Are Larger Pretrained Language Models Uniformly Better? Comparing Performance at the Instance Level

Ruiqi Zhong Dhruba Ghosh Dan Klein Jacob Steinhardt
Computer Science Division, University of California, Berkeley
{ruiqi-zhong, djghosh13, klein, jsteinhardt}@berkeley.edu

Abstract

Larger language models have higher accuracy on average, but are they better on every single instance (datapoint)? Some work suggests larger models have higher out-of-distribution robustness, while other work suggests they have lower accuracy on rare subgroups. To understand these differences, we investigate these models at the level of individual instances. However, one major challenge is that individual predictions are highly sensitive to noise in the randomness in training. We develop statistically rigorous methods to address this, and after accounting for pretraining and finetuning noise, we find that our BERT-LARGE is worse than BERT-MINI on at least 1−4% of instances across MNLI, SST-2, and QQP, compared to the overall accuracy improvement of 2−10%. We also find that finetuning noise increases with model size, and that instance-level accuracy has momentum: improvement from BERT-MINI to BERT-MEDIUM correlates with improvement from BERT-MEDIUM to BERT-LARGE . Our findings suggest that instance-level predictions provide a rich source of information; we therefore recommend that researchers supplement model weights with model predictions.

1 Introduction

Historically, large deep learning models (Peters et al., 2018; Devlin et al., 2019; Lewis et al., 2020; Raffel et al., 2019) have improved the state of the art on a wide range of tasks and leaderboards (Schwartz et al., 2014; Rajpurkar et al., 2016; Wang et al., 2018), and empirical scaling laws predict that larger models will continue to increase performance (Kaplan et al., 2020). However, little is understood about such improvement at the instance (datapoint) level. Are larger models uniformly better? In other words, are larger pretrained models better at every instance, or are they better at some instances, but worse at others?

Prior works hint at differing answers. Hendrycks et al. (2020) and Desai and Durrett (2020) find that larger pretrained models consistently improve out-of-distribution performance, which implies that they might be uniformly better at a finer level. Henighan et al. (2020) claim that larger pretrained image models have lower downstream classification loss for the majority of instances, and they predict this trend to be true for other data modalities (e.g. text). On the other hand, Sagawa et al. (2020) find that larger non-pretrained models perform worse on rare subgroups; if this result generalizes to pretrained language models, larger models will not be uniformly better. Despite all the indirect evidence, it is still inconclusive how many instances larger pretrained models perform worse on.

A naïve solution is to finetune a larger model, compare it to a smaller one, and find instances where the larger model is worse. However, this approach is flawed, since model predictions are **noisy at the instance level**. On MNLI in-domain development set, even the same architecture with different finetuning seeds leads to different predictions on ∼8% of the instances. This is due to under-specification (D'Amour et al., 2020), where there are multiple different solutions that can minimize the training loss. Since the accuracy improvement from our BERT-BASE[1] to BERT-LARGE is 2%, most signals across different model sizes will be dominated by noise due to random seeds.

To account for the noise in pretraining and finetuning, we define *instance accuracy* as "how often a model correctly predicts an instance" (Figure 1 left) in expectation across pretraining and finetuning seeds. We estimate this quantity by pretraining 10 models with different seeds, finetuning 5 times for each pretrained models (Figure 1 middle), and

[1]This is not the original release by Devlin et al. (2019); we pretrained models ourselves.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3813–3827

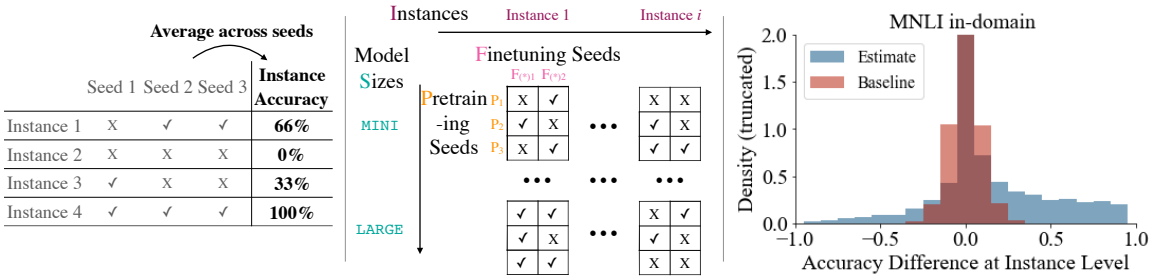


Figure 1: **Left**: Each column represents the same architecture trained with a different seed. We calculate accuracy for each instance (row) by averaging across seeds (column), while it is usually calculated for each model by averaging across instances. **Middle**: A visual layout of the model predictions we obtain, which is a binary-valued tensor with 4 axes: model size s, instance i, pretraining seeds P and finetuning seeds F. **Right**: for each instance, we calculate the accuracy gain from MINI to LARGE and plot the histogram in blue, along with a random baseline in red. Since the blue distribution has a bigger left tail, smaller models are better at some instances.

averaging across them.

However, this estimate is still inexact, and we might falsely observe smaller models to be better at some instances by chance. Hence, we propose a random baseline to estimate the fraction of *false discoveries* (Section 3, Figure 1 right) and formally upper-bound the false discoveries in Section 4. Our method provides a better upper bound than the classical Benjamini-Hochberg procedure with Fisher's exact test.

Using the 50 models for each size and our improved statistical tool, we find that, on the MNLI in-domain development set, the accuracy "decays" from BERT-LARGE to BERT-MINI on at least ∼4% of the instances, which is significant given that the improvement in overall accuracy is 10%. These decaying instances contain more controversial or wrong labels, but also correct ones (Section 4.2). Therefore, larger pretrained language models are not uniformly better.

We make other interesting discoveries at the instance level. Section 5 finds that instance-level accuracy has momentum: improvement from MINI to MEDIUM correlates with improvement from MEDIUM to LARGE . Additionally, Section 6 attributes variance of model predictions to pretraining and finetuning random seeds, and finds that finetuning seeds cause more variance for larger models. Our findings suggest that instance-level predictions provide a rich source of information; we therefore recommend that researchers supplement model weights with model predictions. In this spirit, we release all the pretrained models, model predictions, and code here: https://github.com/ruiqi-zhong/acl2021-instance-level.

2 Data, Models, and Predictions

To investigate model behavior, we considered different sizes of the BERT architecture and finetuned them on Quora Question Pairs (QQP[2]), Multi-Genre Natural Language Inference (MNLI; Williams et al. (2020)), and the Stanford Sentiment Treebank (SST-2; Socher et al. (2013)). To account for pretraining and finetuning noise, we averaged over multiple random initializations and training data order, and thus needed to pretrain our own models rather than downloading off the internet. Following Turc et al. (2019) we trained 5 architectures of increasing size: MINI (L4/H256, 4 Layers with hidden dimension 256), SMALL (L4/H512), MEDIUM (L8/H512), BASE (L12/H768), and LARGE (L24/H1024). For each architecture we pre-trained models with 10 different random seeds and fine-tuned each of them 5 times (50 total) on each task; see Figure 1 middle. Since pretraining is computationally expensive, we reduced the context size during pretraining from 512 to 128 and compensated by increasing training steps from 1M to 2M. Appendix A includes more details about pretraining and finetuning and their computational cost, and Appendix B verifies that our cost-saving changes do not affect accuracy qualitatively.

Notation. We use i to index an instance in the evaluation set, s for model sizes, P for pretraining seeds and F for finetuning seeds. c is a random variable of value 0 or 1 to indicate whether the prediction is correct. Given the pretraining seed P and the finetuning seed F, $c_i^s = 1$ if the model of

[2]https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

	$\text{Diff}_{\text{FTune}}$	$\text{Diff}_{\text{PTrain}}$	Std_{all}
MINI	7.2%	10.7%	0.2%
SMALL	7.2%	10.7%	0.3%
MEDIUM	8.0%	10.7%	0.3%
BASE	8.5%	10.6%	0.2%
LARGE	8.6%	10.1%	0.2%

Table 1: Larger model sizes are at the bottom rows. $\text{Diff}_{\text{FTune}}$: how much do the predictions differ, if two models have the same pretraining seed but different finetuning seeds F? $\text{Diff}_{\text{PTrain}}$: the difference if the pretraining seeds P are different. Std_{all}: the standard deviation of overall accuracy, around 40 times smaller than $\text{Diff}_{\text{FTune}}$.

size s is correct on instance i, 0 otherwise. To keep the notation uncluttered, we sometimes omit these superscripts or subscripts if they can be inferred from context.

Unless otherwise noted, we present results on the MNLI in-domain development set in the main paper.

3 Comparing Instance Accuracy

To find the instances where larger models are worse, a naïve approach is to finetune a larger pretrained model, compare it to a smaller one, and find instances where the larger is incorrect but the smaller is correct. Under this approach, BERT-LARGE is worse than BERT-BASE on 4.5% of the instances and better on 7%, giving an overall accuracy improvement of 2.5%.

However, this result is misleading: even if we compare two BERT-BASE model with different finetuning seeds, their predictions differ on 8% of the instances, while their accuracies differ only by 0.1%; Table 1 reports this baseline randomness across model sizes. Changing the pretraining seed also changes around 2% additional predictions beyond finetuning.

Table 1 also reports the standard deviation of overall accuracy, which is about 40 times smaller. Such stability starkly contrasts with the noisiness at the instance level, which poses a unique challenge.

Instance-Level Metrics To reflect this noisiness, we define the *instance accuracy* Acc_i^s to be how often models of size s predict instance i correctly,

$$\text{Acc}_i^s := \mathbb{E}_{P,F}[c_i^s]. \tag{1}$$

The expectation is taken with respect to the pretraining and finetuning randomness P and F. We estimate Acc_i^s via the empirical average $\hat{\text{Acc}}_i^s$ across 10 pretraining $\times$ 5 finetuning runs.

We histogram $\hat{\text{Acc}}_i^s$ in Figure 2 (a). On most instances the model always predicts correctly or incorrectly ($\hat{\text{Acc}} = 0$ or 1), but a sizable fraction of accuracies lie between the two extremes.

Recall that our goal is to find instances where larger models are less accurate, which we refer to as *decaying* instances. We therefore study the *instance difference* between two model sizes s_1 and s_2, defined as

$$\Delta\text{Acc}_i := \text{Acc}_i^{s_2} - \text{Acc}_i^{s_1}, \tag{2}$$

which is estimated by the difference between the accuracy estimates $\hat{\text{Acc}}_i^s$, i.e.

$$\Delta\hat{\text{Acc}}_i := \hat{\text{Acc}}_i^{s_2} - \hat{\text{Acc}}_i^{s_1}. \tag{3}$$

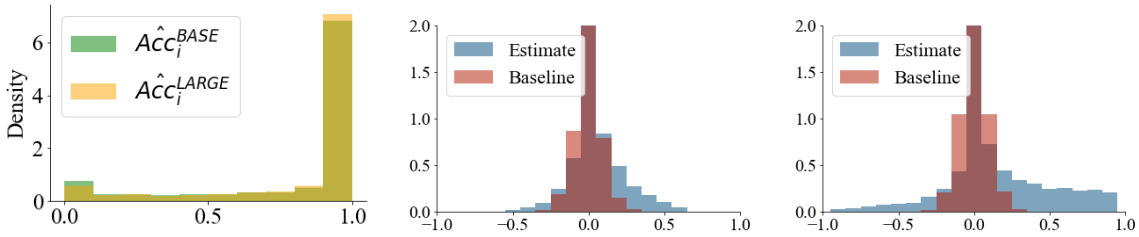
We histogram $^{\text{BASE}}_{\text{LARGE}}\Delta\hat{\text{Acc}}_i$ in Figure 2 (b). We observe a unimodal distribution centered near 0, with tails on both sides. Therefore, the estimated differences for some instances are negative.

However, due to estimation noise, we might falsely observe this accuracy decay by chance. Therefore, we introduce a random baseline $^{s_1}_{s_2}\Delta\text{Acc}'$ to control for these false discoveries. Recall that we have 10 smaller pretrained models and 10 larger ones. Our baseline splits these into a group A of 5 smaller + 5 larger, and another group B of the remaining $5+5$. Then the empirical accuracies $\hat{\text{Acc}}^A$ and $\hat{\text{Acc}}^B$ are identically distributed, so we take our baseline $^{s_1}_{s_2}\Delta\text{Acc}'$ to be the difference $\hat{\text{Acc}}^A - \hat{\text{Acc}}^B$. We visualize and compare how to calculate $^{s_1}_{s_2}\Delta\hat{\text{Acc}}$ and $^{s_1}_{s_2}\Delta\text{Acc}'$ in Figure 3.

We histogram this baseline $^{\text{BASE}}_{\text{LARGE}}\Delta\text{Acc}'$ in Figure 2 (b), and find that our noisy estimate $^{\text{BASE}}_{\text{LARGE}}\Delta\hat{\text{Acc}}$ has a larger left tail than the baseline. This suggests that decaying instances exist. We similarly compare MINI to LARGE in Figure 2 (c) and find an even larger left tail.

4 Quantifying the Decaying Instances

The left tail of $\Delta\hat{\text{Acc}}$ noisily estimates the fraction of decaying instances, and the left tail of the random baseline $\Delta\text{Acc}'$ counts the false discovery fraction due to the noise. Intuitively, the true fraction of decaying instances can be captured by the difference of these left tails, and we formally quantify this below.



(a) BASE vs. LARGE , Acc (b) BASE vs. LARGE , ΔAcc (c) MINI vs. LARGE , ΔAcc

Figure 2: (a) The distribution of instance accuracy $\hat{\text{Acc}}_i$. (b, c) Histogram of instance difference estimate (x-axis), $\Delta\hat{\text{Acc}}$ (blue) and its baseline $\Delta\text{Acc}'$ (red) compares BASE and LARGE . To better visualize, we truncated the density (y-axis) above 2. Since the blue histogram has a larger left tail than the red one, there are indeed instances where larger models are worse.

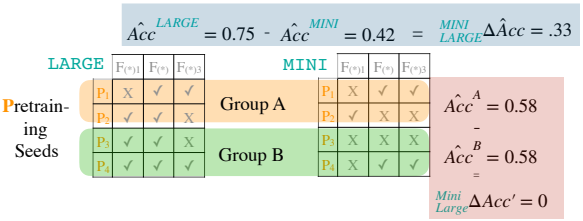


Figure 3: The tables are model predictions with visual notations established in Figure 1 middle. $\Delta\hat{\text{Acc}}$ (blue) is the mean difference between the left and the right table, each corresponding to a model size. The random baseline $\Delta\text{Acc}'$ (red) is the mean difference between group A (orange) cells and group B (green), which are identically and independently distributed.

Suppose instance i is drawn from the empirical evaluation distribution. Then we can define the true decaying fraction Decay as

$$\text{Decay} := \mathbb{P}_i[\Delta\text{Acc}_i < 0]. \tag{4}$$

Since ΔAcc_i is not directly observable and $\Delta\hat{\text{Acc}}_i$ is noisy, we add a buffer and only consider instances with $\Delta\hat{\text{Acc}}_i \le t$, which makes it more likely (but still uncertain) that the true $\Delta\text{Acc}_i < 0$. We denote this "discovery fraction" $\hat{\text{Decay}}(t)$ as

$$\hat{\text{Decay}}(t) := \mathbb{P}_i[\Delta\hat{\text{Acc}}_i \le t]. \tag{5}$$

Similarly, we define a baseline control (false discovery fraction) $\text{Decay}'(t) := \mathbb{P}_i[\Delta\text{Acc}'_i \le t]$. Hence, $\hat{\text{Decay}}$ and Decay' are the cumulative distribution function of $\Delta\hat{\text{Acc}}$ and $\Delta\text{Acc}'$ (Figure 4).

We have the following theorem, which we formally state and prove in Appendix D:

Theorem 1 *(Informal) If all the random seeds are independent, then for all thresholds t,*

$$\text{Decay} \ge \mathbb{E}[\hat{\text{Decay}}(t) - \text{Decay}'(t)] \tag{6}$$

Proof Sketch Suppose we observe $c^{s_1}_{R_{1\ldots 2k}}$ and $c^{s_2}_{R_{2k+1\ldots 4k}}$, where there are $2k$ different random seeds for each model size [3]. Then

$$\Delta\hat{\text{Acc}}_i := \frac{1}{2k}\Big(\sum_{j=1}^{2k} c^{s_1}_{R_j,i} - \sum_{j=2k+1}^{4k} c^{s_2}_{R_j,i}\Big), \tag{7}$$

and hence the discovery rate $\hat{\text{Decay}}(t)$ is defined as

$$\hat{\text{Decay}}(t) := \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|}\mathbf{1}[\Delta\hat{\text{Acc}} \le t]. \tag{8}$$

For the random baseline estimator, we have

$$\Delta\text{Acc}'_i := \frac{1}{2k}\Big(\sum_{j=1}^{k} c^{s_1}_{R_j,i} + \sum_{j=2k+1}^{3k} c^{s_2}_{R_j,i} - \sum_{j=k+1}^{2k} c^{s_1}_{R_j,i} - \sum_{j=3k+1}^{4k} c^{s_2}_{R_j,i}\Big), \tag{9}$$

and the false discovery control Decay' is defined as

$$\text{Decay}'(t) := \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|}\mathbf{1}[\Delta\text{Acc}'_i \le t]. \tag{10}$$

Formally, the theorem states that

$$\text{Decay} \ge \mathbb{E}_{R_1\ldots R_{4k}}[\hat{\text{Decay}}(t) - \text{Decay}'(t)], \tag{11}$$

which is equivalent to

$$\sum_{i=1}^{|\mathcal{T}|}\big(\mathbf{1}[\Delta\text{Acc}_i < 0] - \mathbb{P}[\Delta\hat{\text{Acc}}_i \le t] + \mathbb{P}[\Delta\text{Acc}'_i \le t]\big) \ge 0 \tag{12}$$

[3] We assumed even number of random seeds since we will mix half of the models from each size to compute the random baseline

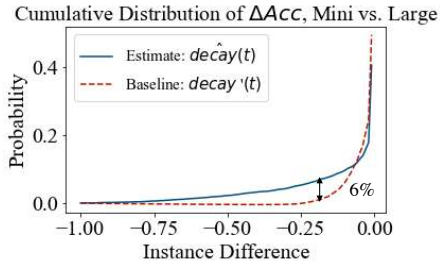


Figure 4: The cumulative distribution function of the histogram in Figure 2 (c); only the negative x-axis is shown because it corresponds to decays. The maximum difference between the two curves (6%) is a lower bound of the true decaying fraction.

Hence, we can declare victory if we can prove that for all i, if $\Delta\text{Acc}_i \geq 0$,

$$\mathbb{P}[\Delta\text{Acc}'_i \leq t] \geq \mathbb{P}[\Delta\hat{\text{Acc}}_i \leq t].$$

This is easy to see, since $\Delta\text{Acc}'_i$ and $\Delta\hat{\text{Acc}}_i$ are both binomial distributions with the same n, but the first has a larger rate. [4] □

Roughly speaking, the true decaying fraction is at least the difference between $\hat{\text{Decay}}(t)$ and $\text{Decay}'(t)$ at every threshold t. Therefore, we take the maximum difference between $\hat{\text{Decay}}(t)$ and $\text{Decay}'(t)$ to lower-bound the fraction of decaying instances.[5] For example, Figure 4 estimates the true decaying fraction between MINI and LARGE to be at least 6%.

We compute this lower bound for other pairs of model sizes in Table 2, and the full results across other tasks and model size pairs are in Appendix C. In all of these settings we find a non-zero fraction of decaying instances, and larger model size differences usually lead to more decaying instances.

Unfortunately, applying Theorem 1 as above is not fully rigorous, since some finetuning runs share the same pretraining seeds and hence are dependent.[6] To obtain a statistically rigorous lower bound, we slightly modify our target of interest. Instead of examining individual finetuning runs, we ensemble our model across 5 different finetuning runs for each pretraining seed; these predictions are essentially the same as individual finetuning runs, except that the finetuning randomness is averaged out. Hence we obtain 10 independent sets

[4] More details are in Appendix D.

[5] Adaptively picking the best threshold t depending on the data may incur a slight upward bias. Appendix E estimates that the relative bias is at most 10% using a bootstrap method.

[6] Although we anticipate such dependencies do not cause a substantial difference, as discussed in Appendix D.1.

$s_1 \setminus s_2$	MINI	SMALL	BASE	LARGE
MINI	N/A	9%	18%	21%
SMALL	3%	N/A	14%	18%
BASE	6%	5%	N/A	10%
LARGE	6%	5%	2%	N/A

Table 2: We lower-bound the fraction of instances that improve when model size changes from s_1 (row) to s_2 (column). For example, when model size decreases from LARGE to MINI , 6% of instances improve (i.e. decays).

Threshold	$\hat{\text{Decay}}$	Decay'	Diff
$t = -0.4$	4.22%	3.49e−3	3.87%
...	...	...	...
$t = -0.9$	0.91%	1.44e−7	0.91%
$t = -1.0$	0.48%	2.06e−8	0.48%

Table 3: Comparing MINI vs. LARGE by calculating the discovery fraction $\hat{\text{Decay}}$, the false discovery control Decay', and their difference (Diff) under different thresholds t. LARGE is worse on at least ∼4% (maximum Diff) of instances.

of model predictions with different random seeds, which allows us to apply Theorem 1.

We compare MINI to LARGE using these ensembles and report the discovery $\hat{\text{Decay}}$ and the baseline Decay' in Table 3. Taking the maximum difference across thresholds, we estimate at least ∼4% of decaying instances. This estimate is lower than the previous 6% estimate, which used the full set of 50 models' predictions assuming they were independent. However, this is still a meaningful amount, given that the overall accuracy improvement from MINI to LARGE is 10%.

4.1 Fisher's Test + Benjamini-Hochberg

Here is a more classical approach to lower-bound the decaying fraction. For each instance, we compute a significance level α under the null hypothesis that the larger model is better, using Fisher's exact test. We sort the significance levels ascendingly, and call the p^{th} percentile α_p. Then we pick a false discovery rate q (say, 25%), find the largest p s.t. $\alpha_p < pq$, and estimate the decaying fraction to be at least $p(1-q)$. This calculation is known as the Benjamini-Hochberg procedure (Benjamini and Hochberg, 1995).

To compare our method with this classical approach, we estimate the lower bound of the decaying fraction for different pairs of model sizes with different numbers of pretrained models available.

s1	s2		2	6	10
MINI	LARGE	ours	1.9%	3.1%	4.0%
MINI	LARGE	BH	0.0%	0.9%	1.9%
BASE	LARGE	ours	0.4%	0.9%	1.2%
BASE	LARGE	BH	0.0%	0.0%	0.0%

Table 4: We compare our method to the Fisher's exact test + Benjamin-Hochberg (BH) procedure described in Section 4. For all different model size pairs and number of pretrained models available, ours always provides a higher (better) lower bound of the decaying fraction.

To make sure our choice of the false discovery rate q does not bias against the classical approach, we adaptively choose q to maximize its performance. Appendix F includes the full results and Table 4 is a representative subset.

We find that our approach is more powerful, particularly when the true decaying fraction is likely to be small and only a few models are available, which is usually the regime of interest. For example, across all pairs of model sizes, our approach only needs 2 random seeds (i.e. pretrained models) to provide a non-zero lower bound on the decaying fraction, while the classical approach sometimes fails to do this even with 10 seeds. Intuitively, when fewer seeds are available, the smallest possible significance level for each instance is larger than the decaying fraction, hence hurting the classical approach.

4.2 Understanding the Decaying Instances

We next manually examine the decaying instances to see whether we can find any interpretable patterns. One hypothesis is that all the decaying fractions are in fact mislabeled, and hence larger models are not in fact worse on any instances.

To investigate this hypothesis, we examined the group of instances where $\overset{\text{MINI}}{\text{LARGE}}\Delta\hat{\text{Acc}}_i \leq -0.9$. MINI is almost always correct on these instances, while LARGE is almost always wrong, and the false discovery fraction is tiny. For each instance, we manually categorize it as either: 1) Correct, if the label is correct, 2) Fine, if the label might be controversial but we could see a reason why this label is reasonable, 3) Wrong, if the label is wrong, or 4) Unsure, if we are unsure about how to label this instance. Each time we annotate, with 50% probability we randomly sample either a decaying instance or an instance from the remaining dataset as a control. We are blind to which group it comes from.

	Correct	Fine	Wrong	Unsure
MNLI^D	66%	17%	9%	5%
MNLI^C	86%	5%	5%	1%
SST-2^D	55%	8%	10%	25%
SST-2^C	88%	4%	0%	6%
QQP^D	60%	26%	10%	2%
QQP^C	87%	10%	1%	0%

Table 5: MINI vs. LARGE . We examine whether there are mislabels for the **D**ecaying fractions (superscript ^{D}) and the rest of the dataset (**C**ontrol group ^{C}). The decaying fraction contains more mislabels, but includes correct labels as well.

For each task of MNLI, QQP, and SST-2, the first author annotated 100 instances (decay + control group) (Table 5). We present all the annotated decaying instances in Appendix J.

Conclusion We find that the decaying fraction has more wrong or controversial labels, compared to the remaining instances. However, even after we adjust for the fraction of incorrect labels, the Decay fraction still exceeds the false discovery control. This implies that MINI models are better than LARGE models on some correctly labeled instances. The second author followed the same procedure and reproduced the same qualitative results.

However, we cannot find an interpretable pattern for these correctly labeled decaying instances by simply eyeballing. We discuss future directions to discover interpretable categories in Section 7.

5 Correlation of Instance Difference

We next investigate whether there is a momentum of instance accuracy increase: for example, if the instance accuracy improves from MINI(s_1) to MEDIUM(s_2), is it more likely to improve from MEDIUM(s_2) to LARGE(s_3)?

The naïve approach is to calculate the Pearson correlation coefficient between $\overset{\text{MINI}}{\text{MEDIUM}}\Delta\hat{\text{Acc}}$ and $\overset{\text{MEDIUM}}{\text{LARGE}}\Delta\hat{\text{Acc}}$, and we find the correlation to be zero. However, this is partly an artifact of accuracies being bounded in $[0, 1]$. If MEDIUM drastically improves over MINI from 0 to 1, there is no room for LARGE to improve over MEDIUM. To remove this inherent negative correlation, we calculate the correlation conditioned on the accuracy of the middle-sized model, $\hat{\text{Acc}}^{\text{MEDIUM}}$.

Therefore, we bucket instances by their estimated MEDIUM accuracy into intervals of size 0.1,

$(s_1, s_2, s_3) \downarrow$ Buckets$\rightarrow$	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
SMALL, MEDIUM, BASE	0.07	0.22	0.29	0.40	0.35	0.33	0.38	0.27	0.24	0.13
MINI, MEDIUM, LARGE	0.03	0.15	0.18	0.33	0.17	0.16	0.22	0.20	0.19	0.09

Table 6: Each row corresponds to a triplet of model sizes. Each column t represents a bucket that contains instances with $\hat{\text{Acc}}^{s_2} \in [t-0.1, t]$. Within each bucket, we calculate the Pearson correlation coefficient between the estimated accuracy improvements: ${}^{s_1}_{s_2}\Delta\hat{\text{Acc}}$ and ${}^{s_2}_{s_3}\Delta\hat{\text{Acc}}$. These correlations are positive and become higher when model size differences are small.

and we find the correlation to be positive within each bucket (Table 6, row 2). This fixes the problem with the naïve approach by getting rid of the negative correlation, which could have misled us to believe that improvements by larger models are uncorrelated.

We additionally find that the correlations between improvements become stronger when model size differences are smaller. Table 6 row 1 reports results for another model size triplet with smaller size difference, i.e. (s_1, s_2, s_3) = (SMALL, MEDIUM, BASE), and the correlation is larger for all buckets. Results for more tasks and size triplets are in Appendix G and the same conclusions hold qualitatively.

6 Variance at the Instance Level

Section 3 found that the overall accuracy has relatively low variance, but model predictions are noisy. This section formally analyzes variance at the instance level. For each instance, we decompose its loss into three components: Bias^2, variance due to pretraining randomness, and variance due to finetuning randomness. Formally, we consider the $0/1$ loss:

$$\mathcal{L}_i := 1 - c_i = (1 - c_i)^2, \tag{13}$$

where c_i is a random variable 0/1 indicating whether the prediction is correct or incorrect, with respect to randomness in pretraining and finetuning. Therefore, by bias-variance decomposition and total variance decomposition, we have

$$\mathcal{L}_i = \text{Bias}^2{}_i + \text{PretVar}_i + \text{FineVar}_i, \tag{14}$$

where, by using P and F as pretraining and finetuning random seeds:

$$\begin{aligned} \text{Bias}^2{}_i &:= (1 - \mathbb{E}_{P,F}[c_i])^2, \\ \text{PretVar}_i &:= \text{Var}_P[\mathbb{E}_F[c_i]], \\ \text{FineVar}_i &:= \mathbb{E}_P[\text{Var}_F[c_i]], \end{aligned} \tag{15}$$

capturing "how wrong is the average prediction", variance due to pretraining, and variance due to finetuning seeds, respectively.

	Bias^2	PretVar	FineVar
MINI	0.203	0.017	0.036
SMALL	0.179	0.017	0.036
MEDIUM	0.157	0.014	0.040
BASE	0.134	0.010	0.043
LARGE	0.111	0.007	0.043

Table 7: The bias, pretraining variance, and finetuning variance for each model size, averaged across all test instances. Finetuning variance is much larger than pretraining variance; larger models have larger finetuning variance.

We can directly estimate FineVar by first calculating the sample variance across finetuning runs for each pretraining seed, and then averaging the variances across the pretraining seeds. Estimating PretVar is more complicated. A naïve approach is to calculate the empirical variance, across pretraining seeds, of the average accuracy across finetuning seeds. However, the estimated average accuracy for each pretraining seed is noisy itself, which causes an upward bias on the PretVar estimate. We correct this bias by estimating the variance of the estimated average accuracy and subtracting it from the naïve estimate; see Appendix H for details, as well as a generalization to more than two sources of randomness. Finally, we estimate Bias^2 by subtracting the two variance estimates from the estimated loss.

For each of these three quantities, Bias^2, PretVar and FineVar, we estimate it for each instance, average it across all instances in the evaluation set, and report it in Table 7. The variances at the instance level are much larger than the variance of overall accuracy, by a factor of 1000.

We may conclude from Table 7 that larger models have larger finetuning variance and smaller pretraining variance. However, lower bias also inherently implies lower variance. To see this, suppose a model has perfect accuracy and hence zero bias; then it always predicts the same label (the correct one) and hence has zero variance. This might favor larger models and "underestimate" their variance,

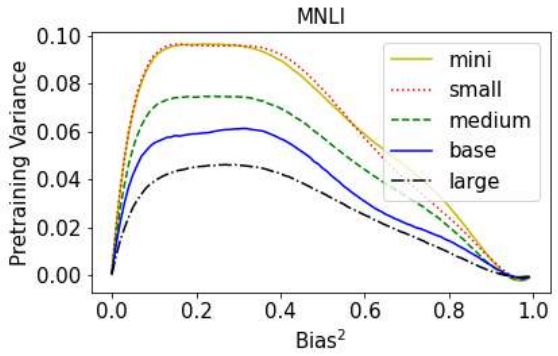


Figure 5: The pretraining variance conditioned on Bias^2 (the level of correctness). Each curve represents a model size. Larger models have lower pretraining variance across all levels of bias.

since they have lower bias. Therefore, we calculate and compare the variances conditioned on the bias, i.e. $\text{PretVar}(b^2) := \mathbb{E}_i[\text{PretVar}_i|\text{Bias}^2{}_i = b^2]$.

We estimate $\text{PretVar}^s(b^2)$ using Gaussian process regression and plot it against b^2 in Figure 5. We find that larger models still have lower pretraining variance across all levels of bias on the specific task of MNLI under the 0/1 loss. To further check whether our conclusions are general, we tested them on other tasks and under the squared loss $\mathcal{L}_i := (1 - p_i)^2$, where p_i is the probability assigned to the correct class. Below are the conclusions that generally hold across different tasks and loss functions.

Conclusion We find that 1) larger models have larger finetuning variance, 2) LARGE has smaller pretraining variance than BASE ; however, the ordering between other sizes varies across tasks and losses, and 3) finetuning variance is $2-8$ times as large as pretraining variance, and the ratio is bigger for larger models.

7 Discussion and Future Directions

To investigate model behaviors at the instance level, we produced massive amounts of model predictions in Section 2 and treated them as raw data. To extract insights from them, we developed better metrics and statistical tools, including a new method to control the false discoveries, an unbiased estimator for the decomposed variances, and metrics that compute variance and correlation of improvements conditioned on instance accuracy. We find that larger pretrained models are indeed worse on a non-trivial fraction of instances and have higher variance due to finetuning seeds; additionally, instance accuracy improvements from MINI to MEDIUM correlate with improvements from MEDIUM to LARGE .

Overall, we treated model prediction data as the central object and built analysis tools around them to obtain a finer understanding of model performance. We therefore refer to this paradigm as "**instance level understanding as data mining**". We discuss three key factors for this paradigm to thrive: 1) scalability and the cost of obtaining prediction data, 2) other information to collect for each instance, and 3) better statistical tools. We analyze each of these aspects below.

Scalability and Cost of Data Data mining is more powerful with more data. How easy is it to obtain more model predictions? In our paper, the main bottleneck is pretraining. However, once the pretrained models are released, individual researchers can download them and only need to repeat the cheaper finetuning procedure.

Furthermore, model prediction data are undershared: while many recent research papers share their code or even model weights to help reproduce the results, it is not yet a standard practice to share all the model predictions. Since many researches follow almost the same recipe of pretraining and finetuning (McCoy et al., 2020; Desai and Durrett, 2020; Dodge et al., 2020), much computation can be saved if model predictions are shared. On the other hand, as the state of the art model size is increasing at a staggering speed[7], most researchers will not be able to run inference on a single instance. The trend that models are becoming larger and more similar necessitate more prediction sharing.

Meta-Labels and Other Predictions Data mining is more powerful with more types of information. One way to add information to each instance is to assign "meta-labels". In the HANS (McCoy et al., 2019) dataset, the authors tag each instance with a heuristic [8] that holds for the training distribution but fails on this instance. Naik et al. (2018a) and Ribeiro et al. (2020) associate each instance with a particular stress test type or subgroup, for example, whether the instance requires the model to reason numerically or handle negations. Nie et al.

[7]e.g. BERT (Devlin et al., 2019) has 340M parameters, while Switch-Transformer has over 1 trillion parameters (Fedus et al., 2021).

[8]For example, "the label [entailment] is likely if the premise and the hypothesis have significant lexical overlap".

(2020) collects multiple human responses to estimate human disagreement for each instance. This meta-information can potentially help us identify interpretable patterns for the disagreeing instances where one model is better than the other. On the flip side, identifying disagreeing instances between two models can also help us generate hypothesis and decide what subgroup information to annotate.

We can also add performance information on other tasks to each instance. For example, Pruksachatkun et al. (2020) studied the correlation between syntactic probing accuracy (Hewitt and Liang, 2019) and downstream task performance. Turc et al. (2019) and Kaplan et al. (2020) studied the correlation between language modelling loss and the downstream task performance. However, they did not analyze correlations at the instance level. We may investigate whether their results hold on the instance level: if an instance is easier to tag by a probe or easier to predict by a larger language model, is the accuracy likely to be higher?

Statistical Tools Data mining is more powerful with better statistical tools. Initially we used the Benjamini-Hochberg procedure with Fisher's exact test, which required us to pretrain 10 models to formally verify that the decaying instances exist. However, we later realized that 2 is in fact enough by using our approach introduced in Section 4. We could have saved 80% of the computation for pretraining if this approach was known before we started.

Future work can explore more complicated metrics and settings. We compared at most 3 different model sizes at a time, and higher order comparisons require novel metrics. We studied two sources of randomness, pretraining and finetuning, but other sources of variation can be interesting as well, e.g. differences in pretraining corpus, different model checkpoints, etc. To deal with more sophisticated metrics, handle different sources and hierarchies of randomness, and reach conclusions that are robust to noises at the instance level, researchers need to develop new inference procedures.

To conclude, for better instance level understanding, we need to produce and share more prediction data, annotate more diverse linguistic properties, and develop better statistical tools to infer under noises. We hope our work can inform researchers about the core challenges underlying instance level understanding and inspire future work.

Acknowledgement

We thank Steven Cao, Cathy Chen, Frances Ding, David Gaddy, Colin Li, and Alex Wei for giving comments on the initial paper draft. We would also like to thank the Google Cloud TPU team for their hardware support.

References

Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. 2020. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*,

pages 2744–2751, Online. Association for Computational Linguistics.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. 2020. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E. Gonzalez. 2020. Train large, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on Machine Learning*.

R. Thomas McCoy, Junghyun Min, and Tal Linzen. 2020. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 217–227, Online. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018a. Stress test evaluation for natural language inference. In *The 27th International Conference on Computational Linguistics (COLING)*, Santa Fe, New Mexico, USA.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018b. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yixin Nie, Xiang Zhou, and Mohit Bansal. 2020. What can we learn from collective human opinions on natural language inference data? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9131–9143, Online. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. 2020. An investigation of why overparameterization exacerbates spurious correlations.

In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8346–8356. PMLR.

Lane Schwartz, Timothy Anderson, Jeremy Gwinnup, and Katherine Young. 2014. Machine translation and monolingual postediting: The AFRL WMT-14 system. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 186–194, Baltimore, Maryland, USA. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Adina Williams, Tiago Pimentel, Hagen Blix, Arya D. McCarthy, Eleanor Chodroff, and Ryan Cotterell. 2020. Predicting declension class from form and meaning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6682–6695, Online. Association for Computational Linguistics.

Summary

In this 4-page appendix, we include 1) details on model training and the datasets we used 2) high-level ideas underlying the theoretical results, and 3) relatively more important discussions. We also have a longer version of this appendix on our github, which contains the full proofs, discussions, plots, and tables.

A Pretraining and Finetuning Details

To obtain model predictions under the "pretraining and finetuning" framework (Devlin et al., 2019), we need to decide a model **size**, perform **pretraining**, finetune on a **training set** with a choice of **hyperparameters**, and test the model on an **evaluation set**. We discuss each bolded aspects below.

Size Similar to Turc et al. (2019), we experimented with the following five model sizes, listed in increasing order: MINI (L4/H256) [9] , SMALL (L4/H512), MEDIUM (L8/H512), BASE (L12/H768), and LARGE (L24/H1024).

Pretraining We used the pretraining code from Devlin et al. (2019) and the pre-training corpus from Li et al. (2020). Compared to the original BERT release, we used context size 128 instead of 512, since computation cost grows quadratically with respect to context size; we also pretrained for 2M steps instead of 1M.

Training Set We consider 3 datasets: Quora Question Pairs (QQP) [10], Multi-Genre Natural Language Inference (MNLI; Williams et al. (2020)), and the Stanford Sentiment Treebank (SST-2; (Socher et al., 2013)). For QQP we used the official training split. For MNLI we used 350K out of 400K instances from the original training split, and added the remaining 50K to the evaluation set, since the original in-domain development set only contains 10K examples. For SST-2, we mix the training and development set of the original split, split the instances into 5 folds, train on four of them, and evaluate on the remaining fold.

Hyperparameters As in Turc et al. (2019), we finetune 4 epochs for each dataset. For each task and model size, we tune hyperparameters in the following way: we first randomly split our new training set into 80% and 20%; then we finetune on the 80% split with all 9 combination of batch size [16, 32, 64] and learning rate [1e-4, 5e-5, 3e-5], and choose the combination that leads to the best average accuracy on the remaining 20%.

Evaluation Set After finetuning our pretrained models, we evaluate them on a range of in-domain, out-of-domain, or challenging datasets to obtain model predictions. Models trained on MNLI are also evaluated on Stanford Natural Language Inference (SNLI; (Bowman et al., 2015)), Heuristic Analysis for NLI Systems (HANS; (McCoy et al., 2019)), and stress test evaluations (STRESS; (Naik et al., 2018b)). Models trained on QQP are also evaluated on Twitter Paraphrase Database (TwitterPPDB; (Lan et al., 2017)).

Since pretraining introduces randomness, for each model size s, we pretrain 10 times with different random seed P; since finetuning also introduces noise, for each pretrained model we pretrain 5 times with different random seed F; besides, we also evaluate the model at the checkpoints after E epochs, where $E \in [3, 3\frac{1}{3}, 3\frac{2}{3}, 4]$.

Pretraining 10 models for all 5 model sizes altogether takes around 3840 hours on TPU v3 with 8 cores. Finetuning all of them 5 times for all three tasks in our paper requires around 1200 hours.

B Compare Our Models to the Original

Since we decreased the pre-training context length to save computation, these models are not exactly the same as the original BERT release by Devlin et al. (2019) and Turc et al. (2019). We need to benchmark our model against theirs to ensure that the performance of our model is still reasonable and the qualitative trend still holds. For each each size and task, we finetune the original model 5 times and calculate the average of overall accuracy.

The comparison can be seen in Table 8. We find that our model does not substantially differ from the original ones on QQP and SST-2. On MNLI, the performance of our BERT-BASE and BERT-LARGE is 2∼3% below the original release, but the qualitative trend that larger models have better accuracy still holds robustly.

C More Instance Difference Results

Similar to Figure 4, for all 10 pairs of model sizes and all in-distribution instances of MNLI, SST-2, and QQP, we plot the cumulative density of $\Delta\hat{\text{Acc}}$ and $\Delta\text{Acc}'$, or say, $\hat{\text{Decay}}(t)$ and $\text{Decay}'(t)$. See the long appendix for the figures.

[9] 4 Layers with hidden dimension 256

[10] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

	QQP	MNLI	SST-2
MINI ^{orig}	88.2%	74.6%	92.8%
MINI ^{ours}	87.3%	74.3%	92.8%
SMALL ^{orig}	89.1%	77.3%	93.9%
SMALL ^{ours}	88.7%	76.7%	93.9%
MEDIUM ^{orig}	89.8%	79.6%	94.2%
MEDIUM ^{ours}	89.5%	78.9%	94.2%
BASE ^{orig}	90.8%	83.8%	95.0%
BASE ^{ours}	90.6%	81.2%	94.6%
LARGE ^{orig}	91.3%	86.8%	95.2%
LARGE ^{ours}	91.0%	83.8%	94.8%

Table 8: Comparing our pretrained model (superscript ^{orig}) to the original release by Devlin et al. (2019) and Turc et al. (2019) (superscript ^{ours}). All pretrained models are finetuned with the training set and tested on the in-distribution evaluation set described in Appendix A.

Additionally, for each pair of model sizes s_1 and s_2, we estimate "how much instances are getting better/worse accuracy?" by taking the maximum difference between the red curve and the blue curve. We report these results for MNLI, SST-2, and QQP in Table 9. We find that larger model size gaps lead to larger decaying fraction, but also larger improving fraction as well.

D Proof of Theorem 1

Formal Setup Suppose each instance is indexed by i, the set of all instances is $\mathcal{T}$, and the random seed is R; then $c^s_R \in \{0,1\}^{|\mathcal{T}|}$ is a random $|\mathcal{T}|$ dimensional vector, where $c^s_{R,i} = 1$ if the model of size s correctly predicts instance i under the random seed R. We are comparing model size s_1 and s_2, where s_2 is larger; to keep notation uncluttered, we omit these indexes whenever possible.

Suppose we observe $c^{s_1}_{R_{1\ldots 2k}}$ and $c^{s_2}_{R_{2k+1\ldots 4k}}$, where there are $2k$ different random seeds for each model size [11]. Then

$$\Delta\hat{\text{Acc}}_i := \frac{1}{2k}\Big(\sum_{j=1}^{2k} c^{s_1}_{R_j,i} - \sum_{j=2k+1}^{4k} c^{s_2}_{R_j,i}\Big), \quad (16)$$

and hence the discovery rate $\hat{\text{Decay}}(t)$ is defined as

$$\hat{\text{Decay}}(t) := \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|} \mathbf{1}[\Delta\hat{\text{Acc}} \le t]. \quad (17)$$

[11] We assumed even number of random seeds since we will mix half of the models from each size to compute the random baseline

For the random baseline estimator, we have

$$\Delta\text{Acc}'_i := \frac{1}{2k}\Big(\sum_{j=1}^{k} c^{s_1}_{R_j,i} + \sum_{j=2k+1}^{3k} c^{s_2}_{R_j,i} - \sum_{j=k+1}^{2k} c^{s_1}_{R_j,i} - \sum_{j=3k+1}^{4k} c^{s_2}_{R_j,i}\Big), \quad (18)$$

and the false discovery control Decay' is defined as

$$\text{Decay}'(t) := \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|} \mathbf{1}[\Delta\text{Acc}'_i \le t]. \quad (19)$$

Formally, theorem states that

$$\text{Decay} \ge \mathbb{E}_{R_1\ldots R_{4k}}[\hat{\text{Decay}}(t) - \text{Decay}'(t)] \quad (20)$$

Proof By re-arranging terms and linearity of expectation, Equation 20 is equivalent to the following

$$\sum_{i=1}^{|\mathcal{T}|}\big(\mathbf{1}[\Delta\text{Acc}_i < 0] - \mathbb{P}[\Delta\hat{\text{Acc}}_i \le t] + \mathbb{P}[\Delta\text{Acc}'_i \le t]\big) \ge 0 \quad (21)$$

Hence, we can declare victory if we can prove that for all i,

$$\mathbf{1}[\Delta\text{Acc}_i < 0] - \mathbb{P}[\Delta\hat{\text{Acc}}_i \le t] + \mathbb{P}[\Delta\text{Acc}'_i \le t] \ge 0 \quad (22)$$

To prove Equation 22, we observe that if $\text{Acc}_i < 0$, since the probabilities are bounded by 0 and 1, its left-hand side must be positive. Therefore, we only need to prove that

$$\Delta\text{Acc}_i \ge 0 \Rightarrow \mathbb{P}[\Delta\text{Acc}'_i \le t] \ge \mathbb{P}[\Delta\hat{\text{Acc}}_i \le t], \quad (23)$$

which will be proved in Lemma 1.□

Lemma 1

$$\Delta\text{Acc}_i \ge 0 \Rightarrow \mathbb{P}[\Delta\text{Acc}'_i \le t] \ge \mathbb{P}[\Delta\hat{\text{Acc}}_i \le t], \quad (24)$$

For $m = 1, 2$, define

$$p^{s_m}_i := \mathbb{E}_R[c^{s_m}_i], \quad (25)$$

then

$$p^{s_1}_i \le p^{s_2}_i \quad (26)$$

MNLI $s_1 \setminus s_2$	MINI	SMALL	MEDIUM	BASE	LARGE
MINI	0.000	0.087	0.136	0.179	0.214
SMALL	0.033	0.000	0.089	0.139	0.180
MEDIUM	0.050	0.028	0.000	0.090	0.143
BASE	0.060	0.048	0.026	0.000	0.101
LARGE	0.059	0.052	0.040	0.021	0.000
QQP $s_1 \setminus s_2$	MINI	SMALL	MEDIUM	BASE	LARGE
MINI	0.000	0.057	0.076	0.100	0.107
SMALL	0.019	0.000	0.039	0.073	0.084
MEDIUM	0.029	0.014	0.000	0.044	0.063
BASE	0.034	0.027	0.016	0.000	0.032
LARGE	0.036	0.031	0.027	0.016	0.000
SST-2 $s_1 \setminus s_2$	MINI	SMALL	MEDIUM	BASE	LARGE
MINI	0.000	0.037	0.043	0.052	0.057
SMALL	0.010	0.000	0.015	0.031	0.036
MEDIUM	0.016	0.008	0.000	0.020	0.028
BASE	0.019	0.014	0.009	0.000	0.014
LARGE	0.020	0.017	0.015	0.008	0.000

Table 9: On QQP, MNLI in domain development set and SST-2 we lowerbound the fraction of instances that improves when model size changes from s_1 (row) to s_2 (column).

Since $c_i^{s_1}$ and $c_i^{s_2}$ are both Bernoulli random variables with rate $p_i^{s_1}$ and $p_i^{s_2}$ respectively, we can write down the probability distribution of $\Delta\hat{\text{Acc}}_i$ and $\Delta\text{Acc}'_i$ as the sum/difference of several binomial variables, i.e.

$$\Delta\hat{\text{Acc}}_i \sim (\text{Binom}(k, p_i^{s_2}) + \text{Binom}(k, p_i^{s_2}) - \text{Binom}(k, p_i^{s_1}) - \text{Binom}(k, p_i^{s_1}))/2k, \quad (27)$$

and

$$\Delta\text{Acc}'^{i} \sim (\text{Binom}(k, p^{s_2,i}) + \text{Binom}(k, p^{s_1,i}) - \text{Binom}(k, p^{s_1,i}) - \text{Binom}(k, p^{s_2,i}))/2k \quad (28)$$

$p_i^{s_1} \leq p_i^{s_2}$, $\text{Binom}(k, p^{s_2,i}))$ first order stochastically dominates $\text{Binom}(k, p^{s_1,i})$. Therefore, $\Delta\text{Acc}'^{i}$ dominates $\Delta\hat{\text{Acc}}_i$, hence completing the proof. □

D.1 Independent Seed Assumption

See the long appendix for discussion.

E Upward Bias of Adaptive Thresholds

In section 3 we picked the best threshold that can maximize the lowerbound, which can incur a slight upward bias. Here we estimate that the bias is at most 10% relative to the unbiased lowerbound with a bootstrapping method.

We use the empirical distribution of 10 pretrained models as the ground truth distribution for bootstrapping. We first compute a best threshold with 10 sampled smaller and larger pretrained models, and then compute the lowerbound L with this threshold on another sample of 10 smaller and larger models. Intuitively, we use one bootstrap sample (which contains 10 smaller pretrained models and 10 larger pretrained models) as the development set to "tune the threshold", and then use this threshold on a fresh bootstrap sample to compute the lowerbound. We refer to the lowerbound that uses the best threshold as L^*, and compute the relative error $\mathbb{E}[(L^* - L)]/\mathbb{E}[L)]$, where the expectation is taken with respect to bootstrap samples.

See the long appendix for more detailed results. In general, we find that the upward bias is negligible, which is at most around 10%.

F Comparison with Significance Testing

See the long appendix for the Table that compares our procedure with the classical method that uses the Fisher's exact test and Benjamini-Hochberg procedure for other tasks and model size pairs.

In general, we find that our method always provide a tighter (higher) lowerbound than the classical method, and 2 models are sufficient to verify the existence (i.e. lowerbound > 0) of the decaying fraction; in contrast, the classical method some-

times fails to do this even with 10 models, e.g., when comparing BASE to LARGE .

Intuitively, our approach provides a better lowerbound because it better makes use of the information that on most instances, both the smaller and the larger models agree and predict completely correctly or incorrectly[12]. For an extreme example, suppose we only observe 2 smaller models and 2 larger models, and infinite number of datapoints, whose predictions are independent. On 99.98% datapoints, both models have instance accuracy 1; on 0.01% datapoints, smaller model is completely correct while bigger completely wrong, while on the rest 0.01% smaller completely wrong but bigger completely correct. Setting threshold to be 2, our decay estimate $\hat{\text{Decay}}$ is 0.01%, while $\text{Decay}' = 0$: since the models either completely predict correct or wrongly, there is never a false discovery. Therefore, our method can provide the tightest lowerbound 0.01% in this case. On the other hand, since we only have 4 models in total, the lowest significance-level given by the fisher exact test is $17\% \gg 0.1\%$, hence the discovery made by the Benjamin-Hochberg procedure is 0.

G More Results on Momentum

See the long appendix for correlations with other model size triplets on other tasks.

H Loss Decomposition and Estimation

The core of the PretVar estimator builds on the following theorem:

Theorem 2 *Suppose $\mathcal{D}_k, k \in [\mathcal{F}]$ are independently sampled from the same distribution Ξ, which is a distribution of distributions; $\hat{\mu}_k$ is an unbiased estimator of $\mathbb{E}_{X\in\mathcal{D}_k}[X]$, and $\hat{\phi}_k$ to be an unbiased estimator of the variance of $\hat{\mu}_k$, then*

$$\hat{Var}_F = \frac{1}{\mathcal{F}-1}\sum_{k\in[\mathcal{F}]}(\hat{\mu}_k - \hat{\mu})^2 - \frac{1}{\mathcal{F}}\sum_{k\in[\mathcal{F}]}\hat{\phi}_k \tag{29}$$

is an unbiased estimator for

$$Var = Var_{\mathcal{D}\sim\Xi}[\mathbb{E}_{X\sim\mathcal{D}}[X]], \tag{30}$$

where

$$\hat{\mu} := \frac{1}{\mathcal{F}}\sum_{k\in[\mathcal{F}]}\hat{\mu}_k \tag{31}$$

In this estimator, the first term "pretends" that $\hat{\mu}_.$ are perfect estimator for the population mean and calculate the variance, while the second term corrects for the fact that the empirical mean estimation is not perfect. Notice the theorem only requires that $\hat{\mu}$ and $\hat{\phi}$ are unbiased, and is agnostic to the actual computation procedure by these estimators.

To estimate PretVar, we need $\hat{\mu}_k$ and $\hat{\phi}_k$. The first term is the empirical accuracy for each pretraining seed; the second is an unbiased estimator of the variance of empirical accuracy for each pretraining seed, which can be estimated by the sample variance of finetuning divided by the number of finetuning runs.

See the long Appendix for a more detailed proof, and also how to generalize this estimator to estimate arbitrary level of variance decomposition.

I Variance Conditioned on Bias

See the long appendix for more figures.

J Example Decaying Instances

Here we present some random annotated decaying instances on MNLI.

Premise : and that you're very much right but the jury may or may not see it that way so you get a little anticipate you know anxious there and go well you know
Hypothesis : Jury's operate without the benefit of an education in law.
Label : Neutral, **Category** : Correct

Premise : In fiscal year 2000, it reported estimated improper Medicare Fee-for-Service payments of $11.
Hypothesis : The payments were improper.
Label : Entailment, **Category** : Fine

Premise : INTEREST RATE - The price charged per unit of money borrowed per year, or other unit of time, usually expressed as a percentage.
Hypothesis : Interest rate is defined as the total amount of money borrowed.
Label : Entailment, **Category** : Wrong

[12]This is for intuition, though, and we do not need any assumption on the prior of instance accuracy, which requires a Bayes interpretation.