The rank correlations use a cluster bootstrap with 20,000 resamples, drawing the K runs within each architecture with replacement and recomputing the architecture means at each iteration.

E.1 MNIST-1D

Linear fit:

- Slope $= 0.960$, $95\,\%$ bootstrap CI: $[0.893,\ 0.988]$
- Intercept $= 3.534$, $95\,\%$ bootstrap CI: $[1.885,\ 7.708]$
- $R^2 = 0.997$, $95\,\%$ bootstrap CI: $[0.992,\ 1.000]$
- p-value $= 5.73 \times 10^{-9}$

Rank correlations:

- Spearman $\rho = 1.000$ $(p \approx 0)$, $95\,\%$ bootstrap CI: $[0.929,\ 1.000]$
- Kendall $\tau = 1.000$ $(p = 4.96 \times 10^{-5})$, $95\,\%$ bootstrap CI: $[0.857,\ 1.000]$

E.2 CIFAR-10

Linear fit:

- Slope $= 0.973$, $95\,\%$ bootstrap CI: $[0.942,\ 1.001]$
- Intercept $= 2.631$, $95\,\%$ bootstrap CI: $[0.043,\ 5.544]$
- $R^2 = 0.999$, $95\,\%$ bootstrap CI: $[0.998,\ 1.000]$
- p-value $= 1.08 \times 10^{-10}$

Rank correlations:

- Spearman $\rho = 1.000$ $(p \approx 0)$, $95\,\%$ bootstrap CI: $[0.976,\ 1.000]$
- Kendall $\tau = 1.000$ $(p = 4.96 \times 10^{-5})$, $95\,\%$ bootstrap CI: $[0.929,\ 1.000]$

E.3 RTE

Linear fit:

- Slope $= 0.916$, $95\,\%$ bootstrap CI: $[0.885,\ 1.007]$
- Intercept $= 7.523$, $95\,\%$ bootstrap CI: $[1.202,\ 9.531]$
- $R^2 = 0.997$, $95\,\%$ bootstrap CI: $[0.992,\ 1.000]$
- p-value $= 1.06 \times 10^{-8}$

Rank correlations:

- Spearman $\rho = 1.000$ $(p \approx 0)$, $95\,\%$ bootstrap CI: $[0.905,\ 1.000]$
- Kendall $\tau = 1.000$ $(p = 4.96 \times 10^{-5})$, $95\,\%$ bootstrap CI: $[0.786,\ 1.000]$

E.4 MRPC

Linear fit:

- Slope $= 0.910$, $95\,\%$ bootstrap CI: $[0.850,\ 0.960]$
- Intercept $= 8.787$, $95\,\%$ bootstrap CI: $[4.610,\ 13.889]$
- $R^2 = 0.994$, $95\,\%$ bootstrap CI: $[0.982,\ 0.999]$
- p-value $= 6.58 \times 10^{-8}$

Rank correlations:

- Spearman $\rho = 0.976$ $(p = 3.31 \times 10^{-5})$, 95 % bootstrap CI: $[0.905,\ 1.000]$
- Kendall $\tau = 0.929$ $(p = 3.97 \times 10^{-4})$, 95 % bootstrap CI: $[0.786,\ 1.000]$

The single rank change is an exchange between `mobilebert` and `electra-small`, whose validation-tuned means differ by 0.30% against a run-to-run standard deviation of 1.34%.

E.5 CoLA

Linear fit:

- Slope $= 1.020$, 95 % bootstrap CI: $[0.956,\ 1.157]$
- Intercept $= -1.040$, 95 % bootstrap CI: $[-12.260,\ 4.374]$
- $R^2 = 0.992$, 95 % bootstrap CI: $[0.985,\ 1.000]$
- p-value $= 1.47 \times 10^{-7}$

Rank correlations:

- Spearman $\rho = 0.976$ $(p = 3.31 \times 10^{-5})$, 95 % bootstrap CI: $[0.952,\ 1.000]$
- Kendall $\tau = 0.929$ $(p = 3.97 \times 10^{-4})$, 95 % bootstrap CI: $[0.857,\ 1.000]$

The single rank change is an exchange between `xlnet-base` and `distilbert`, whose validation-tuned means differ by 0.14% against a run-to-run standard deviation of 0.88%.

F Upper-Bound Reference

Benchmark	Upper bound (%)	Headroom (%)	Captured (%)
MNIST-1D	100.00	35.4–46.0	2.0–3.7
CIFAR-10	100.00	4.6–7.7	1.2–2.4
RTE	99.80–100.00	26.4–40.9	4.4–6.9
MRPC	99.06–100.00	13.6–18.7	6.9–9.6
CoLA	98.87–100.00	13.8–19.7	1.0–5.5

Table 11: Ranges over the eight architectures of each benchmark. *Upper bound* is the best test accuracy among $T = 30$ models trained and tuned directly on the bootstrap test fold. *Headroom* is the distance from the validation-tuned mean to that bound, and *captured* is the share of it that test-tuning actually realises, $(A(\lambda^*_{\text{test}}) - A(\lambda^*_{\text{val}}))$/headroom.

Every architecture very nearly memorises its own test fold: exactly 100% on both image benchmarks, and between 98.9% and 100% on the GLUE tasks, whose bootstrap test folds are smaller ($\sim$509 examples on RTE to $\sim$1,763 on CoLA). Even MLP-Tiny has enough capacity to memorise the $\sim$736-sample MNIST-1D fold. Headroom above the validation-tuned mean varies widely across benchmarks, from 4.6–7.7% on CIFAR-10, where validation-tuned accuracy is already in the low-90%s, up to 35–46% on MNIST-1D. Yet in every case random search at $T = 30$ captures at most 9.6% of it, so the observed gaps lie far below what any procedure exploiting the test set could achieve under this budget.

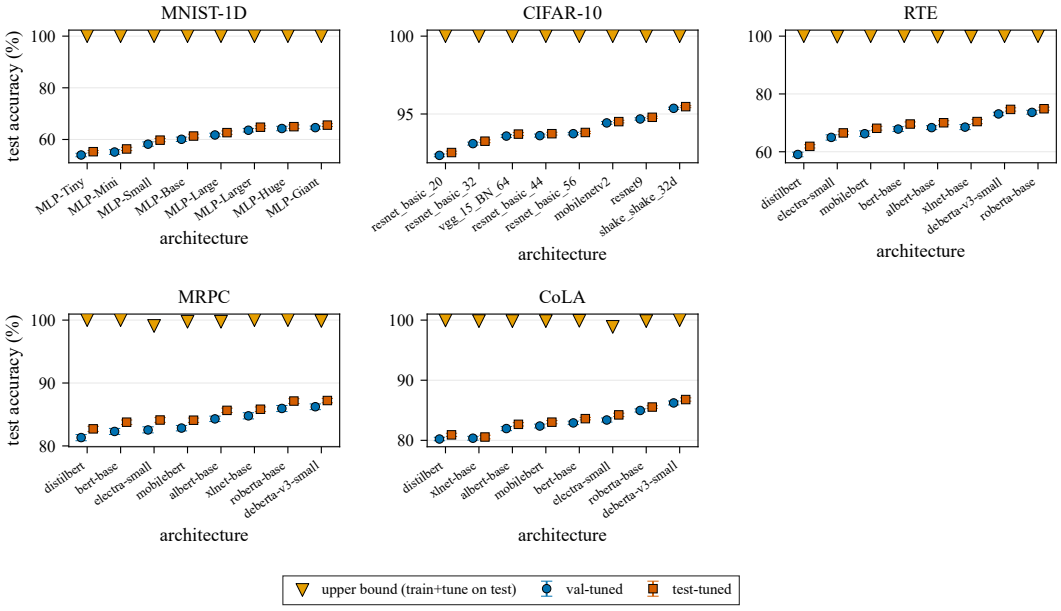


Figure 7: Upper-bound reference per architecture across the five benchmarks, ordered by mean validation-tuned test accuracy. Blue circles are the validation-tuned mean test accuracy and orange squares the test-tuned mean test accuracy (both with 95 % bootstrap CIs over the $K = 30$ runs). Orange triangles are the upper bound: the best test accuracy among $T = 30$ models trained directly on the same bootstrap test fold. The vertical distance between the blue marker and the orange triangle is the headroom available to any procedure that exploits the test set; the much smaller distance between the blue circle and orange square is the share that random search over $T = 30$ trials actually captures.

G MLP-GIANT$_{T=100}$: ADAPTIVE OVERFITTING UNDER INCREASED TUNING AND DATA BUDGET

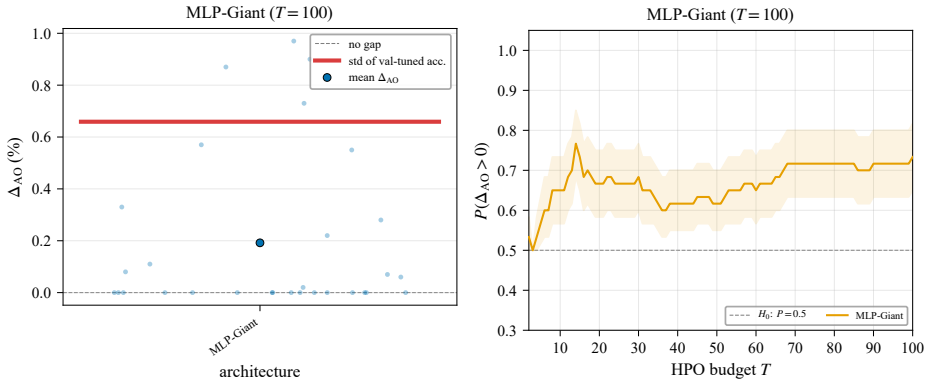


Figure 8: MLP-Giant rerun with $T = 100$ trials per run on larger 10,000/10,000/10,000 MNIST-1D splits. **Left:** per-run Δ_{AO} across the $K = 30$ paired runs; the blue dot is the mean and the red horizontal bar marks the standard deviation of val-tuned accuracy. **Right:** $\hat{P}(\Delta_{AO} > 0)$ as a function of hyperparameter tuning budget T, with the 95 % percentile-bootstrap CI; the dashed grey reference at 0.5 is the significance threshold. The mean gap stays small relative to the val-tuned noise and $\hat{P}(\Delta_{AO} > 0)$ remains significant ($CI_{lower} > 0.5$) across the full budget range.

H ADAPTIVE OVERFITTING TRANSFER TO CIFAR-10.1

Model	Val-tuned (%) on CIFAR-10.1	Test-tuned (%) on CIFAR-10.1	CIFAR-10.1-tuned (%) on CIFAR-10.1
shake_shake_32d	88.79	88.82	89.11
resnet9	86.95	86.93	87.41
mobilenetv2	87.11	87.06	87.29
resnet_basic_56	86.07	86.11	86.39
resnet_basic_44	85.95	85.78	86.26
vgg_15_BN_64	85.40	85.34	85.86
resnet_basic_32	85.23	85.11	85.69
resnet_basic_20	83.81	83.81	84.36
Pooled	**86.16**	**86.12**	**86.55**

Table 12: Per-architecture mean CIFAR-10.1 accuracy under three selection rules. **Val-tuned** and **Test-tuned** pick the trial with the highest accuracy on the bootstrap validation or test fold, respectively, and the picked configuration is then re-evaluated on CIFAR-10.1; **CIFAR-10.1-tuned** picks the trial with the highest CIFAR-10.1 accuracy directly. "Pooled" is the mean across the eight architectures. The val-tuned and test-tuned columns are indistinguishable on CIFAR-10.1 (pooled gap -0.04%, within run-to-run noise), consistent with the small $\Delta_{AO} \approx 0.12\%$ already observed on the bootstrap test fold. CIFAR-10.1-tuned wins by $+0.43\%$ pooled — a small but consistent margin that the bootstrap-test-tuned configuration does not capture.

I DATA SPLITTING

We use the same splitting scheme on all five benchmarks. We start from a single source pool: 4,000 examples for MNIST-1D, the 60,000-example pool that combines CIFAR-10's training and test sets, and for the GLUE tasks the concatenation of the official train and validation splits (2,767 for RTE, 4,076 for MRPC and 9,594 for CoLA). For each paired run we draw the train set by sampling from the pool with replacement, taking as many samples as the pool size. Examples that were never picked end up in the leftover set, which on average is about 36.8% of the pool. We shuffle this leftover set and split it in half: one half is the validation set and the other is the test set.

For MNIST-1D this gives roughly 4,000 training, 736 validation and 736 test examples per run, and for CIFAR-10 roughly 60,000 training, 11,040 validation and 11,040 test examples. For the GLUE tasks it gives roughly 509 validation and 509 test examples on RTE, 750 and 750 on MRPC, and 1,763 and 1,763 on CoLA. The validation and test sets never overlap, and they never share an example with the training set. The split is drawn once at the start of a run and stays the same across all T trials, so validation-tuning and test-tuning pick from the same pool of trained models. A new split is drawn for each of the $K = 30$ paired runs.

We use the pooled train and validation splits for the GLUE tasks because the official GLUE test splits are unlabeled and withheld on the leaderboard server, so the validation split is the de facto test set in community practice.

The MLP-Giant rerun in Appendix G uses a different scheme. It draws three disjoint sets of 10,000 examples each (train, validation, test) from a larger 70,000-example MNIST-1D set, without replacement.

J Hyperparameter Range Validity

The empirical winner distributions in Figures 9 and 10 suggest the search ranges are well-shaped: `lr`, `weight_decay`, and `momentum` have their modes inside the bounds with no edge clustering. The one exception is CIFAR-10's `max_steps`, whose winners pile up at the upper bound of 141,000; extending the range would benefit both selection rules equally and is unlikely to qualitatively change Δ_{AO}. For the GLUE tasks the ranges were instead set a priori to bracket the published fine-tuning grids of all eight model families (Table 7), with one deliberate exception: ELECTRA-small's published 3×10^{-4} recipe lies above our 10^{-4} learning-rate cap, which is kept below the divergence region of the base-size models. The winner distributions in Figures 11–13 confirm those ranges empirically: `lr` peaks around 3–5 $\times$ 10^{-5}, where the published recipes sit, while `weight_decay` and `warmup_ratio` are close to flat, indicating that neither strongly determines which configuration wins. As with CIFAR-10's `max_steps`, only `num_epochs` clusters at its upper bound of 10, and extending it would again benefit both selection rules equally.

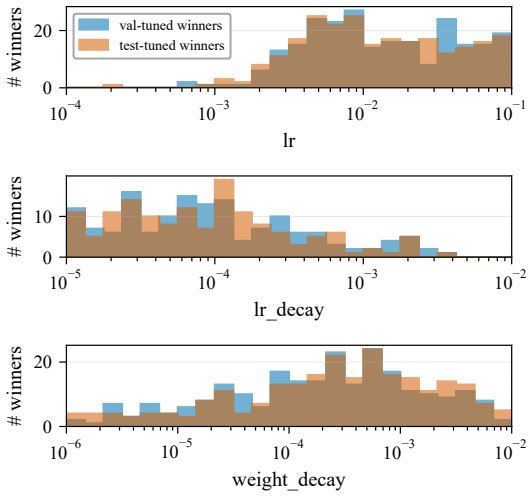


Figure 9: MNIST-1D: empirical distribution of winning hyperparameter values for the validation-tuned and test-tuned selection rules. 30.8% of trials had `lr_decay` $= 0$ and are excluded from the `lr_decay` panel. All three HPs have their modes well inside the bounds with no edge clustering.

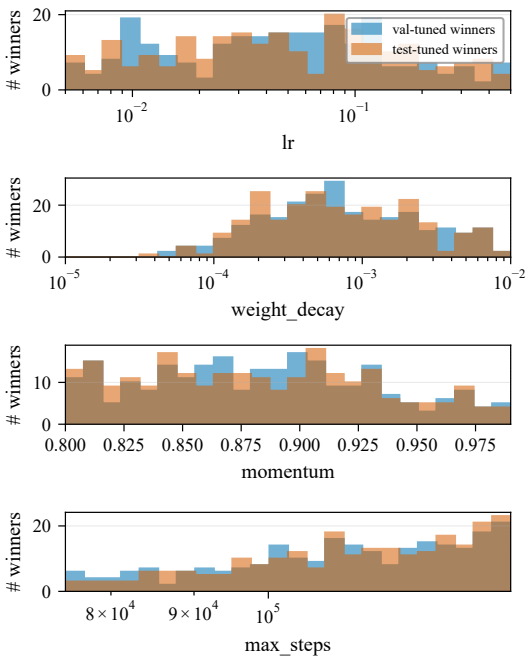


Figure 10: CIFAR-10: empirical distribution of winning hyperparameter values for the val-tuned and test-tuned selection rules. `lr`, `weight_decay`, and `momentum` have their modes well inside the bounds; only `max_steps` clusters at the upper bound of 141,000, suggesting the optimum lies above this cap. Extending it would raise absolute accuracies but is unlikely to change Δ_{AO}, since both selection rules benefit equally from longer training.

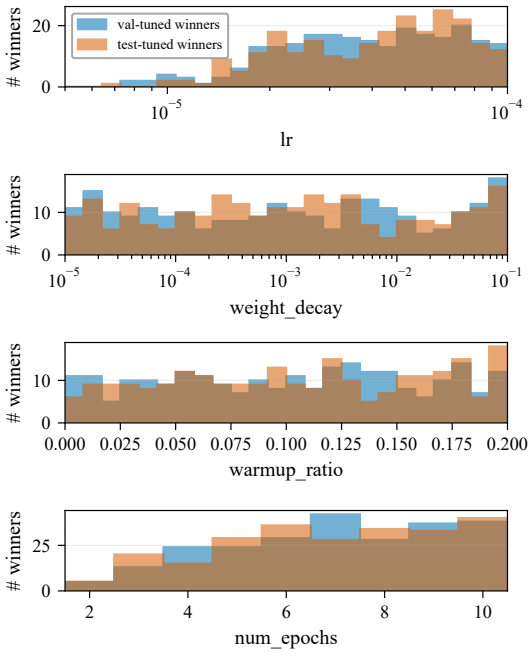


Figure 11: RTE: empirical distribution of winning hyperparameter values for the val-tuned and test-tuned selection rules, over the 240 winners of each rule (8 architectures $\times$ $K = 30$ runs). `num_epochs` is binned on the integers 2–10.

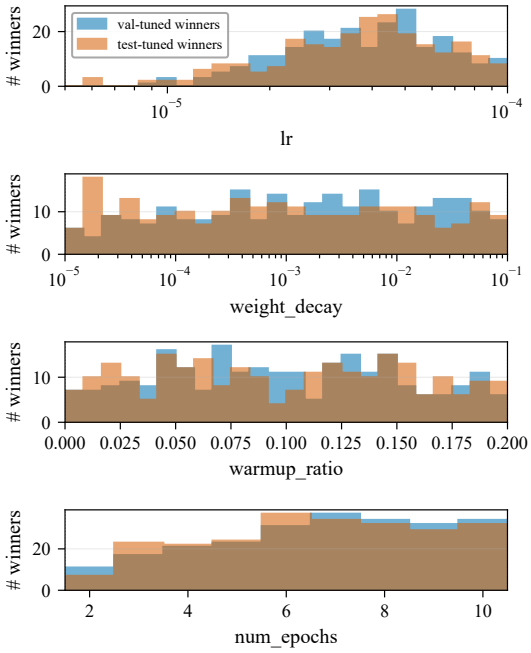


Figure 12: MRPC: empirical distribution of winning hyperparameter values for the val-tuned and test-tuned selection rules, over the 240 winners of each rule.

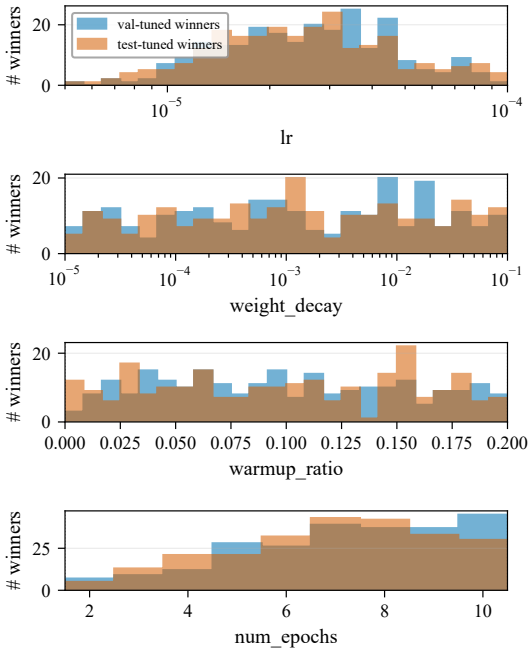


Figure 13: CoLA: empirical distribution of winning hyperparameter values for the val-tuned and test-tuned selection rules, over the 240 winners of each rule. As on the other two GLUE tasks, `num_epochs` rises towards the upper bound of the range.