— 8 —

Forces against crisis

If machine learning thwarted scientific crisis, the question is why. Some powerful explanations emerge. Key are the social norms and practices of the community rather than statistical methodology.

Source: The Emerging Science of Machine Learning Benchmarks. M. Hardt, 2025. URL: https://mlbenchmarks.org. Compiled on 2025-11-02.

Our closer empirical look at the scientific crisis in the applied statistical science made one thing clear. By all accounts, many fields that apply statistical methods have been engulfed in a major replication crisis. Machine learning shares the Achilles heel of statistical measurement with these data-driven disciplines. In addition, the research ecosystem in machine learning is far from perfect. Yet, some central empirical phenomena are different and unique to machine learning.

If the core scientific mandate of machine learning is to identify better and better models, benchmarks seem to be doing a reasonable job at keeping track of progress. At least, they did in the ImageNet era. Replication studies in machine learning reveal that model rankings have internal validity and that they are often stable across different domains. This is a better outcome than we had reason to hope for given the formal guarantees of the holdout method.

The empirical reality begs an explanation. How could it be that a decade of competitive use of the ImageNet test set didn't go south? The excessive use of ImageNet in the deep learning era of the 2010s certainly looks like what Duda and Hart called *training on the testing set*: "a long series of refinements guided by the results of repeated testing". This concern is more than a hypothetical. We saw simple and practical demonstrations, supported by theoretical arguments, showing that it is entirely possible to overfit to test sets through repeated use. A few thousand evaluations are enough, in principle, to break the holdout method. The community has easily made hundreds of thousands of evaluations on popular test sets.

So, why didn't the benchmarking enterprise run off a cliff, leading to overspecialized results of little scientific value?

We now examine a few emerging explanations. These explanatory pieces all have something in common. They aren't so much about statistical method, but rather about how scientists operate individually and as a community. Several psychological and social patterns in the community turn out to be unexpected forces against the scientific crisis.

Throughout this chapter, the focus is on the internal validity of the iron rule. Why does competitive empirical testing on benchmarks clear the bar of replication on fresh data? Theoretically we can answer this question by proving stronger guarantees for the holdout method under different assumptions that capture how the community operates. We'll see a few such assumptions and theorems that follow from them.

Internal validity rules out that you can game a benchmark by climbing the leaderboard with spurious actions, such as ensembling random models as we discussed in Chapter 5. It's the first necessary step towards external validity. External validity of benchmark results is even less understood than internal validity. But at least there is an intuitive link between internal and external validity in benchmarking. If there is no spurious way to climb the leaderboard, your best strategy is to make honest progress. Real progress, in turn, should hold up in other domains as well.

Fully understanding the empirical phenomena is an ongoing research quest, full of intriguing open problems. This chapter provides a starting point.

8.1 The force of competition

A key characteristic of the machine learning community is its focus on achieving state-of-the-art results. State-of-the-art is whatever is at the top of the leaderboard and that's where the community's attention naturally gravitates toward. The basic research mechanic is *beating the previous best*. And, to oversimplify, that's primarily what the community cares about.

Competition over the top spots on the leaderboard is a unique aspect of the benchmarking research ecosystem. In other scientific applications of statistics, we may be interested in estimating numerical quantities, such as the treatment effect of a new intervention. In machine learning, researchers compete over the best model for a given task.

Competition over the top spot in a benchmark might seem narrow and in a sense anti-scientific. Shouldn't science explore more freely?

But if the scientific crisis had something to do with researchers degrees of freedom, we can also see the benefits of competitive testing. Competition over the top spot in a benchmark radically reduces researchers degrees of freedom. The dataset is fixed, the metric is fixed, and the evaluation protocol is fixed. In traditional benchmarks, like ImageNet, also the choice of training data is set in stone and agreed on. The only freedom is in the choice of the training algorithm and model architecture.

For a benchmark to work, it must be able to correctly identify the best model at any point in time under competitive pressure. Correctly identifying the best is also sufficient to get a full ranking. After all, the k-th best model is the best model after discarding the top $k-1$ models. So, we can rank by

repeatedly identifying the best.

As it turns out, identifying the best model from a sequence of adaptively chosen models is easier than accurately estimating the loss of each model. While estimating the loss of each model is *sufficient* for identifying the best, it is not *necessary*. To put it succinctly:

If all we care about is beating the previous best, there is no way to overuse a test set.

This is a theoretical result we prove next.

Leaderboard error

To make the point precise, we need a definition of what it means for a holdout mechanism to correctly identify the best model at any point in time. Given a sequence of models, we require that the holdout mechanism correctly estimates the *smallest risk* seen so far. This is different from trying to give an accurate answer to all requests. The mechanism might err—or decline to answer—on models that are nowhere near the best.

This relaxed requirement motivates the notion of *leaderboard error*. Leaderboard error is small if the holdout method keeps track of the risk of the best performing model over time, rather than the risk of all models ever evaluated.

Definition 1. *The* leaderboard error *of a holdout mechanism producing a sequence of estimates $R_1, \ldots, R_k$ given a sequence of adaptively chosen models $f_1, \ldots, f_k$ is defined as:*

$$\text{lberr}(R_1, \ldots, R_k) = \max_{\text{step } t \in \{1,\ldots,k\}} \left| R_t - \min_{\text{model } i \in \{1,\ldots,t\}} R(f_i) \right| .$$

Leaderboard error is small if at any point in time t, the estimate R_t is a good estimate of the smallest risk seen up until point t, i.e., $\min_{1 \le i \le t} R(f_i)$.

Operationally, if the leaderboard error is bounded by $\epsilon > 0$ and the holdout mechanism returns a value R_t such that $R_t < R_{t-1} - \epsilon$ we know that model f_t improved over all prior models $f_{t-1}, \ldots, f_1$. Note that the leaderboard error, like empirical risk, is a sample quantity, i.e., a function of the sample that the holdout mechanism works with.

There is a simple and natural holdout method that achieves small leaderboard error:

Given a new model, check if it beats the previous best by a significant amount on the test set. If so, update the previous best, otherwise do nothing.

We call this the *Ladder mechanism*. More precisely, for each given model the Ladder compares the empirical risk estimate of the model to the smallest empirical risk achieved by any model encountered previously. If the empirical risk is below the previous best by some margin, it announces the empirical risk of the current model and stores it as the best seen so far. However, if the empirical risk is not smaller by a margin, the Ladder simply continues to report the previous best. This reveals no new information other than that the model didn't improve.

We focus on risk with respect to the zero-one loss. However, the ideas apply more generally to any bounded loss function.

Ladder:
Given dataset S, threshold $\eta > 0$

- Assign initial top score $R_0 = 1$.
- For each round $t = 1, 2 \ldots$:
  1. Receive model $f_t \colon \mathcal{X} \to \mathcal{Y}$
  2. Compute the empirical risk $R_S(f_t)$.
     a. If $R_S(f_t) < R_{t-1} - \eta$, update top score to $R_t = R_S(f_t)$.
     b. Else keep previous top score $R_t = R_{t-1}$.
  3. Output current top score R_t

The next theorem shows that the Ladder algorithm indeed achieves small leaderboard error. The main feature of the error bound is a logarithmic dependence on the number k of models that we evaluate. This is the kind of dependence we get out of the Hoeffding bound in the *non-adaptive* case. However, here we get the logarithmic dependence even in the *adaptive* case, if we only care about identifying the best model.

Theorem 1. *For a suitably chosen threshold parameter, for any sequence of adaptively chosen models $f_1, \ldots, f_k$, the Ladder algorithm achieves with probability $1 - o(1)$:*

$$\mathrm{lberr}(R_1, \ldots, R_k) \leq O\left(\frac{\log^{1/3}(kn)}{n^{1/3}}\right).$$

The proof of the theorem follows from a variant of the adaptive tree argument introduced in Chapter 5. The new element here is that we carefully prune the tree so as to bound its size.

Proof. Set the threshold parameter to $\eta = \log^{1/3}(kn)/n^{1/3}$. This technical choice will become clear later. For this threshold η, we need to show that with probability $1 - o(1)$ we have for every round $t \in [k]$ the error bound

$$|R_S(f_t) - R(f_t)| \leq O(\eta) = O(\log^{1/3}(kn)/n^{1/3}).$$

Let $\mathcal{A}$ be the adaptive analyst generating the function sequence. The algorithm $\mathcal{A}$ naturally defines a rooted tree $\mathcal{T}$ of depth k recursively defined as follows:

1. The root is labeled by $f_1 = \mathcal{A}(\emptyset)$.
2. Each node at depth $1 < i \leq k$ corresponds to one realization $(h_1, r_1, \ldots, h_{i-1}, r_{i-1})$ of the tuple of random variables $(f_1, R_1, \ldots, f_{i-1}, R_{i-1})$ and is labeled by $h_i = \mathcal{A}(h_1, r_1, \ldots, h_{i-1}, r_{i-1})$. Its children are defined by each possible value of the output R_i of Ladder Mechanism on the sequence $h_1, r_1, \ldots, r_{i-1}, h_i$.

We are going to bound the size of the tree using properties of the Ladder algorithm. Let $B = (1/\eta + 1)\log(4k(n+1))$. We claim that the size of the tree satisfies $|\mathcal{T}| \leq 2^B$. To prove the claim, we will uniquely encode each node in the tree using B bits of information. The claim then follows directly.

This is called a compression argument and it goes as follows.

1. We use $\lceil \log k \rceil \leq \log(2k)$ bits to specify the depth of the node in the tree.
2. We then specify the index of each $i \in [k]$ for which the Ladder algorithm performs an "update" so that $R_i \leq R_{i-1} - \eta$ together with the value R_i.

The key observation is that since $R_i \in [0, 1]$ there can be at most $\lceil 1/\eta \rceil \leq (1/\eta) + 1$ many such steps. This is because the loss is lower bounded by 0 and decreases by η each time there is an update.

Moreover, there are at most $n + 1$ many possible values for R_i, since we're talking about the zero-one loss on a dataset of size n. Hence, specifying all such indices requires at most $(1/\eta + 1)(\log(n+1) + \log(2k))$ bits. These bits of information uniquely identify each node in the graph, since for every index i not explicitly listed we know that $R_i = R_{i-1}$. The total number of bits we used is:

$$(1/\eta + 1)(\log(n+1) + \log(2k)) + \log(2k) \leq (1/\eta + 1)\log(4k(n+1)) = B.$$

This establishes the claim we made. The theorem now follows by applying a union bound over all nodes in $\mathcal{T}$ and using Hoeffding's inequality for each fixed node. Let $\mathcal{F}$ be the set of all functions appearing in $\mathcal{T}$. By a union

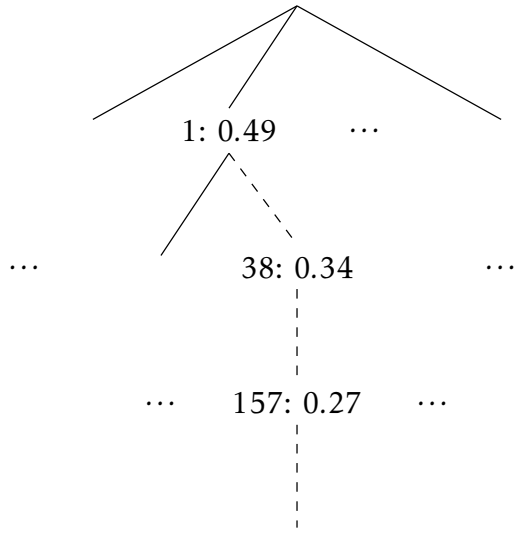

Figure 8.1: Low bit encoding of the adaptive tree. Dashed lines correspond to rounds with no update.

bound, we have

$$\begin{aligned}\mathbb{P}\{\exists f \in \mathcal{F} \colon |R(f) - R_S(f)| > \epsilon\} &\leq 2|\mathcal{F}|\exp(-2\epsilon^2 n) \\ &\leq 2\exp(-2\epsilon^2 n + B).\end{aligned}$$

Verify that by putting $\epsilon = 5\eta$, this expression can be upper bounded by $2\exp(-n^{1/3}) = o(1)$, and thus the claim follows.

□

The logarithmic dependence on the number of evaluations is great, but the term $n^{-1/3}$ that depends on the sample size doesn't meet the more familiar $n^{-1/2}$ rate. The exponent 1/3 in the error just doesn't seem quite right. A bit of work, however, can show that the analysis is tight for the Ladder algorithm. You can find a sequence of functions that forces an error matching the upper bound. But there could still be another algorithm that gets a better exponent.

We know from Chapter 2 that we're not going to beat the exponent 1/2. But between 1/3 and 1/2 we don't know what's possible. Although not obvious, a more sophisticated randomized version of the Ladder algorithm achieves the exponent 2/5. But that's the best researchers know to date. The following table summarizes the state of affairs.

Analyst	General estimation error	Leaderboard error
non-adaptive	$O(\sqrt{\log k/n})$	$O(\sqrt{\log k/n})$
adaptive	$O(\sqrt{k/n})$	$O((\log k/n)^{2/5})$

SOTA psychology

The Ladder algorithm shows how to guarantee benchmark integrity under an excessive number of model evaluations. Rather than guaranteeing accurate evaluations for all possible queries, the algorithm focuses on identifying the best. The mechanism couldn't be any simpler: *Check to see if you beat the previous best by some margin. If you did, publish the result. Otherwise, try again.*

The principle is so simple, in fact, that researchers could easily apply it in their minds. And perhaps they knowingly or unknowingly *do*—as sort of a mental *heuristic*. Heuristics always discard some information and avoid excessive computation in favor of simple ways of going about things. Humans excel at finding good heuristics to navigate complex systems.

So, let's postulate about the community. Imagine we're in a world where researchers primarily care if their model improved over state-of-the-art (SOTA). Call this community mindset "SOTA psychology". It's fundamentally the same principle that lies behind the iron rule. All scientific questions must ultimately be settled by competitive empirical testing. The best model wins and we care about little else.

In this view of the research community, benchmarks have the primary purpose to identify the best model. Rather than providing a numerical estimate of model accuracy for each model we evaluate on the test set, the goal is to identify the best model at any point in time. Even if syntactically researchers apply the standard holdout method, they primarily use the information from the test set to identify when they improved. If they didn't improve, they don't read much into it other than *I didn't improve.* This mental pattern is enough to implement the Ladder algorithm. The community reinforces this pattern by rewarding state-of-the-art results with conference publications, awards, and an increase in attention not afforded to results that didn't beat the previous best.

Consequently, there are two ways to think of the Ladder algorithm. One is *prescriptive*. It's an alternative holdout method you can actively and explicitly implement to avoid test set overuse in a benchmark or a competition. This is the typical way we think about algorithm design. In this view, algorithms

are interventions that change the way a running system works.

The other way to think about the Ladder algorithm and its analysis is as a *descriptive* model of the community. We can think of the Ladder algorithm as a kind of *natural algorithm* that the community implements on its own. From this perspective, the algorithm describes some aspect of how the community works. The guarantees of the algorithm then help to explain some empirical phenomena we observe.

8.2 Biases and heuristics

The Ladder algorithm is a heuristic resulting from an excessive focus on top results. It might seem narrow-minded as a scientific credo to only care about the best, but it has the benefit of protecting the test set from overuse. Ignorance is bliss, even if counterintuitive in the context of scientific work.

Heuristics are key to human decision making. Decades of research have compared human decision making to the old economic ideal of a *rational* decision maker. Rational agents compute optimal—that is, utility maximizing—solutions to decision problems using all available information. Rational agents all strategize alike. Optimality makes them interchangeable so that in theory we're left with one representative agent. The rational agent model—*homo economicus*—is the formal basis for large swaths of economic theory.

Humans almost never meet the economic ideal of rationality. You may be shocked to find out they are *not* all alike. They don't optimize perfectly. And they don't exhaustively use all available information when choosing what to have for dinner. This clash of common sense with economic theory is what motivated economist and AI pioneer Herb Simon to develop the theory of bounded rationality. Simon's work in turn influenced the *biases and heuristics* research program of Kahneman and Tversky. Core to this line of research is the idea of *cognitive biases*. Cognitive biases are systematic deviations from rationality commonly found in human decision making.

The last chapter touched on how cognitive biases can fuel the scientific crisis. Researchers, for example, are vulnerable to *confirmation bias*. Scientists tend to confirm what they think is true in the first place. They select evidence that supports their beliefs, while discarding information that is in conflict with their prior assumptions. Fischhoff's *hindsight bias* is another such culprit.[1] Once an uncertain event is resolved one way or the other, we mistakenly believe—with hindsight—that we predicted this outcome all

along. This cognitive pattern makes us dismiss valuable experimental data as obvious or unsurprising.

But cognitive shortcuts of the analyst can also mitigate test set overuse. A researcher is unlikely to behave like the worst-case analyst that gave us the pessimistic bounds in Chapter 5. In this worst case example, the analyst had to act on minor differences, such as distinguishing between a model achieving error 0.501 and one achieving error 0.499. This defies common sense. Moreover, the analyst had to look back at all the things ever tried and precisely put them together in some specific way. This again seems to run counter to intuitive human reasoning.

Try it out. Ask your office mates to multiply out $1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8$ in under 5 seconds. Because they'll likely run out of time, they'll make some approximation. Now ask another group of friends to multiply out the same sequence but in reverse order $8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$. They will again make some approximation. It turns out that humans tend to come up with an underestimate if they go about multiplying the numbers in natural order and an overestimate if they go in reverse. Kahneman and Tversky discovered this *bias* some fifty years ago and called it *anchoring effect*.[2] Humans rely heavily on initial pieces of information (*anchors*) when solving problems. This is the kind of observation that launched the study of cognitive biases.

A related observation is even older.[3] Humans tend to have a *recency bias* that assigns greater weight and significance to more recent events compared to earlier ones. A similar but broader cognitive tendency is known as the *availability heuristic*.[4] Humans instinctively prioritize and give undue weight to evidence that's easier to recall or retrieve, whether due to recency, convenience, prominence, or frequency of exposure.

Anchoring, recency, and availability all have something in common. When applied to a scientist reusing a test set, they limit the amount of information that can pass from the test set to the analyst. This in turn mitigates overfitting. It's as if the analyst made fewer evaluations. Just like the obsession with chasing state-of-the-art protects the test set, so do many cognitive biases.

We can make the intuition mathematically rigorous by modeling the analyst as a dynamical system.[5] The analyst has some internal mental state $h_{t+1} = \Psi(h_t, a_t)$ evolving according to state transition map Ψ. The state transition map depends on the internal state h_t and the holdout answer a_t that the analyst observes in response to the query $q_t = q_t(h_t)$ chosen based on the internal state. By making different assumptions on the state transition map, we can model different cognitive limitations. Consider, for example, a

contractive map satisfying

$$\|\Psi(h,a) - \Psi(h',a)\| \leq \lambda \|h - h'\|$$

for some scalar $\lambda \in (0,1)$. This corresponds to an analyst that discounts past information in each step, resulting in recency bias. We can dial λ between 0 and 1 to quantify the strength of the bias. What's neat is that this interpolates between the two extremes of a non-adaptive analyst and a fully adaptive analyst, giving us a continuum of generalization bounds from optimistic to pessimistic.

Gigerenzer and Brighton argue that heuristics that rely on less information and computation can often lead to better results—a *less is more* phenomenon.[6] We can see an instance of this general pattern in holdout reuse. A simple mental heuristic like the Ladder principle that discards information ultimately leads to better estimates. Likewise, different cognitive shortcuts effectively reduce test set leakage.

8.3 Rip van Winkle's replication problem

Limiting information is a powerful tool in arguing about test set reuse. Formally, the proof of the Ladder theorem is based on a *compression argument*: If we can show that at most B bits of information were revealed about the test set, then the generalization gap can be at most $O(\sqrt{B/n})$. The reason is that B bits of information can index at most $k = 2^B$ different functions. We can therefore apply Hoeffding's inequality with a union bound over $k = 2^B$ functions and make use of its $\sqrt{\log k} = \sqrt{B}$ dependence in the error. You have all the tools you need to formalize and prove this statement as an exercise.

Compression is a useful tool, since it works for arbitrary algorithms and also applies to the adaptive case. Any deterministic adaptive analyst using B bits of information from the test set can implicitly only search over 2^B functions, even if the analyst uses the test set adaptively. It makes no difference for the argument.

The compression argument has another application to benchmark longevity that starts from a lovely thought experiment proposed by researchers Arora and Zhang. Call it *Rip van Winkle's replication problem*.

Rip Van Winkle is an American folk tale about a Dutch-American villager who drinks a mysterious liquor in the New York state mountains, falls asleep for 20 years, and wakes to find he has slept through the American Revolution.

Imagine a modern day Rip who fell asleep in 2012, days after the release of ILSVRC-2012, and wakes up in 2022. Before Rip fell asleep, he trained linear models on top of handcrafted feature transforms. On ILSVRC-2012, these models would include the correct label among their top five predictions about 75% of the time, showing modest success in a metric called *top-5 accuracy*. Building highly accurate models for ImageNet looked like the holy grail of computer vision. When Rip woke up, however, end-to-end trained deep neural networks would get the correct label right with one guess about 90% of the time. People had stopped looking at top-5 accuracy, because it was too easy. To the extent that models made errors at all, it was largely due to ambiguous instances.[7] For academic purposes, researchers considered ImageNet *solved*.

Shocked by the developments, Rip rushes to get caught up. Now, how many bits of information about the ILSVRC-2012 test set do you have to tell Rip so that he can reproduce, say, ResNet-152?

Rip has everything you could've asked for in 2012, but not more than that. You need to describe whatever is new, the model architecture, the hyperparameters, and so forth. Since these design choices directly or indirectly depend on test set information, they potentially reveal information about the test set.

Arora and Zhang estimate that 1032 bits are enough to replicate ResNet-152 from the state of knowledge in 2012. With an optimized compression bound, this gives an upper bound on the generalization gap of at most 5%. Not bad!

There is a subtlety though. It would likely be quite hard to actually train ResNet-152 in the 2012 software ecosystem. Modern deep learning libraries and GPU optimizations were still missing. On the other hand, if we allow Rip to use the 2022 software ecosystem, we're cheating a little bit. After all, the software ecosystem co-evolved in tandem with progress on ImageNet. Various software developments happened with ImageNet in mind. It's not clear how much information from the test set leaked into the software ecosystem.

8.4 The force of the Blenheim Spaniel

How much can we overfit to the ImageNet test set if we really try? Let's put on our adversarial hat and try to come up with an *attack* on the ILSVRC

test set. Ignoring all better judgment, our only goal is to find a sequence of adaptive queries to the test set that drives up the generalization gap as much as possible. In Chapter 5, we saw such an attack for binary classification problems that works by ensembling random models. This attack is quite effective, forcing a large generalization gap with a few thousand queries on relatively large test sets. Can we do the same on ImageNet?

The attack from Chapter 5 was for binary classification problems and doesn't directly apply to ImageNet, since ILSVRC-2012 has 1000 classes. But it's not hard to generalize the idea. First, we pick k random classifiers as we did before. We evaluate each on the test set. Select the classifiers that have better than random chance accuracy on the test set. For $C = 1000$ classes, accuracy of random guessing equals $1/C$. So, we select the random classifiers that exceed accuracy $1/C$. We then ensemble the selected classifiers. To do so, replace the majority vote with a *plurality* vote that picks the most frequently predicted label on each data point.

Code it up and try it out. You'll be surprised to find out how poorly it performs. The attack barely gets off the ground. Although the algorithmic idea of ensembling random functions generalizes from two classes to any number of classes, it somehow just doesn't work well at all. This is for a fundamental reason. Ensembling random functions in the binary case forced a generalization gap around $\sqrt{k/n}$ where k is the number of functions and n is the sample size. But in the case of C classes, the analogous attack only achieves generalization gap about $C^{-1}\sqrt{k/n}$. It's worse by a factor C. So, for 1000 classes, we'd need one million times as many queries to achieve the same error as in the binary case.

There is a more clever attack that forces a generalization gap $\sqrt{k/Cn}$. But that is where it ends. No attack in the multi-class setting can do asymptotically better than this (up to perhaps a logarithmic factor). Having many classes therefore mitigates the risk of overfitting due to test set reuse. The larger the number of classes, the harder it is to overfit to a test set. It's as though the sample size increases from n to Cn. For ImageNet ILSVRC-2012 that's a factor 1000 improvement in effective sample size. It's like having a test set of size 50,000,000.

There is only one catch. These bounds are for an analyst that knows nothing about the test set and just starts querying the holdout from scratch. This is the same setup as the method we saw in Chapter 5 that overfits by ensembling random functions. What the above bounds show is that any such method no longer works well on a task with many classes.

In practice, however, researchers always have some prior information about the test set. In particular, we might already know a model that performs reasonably well on the test set. This gives us some good guess to start from about what the labels in the test set are. So, to make our attack more powerful, we give it access to a good model to begin with. What this means is that our attack doesn't have to start from scratch. It can utilize good predictions.

But how exactly can we use a well-performing model to overfit with fewer queries? There are at least three increasingly effective strategies:

1. The attacker uses the model's logits as the prior information about what the unknown labels are in the test set. This gives only a minor improvement over a prior-free attack.
2. The attacker uses the model's logits to restrict the attack to a subset of the test set corresponding to the lowest "confidence" points. This strategy gives modest improvements over a prior-free attack.
3. The attacker can exploit the fact that the model has good top-R accuracy, meaning that, for every image, the R highest weighted categories are likely to contain the correct class label. The attacker then focuses only on selecting from the top R predicted classes for each point. For $R = 2$, this effectively reduces class count to the binary case.

While the first two show only minor improvements over the baseline, the third idea has legs. Based on this idea, researchers came up with a clever attack on the ImageNet ILSVRC test set that forces a 3% generalization error with 5000 queries.[8] This is not too far from the Rip van Winkle estimate.

You might wonder if we can prevent overfitting by spuriously copying two classes into many classes. Unfortunately, this alone doesn't help since a good model for the problem would have good top-2 accuracy, therefore reducing the problem to the binary case. For multiple classes to help, the classes have to be reasonably hard to distinguish.

What's the moral of this story? ImageNet curiously contains some 120 different dog breeds, such as the Appenzeller, Blenheim Spaniel, Otterhound, Bluetick Coonhound, Groenendael, and the Schipperke, to name a few. This benchmark artifact might seem frivolous. After all, most humans struggle to identify any of these. Unexpectedly, though, having many similar classes makes it harder—not easier—to overfit to the test set.

8.5 Code and collaboration

Researchers compete but they collaborate just as much. The impressive software ecosystem growing around benchmarks makes it easy to share resources for model development. You'd never start from scratch if you tried out a new idea. You'll always take the most similar piece of code you can find and incrementally tweak it.

Collaboration and code sharing mean that researchers try out similar things. We can see this empirically by looking at *model similarity*. Suppose one model has 90% accuracy and another model has 75% accuracy. What's the chance that the two models agree on a randomly chosen point? If each model erred on a random subset of points independently, we'd see the two models agree on a randomly chosen point with probability

$$0.9 \times 0.75 + 0.1 \times 0.25 = 0.7\,.$$

This is, in a sense, the baseline level of agreement that follows from the accuracy numbers if errors are independent and random. Highly accurate models must agree a lot. Two models are more similar than expected if the agreement is significantly higher than this baseline value.

We can look at a set of ImageNet models and ask, what is the average similarity between pairs of models from this set. ImageNet models turn out to have higher than expected model similarity. This has interesting theoretical consequences. Evaluating models with high similarity is like evaluating fewer models. There can only be so many functions that agree a lot. As a result, it's possible to prove stronger generalization bounds for the holdout method for model families of high similarity. This works for in the non-adaptive as well as the adaptive case.

To summarize, dataset and code sharing lead to high model similarity, which in turn mitigates test set overuse somewhat.

Donoho coins the term *frictionless reproducibility* to describe the advances in the data and software ecosystem that power applied machine learning research. Frictionless reproducibility has three components:

1. Dataset creation
2. Code ecosystem enabling easy re-execution
3. Challenges, including both data science competition and machine learning benchmarks.

These aspects of machine learning research deserve attention. Whereas model architectures and optimization methods often take the limelight,

the software ecosystem around machine learning benchmarks is the quiet infrastructure that we all use but often take for granted.

Frictionless reproducibility acts as a lubricant in the gears of a scientific machine built after the iron rule. If we're going to settle all disputes by competitive empirical testing, it helps that we can carry out these empirical tests efficiently. Lower friction allows for greater activity. The software ecosystem around benchmarks has greatly increased productivity.

Take the data science platform Kaggle as an example. In its heyday, Kaggle organized hundreds of popular data science competitions. What started as a competition platform grew into a general data science community hub. Kaggle checks all three boxes of frictionless reproducibility. The datasets are easily available. The code is available and ready to run; Kaggle even provides Docker images—virtual containers to run reproducible workflows. The platform also offers a steady stream of challenges, with public leaderboards that gamify the experience. Cash prizes incentivize participants. Kaggle doesn't just meet the criteria for frictionless reproducibility—it excels at all three.

It's clear that Kaggle and similar platforms promote productivity. But to add a word of caution, there's a difference between productivity and scientific advances. Productivity doesn't guarantee scientific value. In particular, Kaggle competitions rarely produce novel techniques. Winning submissions in competitions often tweak standard methods, such as gradient boosting, problem-specific feature engineering, and ensembling. The primary model building lessons from years of ML competitions is that gradient boosting is the go to choice for most competitions.[9,10]

Code from competitions tends to be of little reusability value. Companies sponsoring competitions rarely ended up using the winning submissions in their own systems. Netflix notably discarded the code from its $1 million Netflix Prize competition, one of the most widely publicized machine learning competitions, held years before Kaggle launched.[11] Companies were more interested in branding and recruiting than in the code itself. Kaggle recognized this early on. Before its acquisition by Google in 2017, the startup had pivoted from competition platform to community hub. What mattered wasn't the output of the competitions, but the people competing in them.

Lack of friction alone doesn't create value. Conversely, scientists often overcome significant friction in pursuit of scientific advances. Return to ImageNet. Early work on ImageNet was relatively high friction. Predating the software we use today, building the breakthrough AlexNet architecture

required writing low-level GPU optimizations from scratch. Especially in the early days, training and testing on ImageNet was slow. Reproducing the results from other labs was tedious, if at all possible. Yet, progress was steepest in those early days. ResNets, one of the greatest highlights of era, came long before HuggingFace, PyTorch, or Weights and Biases. The development of residual networks even predates the first public release of TensorFlow. The first release of the ResNet code was written in Caffe, an academic deep learning framework mostly senior researchers still remember. The early ImageNet model development happened despite relatively high friction. The same is true for the early large language model development that we'll get to in Chapter 10.

8.6 From internal validity to progress

Several theoretical results speak to the strong internal validity of machine learning benchmarks. The holdout method and its variants often do better than the results from Chapters 4 and 5 suggest. Key to these results are psychological and social factors in how individual scientists and the community as a whole engage with benchmarks. Focusing on beating the previous best promotes benchmark longevity. Mental heuristics and shortcuts may prevent overfitting. Practices like code sharing give test sets higher mileage. Seemingly mundane problem artifacts can curb the possibility of climbing a leaderboard through spurious improvements. All of these results give guarantees that limit the extent to which we can deliberately or inadvertently climb a leaderboard with actions that don't correspond to progress on the task. Benchmarks turn out to be more resilient than we had reason to believe.

Successful benchmark design has two components.

First, we need to make sure that there is at least one interesting solution to the problem. In some domains—like image classification—this is easy to guarantee, because we can verify that humans solve the task with high accuracy. In other domains, this is not as straightforward as it sounds. The Fragile Families Challenge, for example, was a well-run machine learning competition about predicting life outcomes. But the dataset seemed to contain no signals that allowed for models better than baseline predictions.[12] When benchmarks go wrong, it's often because there isn't a good solution.

This first part of benchmark design is a matter of domain knowledge. We need to have substantive reasons to believe that there is an interesting

solution to the benchmark problem. Benchmarks can also go wrong, if there are multiple ways of being good at the task, of which only some are interesting. In this case, we might end up with the uninteresting solution. If you unleash the community on a benchmark, you will eventually find a risk minimizer, but you can't choose which one you'll get. You need to set up the problem so that you're happy with any risk minimizer. For example, there seems to be no easy way to do image classification on a sufficiently large and diverse datasets. Whatever the dataset—be it ImageNet or ImageNot—achieving high accuracy forces the model to do something impressive.

For the second component of benchmarking, we need to make sure that climbing the leaderboard corresponds to progress toward low risk on the task. If there are easy ways to climb the leaderboard without improving on the task, it's likely that one way or the other competitors will—knowingly or unknowingly—exploit these channels. This is where the results of this chapter come in. There are several factors that limit the scope of spurious improvements.

Put together, if we curb spurious improvements and there is at least one good solution, we have reason to hope that leaderboard climbing converges to the good solution.

Notes

Blum and Hardt introduced the Ladder algorithm and the definition of leaderboard error.[13] Hardt gave improved bounds on the leaderboard error using noise addition ideas from differential privacy.[14] Zrnic and Hardt made the connection between cognitive biases and overfitting.[5] Arora and Zhang contributed the Rip van Winkle thought experiment and calculated the compression bounds for ImageNet.[15] Compression bounds in the context of adaptive data analysis appeared in the work of Dwork et al.[16] Mania et al.[17] study the role of model similarity in mitigating test set overuse. Mania and Sra offer a theoretical explanation of *on the line* phenomenon in terms of model similarity and distributional closeness.[18]

Feldman, Frostig, Hardt studied the problem of adaptive test set reuse in the multi-class setting, providing both theoretical bounds, as well as the empirical results for ImageNet that I discussed.[8,19] In particular, they showed how an adaptive analyst can force an error of $\Omega(\sqrt{k/nC^2})$. This bound follows from a boosting argument that is, roughly speaking, a more sophisticated version of the ensembling argument in Chapter 5. Acharya and

Suresh[20] achieved the nearly optimal bound $\Omega(\sqrt{k/nC})$ using a sophisticated adaptive strategy.

Donoho's article *data science at the singularity* develops the argument for frictionless reproducibility, emphasizing its role in the progress of empirical machine learning.[21] Chazelle coined the term *natural algorithms* to describe algorithms that nature implements on its own.[22,23]

Simon developed bounded rationality in the 1950s.[24–26] Kahneman and Tversky developed and popularized the idea of behavioral biases.[2,27,28] These ideas have been hugely influential, spawning research fields in several disciplines. Gigerenzer argues against the tendency in behavioral economics to overemphasize biases as flaws in human decision making, a problem he calls *bias bias*, pointing out how human heuristics often successfully navigate complex scenarios.[29]

Bibliography

1. Fischhoff, B. Hindsight ≠ foresight: the effect of outcome knowledge on judgment under uncertainty. *Journal of Experimental Psychology: Human Perception and Performance* **1,** 288–299 (1975) (↑ 9).
2. Tversky, A. & Kahneman, D. Judgment under Uncertainty: Heuristics and Biases. *Science* **185,** 1124–1131. https://api.semanticscholar.org/CorpusID:6196452 (1974) (↑ 10, 19).
3. Murdock Jr, B. B. The serial position effect of free recall. *Journal of Experimental Psychology* **64,** 482–488 (1962) (↑ 10).
4. Tversky, A. & Kahneman, D. Availability: A heuristic for judging frequency and probability. *Cognitive Psychology* **5,** 207–232 (1973) (↑ 10).
5. Zrnic, T. & Hardt, M. *Natural analysts in adaptive data analysis* in *International Conference on Machine Learning (ICML)* (2019), 7703–7711 (↑ 10, 18).
6. Gigerenzer, G. & Brighton, H. Homo heuristicus: Why biased minds make better inferences. *Topics in cognitive science* **1,** 107–143 (2009) (↑ 11).
7. Vasudevan, V., Caine, B., Gontijo Lopes, R., Fridovich-Keil, S. & Roelofs, R. *When does dough become a bagel? analyzing the remaining mistakes on imagenet* in *Neural Information Processing Systems (NeurIPS)* (2022), 6720–6734 (↑ 12).
8. Feldman, V., Frostig, R. & Hardt, M. *The advantages of multiple classes for reducing overfitting from test set reuse* in *International Conference on Machine Learning (ICML)* (2019), 1892–1900 (↑ 14, 18).
9. Fernández-Delgado, M., Cernadas, E., Barro, S. & Amorim, D. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research* **15,** 3133–3181 (2014) (↑ 16).
10. Grinsztajn, L., Oyallon, E. & Varoquaux, G. *Why do tree-based models still outperform deep learning on typical tabular data?* in *Neural Information Processing Systems (NeurIPS)* (2022), 507–520 (↑ 16).
11. Johnston, C. Netflix Never Used Its $1 Million Algorithm Due to Engineering Costs. *Wired*. https://www.wired.com/2012/04/netflix-prize-costs/ (2012) (↑ 16).
12. Salganik, M. J. *et al.* Measuring the predictability of life outcomes with a scientific mass collaboration. *Proc. National Academy of Sciences* **117,** 8398–8403 (2020) (↑ 17).
13. Blum, A. & Hardt, M. *The ladder: A reliable leaderboard for machine learning competitions* in *International Conference on Machine Learning (ICML)* (2015), 1006–1014 (↑ 18).
14. Hardt, M. Climbing a shaky ladder: Better adaptive risk estimation. *arXiv:1706.02733* (2017) (↑ 18).

15. Arora, S. & Zhang, Y. Rip van Winkle's Razor: A Simple Estimate of Overfit to Test Data. *arXiv:2102.13189* (2021) (↑ 18).
16. Dwork, C. *et al. Generalization in adaptive data analysis and holdout reuse* in *Neural Information Processing Systems (NeurIPS)* (2015) (↑ 18).
17. Mania, H., Miller, J., Schmidt, L., Hardt, M. & Recht, B. *Model similarity mitigates test set overuse* in *Neural Information Processing Systems (NeurIPS)* (2019) (↑ 18).
18. Mania, H. & Sra, S. Why do classifier accuracies show linear trends under distribution shift? *arXiv:2012.15483* (2020) (↑ 18).
19. Feldman, V., Frostig, R. & Hardt, M. *Open Problem: How fast can a multiclass test set be overfit?* in *Conference on Learning Theory* (2019), 3185–3189 (↑ 18).
20. Acharya, J. & Suresh, A. T. *Optimal multiclass overfitting by sequence reconstruction from hamming queries* in *Algorithmic Learning Theory* (2020), 3–21 (↑ 19).
21. Donoho, D. Data science at the singularity. *Harvard Data Science Review* **6** (2024) (↑ 19).
22. Chazelle, B. *Natural algorithms* in *ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2009), 422–431 (↑ 19).
23. Chazelle, B. Natural algorithms and influence systems. *Communications of the ACM* **55,** 101–110 (2012) (↑ 19).
24. Simon, H. A. A behavioral model of rational choice. *The quarterly journal of economics*, 99–118 (1955) (↑ 19).
25. Simon, H. A. *Models of man: social and rational; mathematical essays on rational human behavior in society setting* (Wiley, 1957) (↑ 19).
26. Simon, H. A. Theories of bounded rationality. *Decision and organization* **1,** 161–176 (1972) (↑ 19).
27. Kahneman, D., Slovic, P. & Tversky, A. *Judgment under uncertainty: Heuristics and biases* (Cambridge University Press, 1982) (↑ 19).
28. Kahneman, D. & Tversky, A. in *Handbook of the fundamentals of financial decision making: Part I* 99–127 (World Scientific, 2013) (↑ 19).
29. Gigerenzer, G. The bias bias in behavioral economics. *Review of Behavioral Economics* **5,** 303–336 (2018) (↑ 19).