# Hyperparameter Optimization Is Deceiving Us, and How to Stop It

**A. Feder Cooper** [1] **Yucheng Lu** [1] **Christopher De Sa** [1]

*I will suppose...an evil genius, supremely powerful and clever, who has directed his entire effort at deceiving me. I will regard the heavens, the air, the earth, colors, shapes, sounds, and all external things as nothing but the bedeviling hoaxes of my dreams, with which he lays snares for my credulity...even if it is not within my power to know anything true, it certainly is within my power to take care resolutely to withhold my assent to what is false, lest this deceiver, however powerful, however clever he may be, have any effect on me.* — Descartes

## Abstract

While hyperparameter optimization (HPO) is known to greatly impact learning algorithm performance, it is often treated as an empirical afterthought. Recent empirical works have highlighted the risk of this second-rate treatment of HPO. They show that inconsistent performance results, based on choice of hyperparameter subspace to search, are a widespread problem in ML research. When comparing two algorithms, $\mathcal{J}$ and $\mathcal{K}$, searching one subspace can yield the conclusion that $\mathcal{J}$ outperforms $\mathcal{K}$, whereas searching another can entail the opposite result. In short, your choice of hyperparameters can deceive you. We provide a theoretical complement to this prior work: We analytically characterize this problem, which we term *hyperparameter deception*, and show that grid search is inherently deceptive. We prove a defense with guarantees against deception, and demonstrate a defense in practice.

## 1. Introduction

Unlike the learned output parameters of a ML model, hyperparameters are inputs provided to the learning algorithm that guide the learning process (Claesen & Moor, 2015; Feurer & Hutter, 2019). While hyperparameters are known to greatly influence overall ML algorithm performance (e.g., convergence rate, correctness, generalizability), research papers that do not focus explicitly on mechanisms of hyperparameter optimization (HPO) tend to treat hyperparameter selection as an empirical afterthought (Bergstra & Bengio, 2012; Sivaprasad et al., 2020; Melis et al., 2018). The learning problem is of primary theoretical interest. HPO particulars are relegated to the realm of empirical curiosity (Wilson et al., 2017; Schneider et al., 2019; Chen et al., 2018) — "just an engineering problem" at the boundary of what ML research treats as "real science" (Gieryn, 1983).

This is evident from common practices in the literature to conduct HPO: It is typical to pick a small subspace of possible hyperparameters and to perform grid search or random search over that subspace. This involves comparing the empirical performance of the resulting trained models, and then deploying or reporting on the model that performs best in terms of a chosen validation metric (Feurer & Hutter, 2019; John, 1994; Hsu et al., 2003; Wilson et al., 2017; Larochelle et al., 2007). For grid search, the points in the subspace constituting the grid are often manually set to "folklore" parameters—values put forth in now-classic papers as good rules-of-thumb concerning, for example, how to set the learning rate (Larochelle et al., 2007; LeCun et al., 1998; Hinton, 2012; Pedregosa et al., 2011).

This kind of ad-hoc decision-making in HPO, while generally accepted in the ML community, does not reflect a search for scientific truth about performance. The chosen hyperparameters might appear to entail application-specific, desirable empirical performance; however, this process provides no actual insights or assurances concerning whether the selected hyperparameters are the optimal choice.

Why does the ML community accept this rather capricious standard for HPO, especially since it is known to have a huge impact on performance? One answer is that HPO is expensive. The search space is enormous, suffering from the curse of dimensionality; learning, which is also expensive, has to be run for each hyperparameter configuration tested. There are clear efficiency benefits from using an easy-to-implement method like grid search with grid points that other published papers have already vetted, as opposed to more complex methods (Bergstra et al., 2011). Another

[1] Department of Computer Science, Cornell University, Ithaca, New York, United States. Correspondence to: A. Feder Cooper <afc78@cornell.edu>, Yucheng Lu <yl2967@cornell.edu>, Christopher De Sa <cdesa@cs.cornell.edu>.

answer is that there has not been a strong push to do HPO differently. To publish, researchers are incentivized to show that their new method performs better than a baseline from prior work. In the degenerate case, much like p-hacking in statistics (Gelman & Loken, 2019), this can involve tweaking the hyperparameters or grid search points until yielding a positive result. If the performance results are "good enough," then it might not be worth the effort to evaluate whether the hyperparameter choices are suboptimal; as long as the HPO procedure "seems reasonable," a researcher can pass review.

Taken together, these factors have created an environment in which HPO is neglected relative to the "flashier" learning problem. Yet there is now an emerging consensus to stop neglecting it. In recent years, many have argued that insufficient attention has been paid to the origins of empirical gains in ML—that it is often impossible to disentangle whether measured performance improvements are due to properties inherent to the learning algorithm under examination or to well-chosen (or lucky) hyperparameters (Lipton & Steinhardt, 2018; Musgrave et al., 2020; Sivaprasad et al., 2020; Choi et al., 2019; Bouthillier et al., 2019). Yet, this prior work does not suggest a potential path forward for addressing this problem with theoretical rigor.

We argue that **the process of drawing conclusions using HPO should itself be an object of study**. Our contribution is to put forward, to the best of our knowledge, the first theoretically-backed characterization for making conclusions we can trust about algorithm performance using HPO. We address theoretically the following empirically-observed problem: When comparing two algorithms, $\mathcal{J}$ and $\mathcal{K}$, searching one subspace can pick hyperparameters that yield the conclusion that $\mathcal{J}$ outperforms $\mathcal{K}$, whereas searching another can select hyperparameters that entail the opposite result. In short, your choice of hyperparameters can deceive you—a problem that we term *hyperparameter deception*. We formalize this problem and prove a defense to counteract it. In summary:

- We provide an intuition for the problem of hyperparameter deception (Section 3).
- We offer a formal definition for the process of drawing conclusions from running HPO, which we call epistemic HPO (EHPO) (Section 4).
- We characterize a logical system for reasoning about EHPO, which enables us to reason formally about hyperparameter deception (Section 5).
- We exercise the utility of our formalization: It can describe scenarios of deception and enable guaranteeing a method of defense against it (Section 6).

## 2. Hyperparameter Optimization

We start by providing the necessary background on hyperparameter optimization. Running supervised learning is often thought of as a double-loop optimization problem, $H$:

$$\arg\min_{\lambda \in \Lambda} \; \mathbb{E}_x[\mathcal{L}_{\text{HPO}}(x; \mathcal{A}_\lambda(\mathcal{M}_\lambda, X_{\text{train}})] = \lambda^* \qquad (1)$$

The inner-loop optimization problem, marked in blue, is what is typically called "training." It learns the parameters $\theta$ of some model $\mathcal{M}_\lambda$ by running a training algorithm $\mathcal{A}_\lambda$ on a training set $X_{\text{train}}$. This is usually done to minimize some training loss function $\mathcal{L}_{\text{train}}$ via an algorithm such as stochastic gradient descent. Both the inner-loop training algorithm and the model are parameterized by a vector of *hyperparameters* $\lambda$ (e.g. the learning rate and network size).

The outer-loop optimization problem is to find hyperparameters $\lambda^*$ from a set of allowable hyperparameters $\Lambda$: $\lambda^*$ results in a trained model that performs the best in expectation on "fresh" examples $x$ drawn from the same source as the training set, as measured by some loss $\mathcal{L}_{\text{HPO}}$. An algorithm $H$ that attempts this task is called a *hyperparameter optimization* (HPO) procedure. In practice, HPO is complicated by the fact that, since we do not have access to the distribution from which the $x$ are sampled, we cannot calculate the expected loss exactly. Instead, it is standard to use a validation dataset distinct from the training dataset to compute an approximation, which is used to determine $\lambda^*$.

From these definitions comes the natural question: How do we pick the $\Lambda$ within which $H$ looks for the best-performing $\lambda^*$? Often, $\Lambda$ is hand-picked: It is common to refer to "folklore parameters" suggested by classic papers as good "tips" and "tricks" for yielding good performance (LeCun et al., 1998; Hinton, 2012). $H$ in this case involves manually testing these popular options and selecting the $\lambda$ that performs best on the chosen validation metric. More principled methods include grid search, which has been used for decades (John, 1994), and random search, popularized by Bergstra & Bengio (2012). For the former, HPO evaluates $\mathcal{A}_\lambda$ on a grid of hyperparameter values $\lambda$, constructed by picking a set for each hyperparameter and taking the Cartesian product. For the latter, the values in each tested configuration $\lambda$ are randomly sampled from chosen distributions.

Importantly, both of these algorithms are parameterized themselves: Random search requires distributions from which to sample and grid search requires inputting the spacing between different configuration points in the grid. We call these HPO-procedure-input values *hyper-hyperparameters*. With this, we can define HPO formally.

**Definition 1.** *An HPO procedure $H$ is a tuple $(H_*, \mathcal{S}, \Lambda, \mathcal{A}, \mathcal{M}, X)$ where $H_*$ is a randomized algorithm, $\mathcal{S}$ is a set of allowable hyper-hyperparameters, $\Lambda$ is a set of allowable hyperparameters, $\mathcal{A}$ is a learning algorithm, $\mathcal{M}$ is a model, and $X$ is some dataset (usually split into train and validation sets). When run, $H_*$ takes as input a hyper-hyperparameter configuration $s \in \mathcal{S}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}$ using data from $X$) some number of times* *for different hyperparameters $\lambda \in \Lambda$. Finally, $H_*$ outputs a tuple $(\lambda^*, \theta^*, T, \ell)$, where $\lambda^*$ is the optimal hyperparameter choice, $\theta^*$ are the corresponding parameters found for $\mathcal{M}_\lambda$, $T$ is the total amount of* time *the HPO algorithm took to run, and $\ell$ is a* ***log*** *that records all the choices and measurements made during HPO. The log has all of the information necessary to make running $H$ reproducible.*

The log can be thought of as everything we need to produce the results of a data table in a research paper and the code used to produce those results: the random seed used, the choice of hyper-hyperparameters, information about the learning task, properties of the learning algorithm, and all of the observable performance results.

**Differing Goals: Deployment vs. Scientific Knowledge** Running $H$ is a crucial part of model development, regardless of whether one is producing a model to deploy in the real world or doing ML research. However, the goals of these activities differ: ML practitioners aim to deploy the best possible model for solving a particular task, whereas the ML research community aims to discover knowledge.

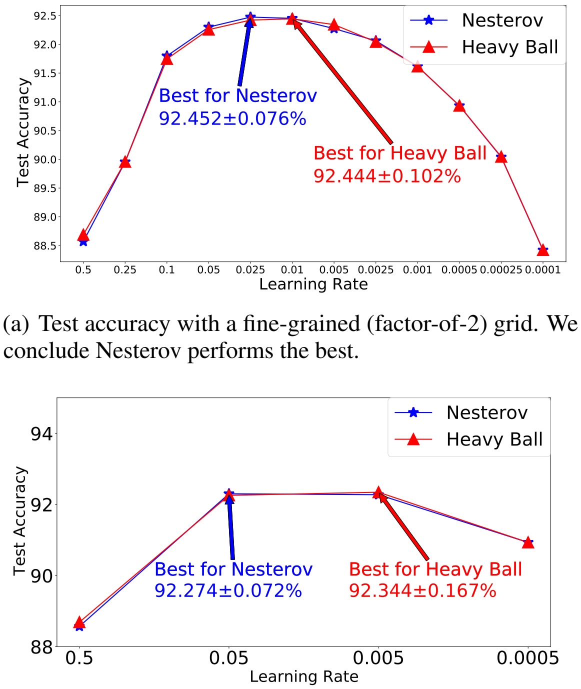
A practitioner picks a particular set of hyper-hyperparameters and runs $H$ for *model selection*: The $\mathcal{M}_{\lambda^*}$ with parameters $\theta^*$ learned via $\mathcal{A}_{\lambda^*}$ is the one that performs the best (in terms of minimizing $\mathcal{L}_{HPO}$), and is deployed as the chosen model to solve the learning task at hand. It is well known and supported empirically that the $\lambda^*$ that yield the selected model greatly affect model generalization error and thus overall ML performance (Bishop, 1995; Feurer & Hutter, 2019).

In contrast, a researcher executes $H$ as part of an empirical, scientific procedure to try to make more general conclusions about overall algorithm performance. They specify different training algorithms and a learning task, run potentially many HPO passes, and try to develop knowledge regarding whether one of the algorithms outperforms the others. Despite these different aims, most ML research work has followed a procedure similar to what ML practitioners do. A researcher runs HPO (perhaps a few times) for the algorithm under evaluation, until they achieve and can report results (and logs) that align with the argument they want to make about the algorithm's performance. This process is rather ad-hoc and does not necessarily yield reliable knowledge about the algorithm's performance more generally.

Our goal is to study HPO in this scientific-knowledge sense: We want to develop ways to reason about how we derive knowledge from empirical investigations involving HPO.

## 3. Illustrating Deception: Intuition

Studying grid search demonstrates the need for bringing rigor to how we draw conclusions from HPO. How we set the hyper-hyperparameters to determine the search grid can directly impact our conclusions. Using different grids makes it possible to draw contradictory conclusions about algorithm performance—an observation that informs our formalization for reasoning about hyperparameter deception in Section 5. Before introducing this formalization, we first explain by example the intuition behind deception.



(a) Test accuracy with a fine-grained (factor-of-2) grid. We conclude Nesterov performs the best.

(b) Test accuracy with a coarse-grained (factor-of-10) grid. We conclude that heavy ball performs the best.

*Figure 1.* Comparing test accuracy of Nesterov acceleration and heavy ball in logistic regression on MNIST for different learning rates, on a fine-grained grid (a) and coarse-grained grid (b). Each point contains an error bar within $0.2\%$.

While methods like GD and SGD have been used for decades, and are guaranteed to converge to the global minimum on convex learning problems, developing new methods for solving convex problems at scale is an active area of research. It is common to develop a new algorithm and then compare it against a baseline: The two methods are trained for the same training budget and then compared to see which has superior performance.

We compare two commonly-used momentum-based SGD variants: SGD with Nesterov acceleration (Liu & Belkin, 2018) and SGD with heavy ball momentum (Gadat et al., 2018) for logistic regression on MNIST (Figure 1). These algorithms are of a similar flavor; they both modify the update of SGD with a momentum term, so it is natural to compare their performance. We apply grid search on the learning rate and compare the final test accuracy. How we set the hyper-hyperparameter for grid spacing can lead to

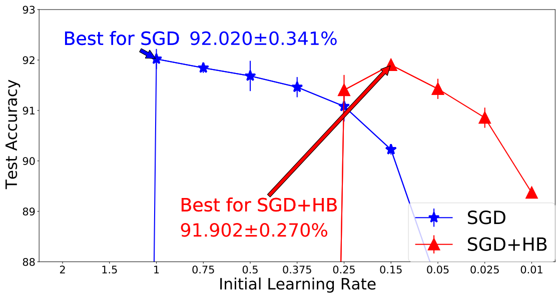


*Figure 2.* Demonstrating *hyperparameter deception* in Wilson et al. (2017)'s VGG16 experiment. Following their original experiments, we obtain these results by using 5 different seeds and averaging. Each point contains an error bar within 0.3%

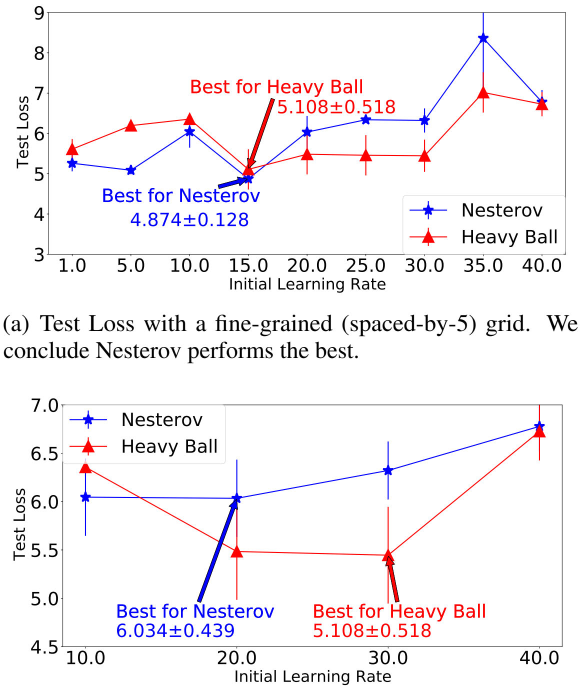
logically inconsistent conclusions. Figure 1a, which uses a finer-grained powers-of-2 grid, leads to the conclusion that Nesterov performs best, while the coarser powers-of-10 grid in 1b leads to the contradictory conclusion that heavy ball gives superior performance (See Appendix). In short, we can be deceived into concluding the wrong method performs better based on how we specify the grid.

**Illustrating deception in ML research.**

We emphasize that being inadvertently deceived by HPO is a real problem, even in excellent research; it is not limited to our toy example above. We found instances of this phenomenon in well-cited papers across multiple domains: Wilson et al. (2017), in which they compare different optimizers training VGG16 on CIFAR10, and Merity et al. (2016)'s experiments with a LSTM on Wikitext-2.

For Wilson et al. (2017), we focus on the comparison between SGD and SGD with heavy ball momentum and plot our grid search results[1] in Figure 2. The original step size $\alpha$ grid from Wilson et al. (2017) is $\{2, 1, 0.5, 0.25, 0.05, 0.01\}$ and the best test accuracy occurs for SGD at $\alpha = 1$. However, when we change the grid range for $\alpha$ to be $\{1.5, 0.75, 0.375, 0.15, 0.025\}$—notably excluding $\alpha = 1$—we get the opposite conclusion: SGD with heavy ball performs the best in terms of test accuracy, when $\alpha = 0.15$. For Merity et al. (2016), we train a tied LSTM model in a language modeling task on the Wikitext-2 dataset for 50 epochs. We observed similarly contradicting results: For a fine-grained grid we conclude that Nesterov outperforms heavy ball, while for a coarser-grained grid, we conclude the opposite.

Together these results reveal that, even in highly cited and respected experiments, it is possible to be deceived; we can draw inconsistent conclusions simply by changing the hyper-hyperparameter for grid spacing. More detailed figures can be found in the Appendix.



(a) Test Loss with a fine-grained (spaced-by-5) grid. We conclude Nesterov performs the best.

(b) Test Loss with a coarse-grained (spaced-by-10) grid. We conclude that heavy ball performs the best.

*Figure 3.* Demonstrating *hyperparameter deception* in Merity et al. (2016)'s LSTM experiment on Wikitext-2 (with 50 epochs). We compare the loss of Nesterov acceleration and heavy ball for different learning rates, on a fine-grained grid (a) and coarse-grained grid (b). We repeat each grid with 5 different random seeds.

**Related work.** Recent empirical work supports our results. While ad-hoc subsetting of the search space is a common and tacitly accepted practice among ML researchers, it can result in suboptimal performance—results that do not impart knowledge about how algorithms actually perform. Reported results tend to be impressive for some subset of the hyperparameters that the authors chose to test, but modifying HPO can lead to vastly different performance outcomes (Choi et al., 2019; Sivaprasad et al., 2020; Melis et al., 2018; Musgrave et al., 2020; Bouthillier et al., 2019). By varying the HPO procedure, it is possible to develop results that are wrong about performance, or else correct about performance but for the wrong reasons (e.g., by picking "lucky" hyperparameters). Neither of these outcomes constitutes rigorous knowledge (Gettier, 1963; Lehrer, 1979).

## 4. Epistemic Hyperparameter Optimization

Our results of Section 3 show that applying standard HPO methodologies can be deceptive: Our beliefs about algorithm performance can be controlled by happenstance, wishful thinking, or, even worse, potentially by an adversary trying to trick us with a tampered set of HPO logs. This

[1] Our final accuracy is within 0.3% of Wilson et al. (2017), due to the two experiments using different random seeds.

leaves us in a position where the "knowledge" we derived may not be knowledge at all—since we could easily, had circumstances been different, have concluded the opposite. To address this, we propose that **the process of drawing conclusions using HPO should itself be an object of study**. In this section, we formalize this sort of reasoning process, which we call Epistemic Hyperparameter Optimization (EHPO) and we provide an intuition for how EHPO can help us think about the deception problem.

**Definition 2.** *An **epistemic hyperparameter optimization procedure (EHPO)** is a tuple $(\mathcal{H}, \mathcal{F})$ where $\mathcal{H}$ is a set of HPO procedures $H$ (Definition 1) and $\mathcal{F}$ is a function that maps a set of HPO logs $\mathcal{L}$ to a set of logical sentences $\mathcal{P}$. An execution of EHPO involves running each $H \in \mathcal{H}$ some number of times (each run produces a log $\ell$) and then evaluating $\mathcal{F}$ on the set of logs produced to output the* conclusions *we draw from all of the HPO runs.*

In practice, it is most common to run EHPO for two learning algorithms, $\mathcal{J}$ and $\mathcal{K}$, and to compare their performance to conclude which is better-suited for the learning task at hand. In other words, $H$ contains at least one HPO procedure that runs $\mathcal{J}$ and one that runs $\mathcal{K}$, and the possible conclusions in the co-domain of $\mathcal{F}$ include $p$ = "$\mathcal{J}$ performs better than $\mathcal{K}$", and $\neg p$ = "$\mathcal{J}$ does not perform better than $\mathcal{K}$".

Intuitively, EHPO is deceptive whenever it could produce both $p$ and also could (if configured differently or due to randomness) produce $\neg p$. In other words, we can be deceived if the procedure we are using to derive knowledge about algorithm performance could entail logically inconsistent results. Of course, though, this intuitive definition is lacking: It is not clear what is meant by "could." Our main contribution in the sections that follow is to pin down a formal, reasonable definition of "could" in this context.

**Framing an adversary who can deceive us.** To start, we find it useful to frame the hyperparameter deception problem in terms of an adversary trying to deceive us, akin to Descartes's Evil Demon thought experiment: Imagine an evil demon who is trying to deceive us about the relative performance of different algorithms via running EHPO. At any time, the demon maintains a set $\mathcal{L}$ of HPO logs, which it can modify either by running an HPO $H \in \mathcal{H}$ with whatever hyper-hyperparameters $s \in \mathcal{S}$ it wants (producing a new log $\ell$, which it adds to $\mathcal{L}$) or by erasing some of the logs in its set. Eventually, it stops and presents us with $\mathcal{L}$, from which we will draw some conclusions using $\mathcal{F}$. The demon may be trying to deceive us via the conclusions it is possible to draw from the set of logs it produces. For example, $\mathcal{L}$ may lead us to conclude that one algorithm performs better than another, when in fact picking a different set of hyper-hyperparameters could have generated logs that would lead us to conclude differently. We want to be sure that we will not be deceived by any logs the demon "could" produce.

## 5. A Logic for Reasoning about Deception

In this section we develop axioms for EHPO to help us reason about hyperparameter deception. We find modal logic to be a useful way to express this problem (Chellas, 1980; Emerson, 1991; Garson, 2018). Modal logic inherits the tools of more-familiar propositional logic and adds two operators: $\Diamond$ to represent *possibility* and $\Box$ to represent *necessity*. These operators enable reasoning about *possible worlds*—a semantics for representing how the world *is* or *could be*, making modal logic the natural choice to express the "could" intuition from the previous section.

The well-formed formulas $\phi$ of modal logic are given recursively in Backus-Naur form by[2]

$$\phi := P \mid \neg\phi \mid \phi \wedge \phi \mid \Diamond\phi$$

where $P$ is any atomic proposition. For example, $\Diamond p$ reads, "It is possible that $p$."; $p$ is true at *some* possible world, which we *could* reach (See Appendix). The axioms of modal logic are as follows,[3] where $Q$ and $R$ are any formula:

$$\begin{aligned} &\vdash Q \rightarrow \Box Q && \textit{(necessitation)} \\ &\Box(Q \rightarrow R) \rightarrow (\Box Q \rightarrow \Box R) && \textit{(distribution).} \end{aligned}$$

As we will see, to reason about EHPO requires an extension of standard modal logic, as we need *two* modal operators: one to express the possible results of the demon running EHPO (Section 5.1) and one to express our belief (Section 5.2). The former must also be an *indexed* modal logic, where "how possible" something is is quantified by the compute capabilities of the demon. Combining these logics yields us well-formed formulas given by

$$\psi := P \mid \neg\psi \mid \psi \wedge \psi \mid \Diamond_t\psi \mid \mathcal{B}\psi$$

for any atomic proposition, $P$, and for any positive real $t$ (which below we assign the semantics of "time").

We next give semantics and axioms for this logic, culminating in a combined logic that lets us reason about whether or not we can be deceived by EHPO (Section 5.3). Our semantics are defined using sets of logs $\mathcal{L}$ as models over which a logical sentence $p$ can either be true or not. We use $\mathcal{L} \models p$, read "$\mathcal{L}$ models $p$" to mean that the sentence $p$ is true for the set of logs $\mathcal{L}$. As a starting point, we suppose that an EHPO user has in mind some atomic propositions (propositions of the background logic unrelated to possibility or belief, such as "$\mathcal{J}$ performs better than $\mathcal{K}$" or "the best-performing log for $\mathcal{J}$ has lower loss than the best-performing log for $\mathcal{K}$") with semantics that are already defined. We also inherit the usual semantics of $\wedge$ ("and") and $\neg$ ("not") from ordinary

[2] Note $\Box$ is syntactic sugar, with $\Box p \equiv \neg\Diamond\neg p$. Similarly, "or" has $p \vee q \equiv \neg(\neg p \wedge \neg q)$ and "implies" has $p \rightarrow q \equiv \neg p \vee q$.

[3] Here, "$\vdash Q$" means that $Q$ is a theorem of propositional logic.

propositional logic. In what follows, we show how we can use this to construct semantics for our modal operators of possibility and belief.

### 5.1. Expressing the possible outcomes of EHPO

Our formalization of possibility is based on the demon of Section 4. Unlike Descartes, we need not concern ourselves with "supremely powerful" demons: our potential deceivers are mere mortal ML researchers—or at worst, adversarial attackers—with bounded compute resources. The notion of possibility we define here gives limits on what possible world a demon *with bounded EHPO time* could reliably bring about. We first define a notion of a *strategy* the demon of Section 4 can execute for EHPO:

**Definition 3.** *Let $\Sigma$ denote the set of randomized* ***strategies*** *for the demon. Each $\sigma \in \Sigma$ is a function that specifies which action the demon will take: Given its current set of logs $\mathcal{L}$, either 1) running a new $H$ with hyper-hyperparameters $s$ (for which the demon gets a new, randomly-generated seed) 2) erasing some logs, or 3) returning.*

We can now define what the demon can reliably bring about, in terms of executing a strategy in bounded time:

**Definition 4.** *We let $\sigma[\mathcal{L}]$ denote the outputs of $\sigma$ running, starting from $\mathcal{L}$ (i.e., the demon is given the logs in $\mathcal{L}$ to start, then gets to run a strategy $\sigma$). Let $\tau_\sigma(\mathcal{L})$ denote the total time taken to run strategy $\sigma$; this is equivalent to the sum of the times $T$ it takes each HPO procedure $H \in \mathcal{H}$ used in the demon's strategy to run. Note that since $\sigma$ is a randomized strategy (and HPO runs $H$ are randomized as well), both $\sigma[\mathcal{L}]$ and $\tau_\sigma(\mathcal{L})$ are random variables.*

*For any formula $p$, we say $\mathcal{L} \models \Diamond_t p$ if and only if*

$$\exists \sigma \in \Sigma, \; \mathbb{P}(\sigma[\mathcal{L}] \models p) = 1 \;\wedge\; \mathbb{E}[\tau_\sigma(\mathcal{L})] \leq t.$$

Informally, $\Diamond_t p$ means that an adversary could adopt a strategy $\sigma$ that is guaranteed to cause the desired outcome $p$ to be the case while taking time at most $t$ in expectation. We will usually choose $t$ to be an upper bound on what is considered a reasonable amount of time to run HPO. In this case, any practical adversary cannot consistently bring about $p$ unless $\Diamond_t p$. It does not make sense for $t$ to be unbounded, since this would correspond to the unrealistic setting of having infinite compute time to perform HPO. We have chosen to model our budget in terms of time; however, it is worth noting we could use this setup to reason about other resource budgets, such as energy usage.

Our indexed modal logic inherits many axioms of modal logic, with indexes added (See Appendix), e.g.:

$$\begin{aligned} \vdash (p \to q) \to (\Diamond_t p \to \Diamond_t q) \quad & \textit{(necess. + distribution)} \\ p \to \Diamond_t p \quad & \textit{(reflexivity)} \\ \Diamond_t \Diamond_s p \to \Diamond_{t+s} p \quad & \textit{(transitivity)} \\ \Diamond_s \Box_t p \to \Box_t p \quad & \textit{(modal axiom 5)}, \end{aligned}$$

To summarize, the demon has knowledge of all possible hyper-hyperparameters, and it can pick whichever ones it wants to run EHPO within a bounded time budget $t$ to realize the outcomes it wants: $\Diamond_t p$ means it can realize $p$.

### 5.2. Expressing how we draw conclusions

We employ the modal operator $\mathcal{B}$ from the logic of belief[4] to model ourselves as an observer who believes in the truth of the conclusions drawn from running EHPO. $\mathcal{B}$ in this modal logic is syntactically analogous to the $\Box$ modal operator (See Appendix). We model ourselves as a consistent *Type 1* reasoner (Smullyan, 1986; Barcan-Marcus, 1995): i.e. for any formulas $p$, $q$,

$$\begin{aligned} \vdash p \to \mathcal{B}p \quad & \textit{(necessitation)} \\ \mathcal{B}(p \to q) \to (\mathcal{B}p \to \mathcal{B}q) \quad & \textit{(distribution)} \\ \neg(\mathcal{B}p \wedge \mathcal{B}\neg p) \quad & \textit{(consistency)}, \end{aligned}$$

where $\mathcal{B}p$ reads "It is concluded that $p$." For example, when comparing the performance of two algorithms, $\mathcal{B}p$ can be understood as, "It is concluded that $\mathcal{J}$ is better than $\mathcal{K}$." Our axioms ensure that we believe all propositional tautologies, our belief distributes over implication, and we do not derive contradictions. Note that we do not require completeness: We allow for the possibility that we do not conclude anything about $p$ (i.e., neither $\mathcal{B}p$ nor $\mathcal{B}\neg p$).

The semantics for our belief are straightforward to define. In the context of EHPO, in which our conclusions are based on function $\mathcal{F}$, we say that a set of logs $\mathcal{L}$ models a formula $\mathcal{B}p$ when our set of conclusions $\mathcal{F}(\mathcal{L})$ contains $p$, i.e.,

$$\mathcal{L} \models \mathcal{B}p \;\equiv\; p \in \mathcal{F}(\mathcal{L}).$$

Note that our axioms here constrain what $\mathcal{F}$ can output: for it to be a reasonable account of belief, the semantics given by $\mathcal{F}$ must model the reasoner axioms above (otherwise, deception aside, $\mathcal{F}$ is an unreasonable way for us to draw conclusions, since it is not even compatible with logic).

### 5.3. Expressing hyperparameter deception

So far in this section, we have provided an intuition for and defined formally the semantics of our two separate modal operators, $\Diamond_t$ and $\mathcal{B}$. Now, we want to show how these operators interact with each other to formally express what we informally illustrated in Section 3: a notion of hyperparameter deception.

It is a well-known result that we can combine modal logics (Scott, 1970) (See Appendix). We do so to define an axiom

[4] Technically, the belief operator from doxastic logic (Halpern et al., 2009; Rendsvig & Symons, 2019; van Benthem, 2006)

for EHPO being deception-free: for any formula $p$,

$$\neg\left(\Diamond_t \mathcal{B} p \ \wedge\ \Diamond_t \mathcal{B} \neg p\right) \qquad \textit{(t-non-deceptive).}$$

Informally, our $t$-non-deceptiveness axiom expresses the following: If there exists a strategy by which the demon could get us to conclude $p$ in $t$ expected time, then there can exist no $t$-time strategy by which the demon could have gotten us to believe $\neg p$.

Given a reasonable maximum time budget $t$, we say that EHPO is non-deceptive if it satisfies all of axioms above. Moreover, based on this notion of non-deceptiveness, we can express what it means to have a defense to being deceived. Intuitively, if there is no adversary that can consistently control whether we believe algorithm $\mathcal{J}$ is better than $\mathcal{K}$ or its negation (and $p$ or $\neg p$ more generally), then we are defended against deception. Otherwise, the EHPO procedure is potentially gameable: an adversary can consistently control our conclusions. If our $t$-non-deceptive axiom does not hold for some $p$, then even if we conclude $p$ after running EHPO, we cannot claim to *know* $p$—as our belief as to the truth-value of $p$ could be under the complete control of an adversary.

## 6. Defending against Deception

We now exercise our logical formulation from Section 5 to demonstrate that it is not trivial: It is sufficiently expressive to show when deception occurs and to reason about mechanisms of defense against deception. First, we complement our empirical results from Section 3, using our formalization to show that grid search can cause deception. Then we offer a defense involving random search.

**Grid search can cause deception.** We return to our empirical demonstration of hyperparameter deception in Section 3, and provide an intuition for characterizing what we observe in terms of the demon using a strategy $\sigma$ to deceive us about the conclusions of EHPO (See Appendix for more formal results). We run EHPO twice, using two strategies $\sigma_1$ and $\sigma_2$. For $\sigma_1$, there is one HPO procedure $H \in \mathcal{H}$, which is grid search with a powers-of-two grid (Figure 1a) to produce one log $\ell \in L_1$. The total time to run $\sigma_1$, $\tau_{\sigma_1}(\mathcal{L}_1)$, was $\sim$2 hours and 20 minutes. We have a similar setup for $\sigma_2$, using a coarser powers-of-ten grid (Figure 1b), where $\tau_{\sigma_2}(\mathcal{L}_2)$ was $\sim$1 hour and 10 minutes, which we deem to be reasonable HPO time budgets. We denote $p$ to be "Training LR with Nesterov performs better than with heavy ball on MNIST." $\mathcal{F}$, which maps from logs to conclusions, can be as naive as checking which algorithm yields the best overall test accuracy. For this example, we additionally test for statistical significance. Based on the results of running $\sigma_1$, we conclude $p$ (Figure 1a); for $\sigma_2$ we conclude $\neg p$ (Figure 1b). This violates the $t$-non-deceptive axiom. In other words, when using grid search in EHPO with different grids, we could conclude the inconsistent results $p$ and $\neg p$ within a reasonable time budget.

**A defense using random search.** Suppose that we have been drawing conclusions using some "naive" belief operator $\mathcal{B}_{\text{naive}}$ (based on a conclusion function $\mathcal{F}_{\text{naive}}$) that satisfies the axioms of Section 5.2, and we want to use it to construct a new operator $\mathcal{B}_*$ that is guaranteed to be deception-free. One straightforward way to do this is to define the belief operator $\mathcal{B}_*$ such that for any statement $p$,

$$\mathcal{B}_* p \ \equiv\ \mathcal{B}_{\text{naive}} p \ \wedge\ \neg \Diamond_t \mathcal{B}_{\text{naive}} \neg p.$$

That is, we conclude $p$ only if both our naive reasoner would have concluded $p$, and it is impossible for an adversary to get it to conclude $\neg p$ in time $t$. This enables us to show $t$-non-deceptiveness, following directly from the axioms:

$$\begin{aligned} \Diamond_t \mathcal{B}_* p &\equiv \Diamond_t \left(\mathcal{B}_{\text{naive}} p \ \wedge\ \neg \Diamond_t \mathcal{B}_{\text{naive}} \neg p\right) \\ &\to \Diamond_t \left(\neg \Diamond_t \mathcal{B}_{\text{naive}} \neg p\right) && \text{(by nec. + dist.)} \\ &\to \neg \Diamond_t \mathcal{B}_{\text{naive}} \neg p && \text{(by modal ax. 5),} \end{aligned}$$

$$\begin{aligned} \Diamond_t \mathcal{B}_* \neg p &\equiv \Diamond_t \left(\mathcal{B}_{\text{naive}} \neg p \ \wedge\ \neg \Diamond_t \mathcal{B}_{\text{naive}} p\right) \\ &\to \Diamond_t \mathcal{B}_{\text{naive}} \neg p && \text{(by nec. + dist.),} \end{aligned}$$

which immediately lets us derive the $t$-non-deceptive axiom by contradiction. This derivation illustrates the power of our logical formulation: We can validate defenses against deception in only a few lines, without needing to refer to the underlying semantics of EHPO. While this analysis shows that some sort of defense is always possible, it may not be practical to compute $\mathcal{B}_*$ as defined here because we cannot easily evaluate whether $\Diamond_t \mathcal{B}_{\text{naive}} \neg p$.

We now illustrate that it is possible to implement a defense by constructing a concrete EHPO that satisfies our axioms for a particular HPO setup. We use *random search*, popularized by Bergstra & Bengio (2012), as the underlying HPO procedure. Random search takes two hyper-hyperparameters, a distribution $\mu$ over the hyperparameter space and a number of trials $K$ to run. Running HPO consists of $K$ independent trials of the learning algorithm $\mathcal{A}_{\lambda_1}, \mathcal{A}_{\lambda_2}, \ldots, \mathcal{A}_{\lambda_K}$, where the hyperparameters $\lambda_k$ are independently drawn from $\mu$, taking expected time proportional to $K$. For simplicity, we will suppose there is only one algorithm under consideration, $\mathcal{A}$. We suppose that the set of allowable hyper-hyperparameters (and in turn the set of allowable hyperparameters) is constrained, such that any two allowable random-search distributions $\mu$ and $\nu$ have Renyi-$\infty$-divergence at most a constant $D_\infty(\mu \| \nu) \leq \gamma$.

We start with a naive reasoner $\mathcal{B}_{\text{naive}}$, which draws conclusions from only a single log containing $K$ trials. Our goal is to construct a $\mathcal{B}_*$ that has a "defended reasoner" $\mathcal{F}$. This $\mathcal{F}$ should weaken the conclusions of $\mathcal{F}_{\text{naive}}$ (i.e., $\mathcal{F}(\mathcal{L}) \subseteq \mathcal{F}_{\text{naive}}(\mathcal{L})$ for any $\mathcal{L}$) and be guaranteed to be $t$-non-deceptive. The $\mathcal{B}_*$ we construct similarly draws conclusions

*Table 1.* A defense for Logistic Regression (LR) on MNIST. $p_{a,b}$ denotes the proposition $\mathcal{O}_a > \mathcal{O}_b$, i.e. *Training LR with optimizer $\mathcal{O}_a$ generalizes better than with optimizer $\mathcal{O}_b$ on MNIST.* We similarly define $p_{b,a}$. We adopt $\epsilon = 0.1$ for drawing conclusions.

| Comparison | $p_{a,b}$ | $p_{b,a}$ | Conclude |
|---|---|---|---|
| Nes vs. HB | Nes > HB<br>47.52% | Nes < HB<br>52.48% | None |
| Adam vs. HB | Adam > HB<br>0.83% | Adam < HB<br>99.17% | $p_{b,a}$ |
| Nes vs. Adam | Nes > Adam<br>94.91% | Nes < Adam<br>5.09% | $p_{a,b}$ |

from a single log, but does so from a log containing $KR$ trials for some fixed $R \in \mathbb{N}$. It evaluates a conclusion $p$ by dividing the log's $KR$ trials into $R$ groups of $K$ trials, evaluating $\mathcal{B}_{\text{naive}}p$ on each group, and concluding $p$ only if $\mathcal{B}_{\text{naive}}$ also concluded $p$ on all $R$ groups. In the Appendix, we prove that $\mathcal{B}_*$ will be $t$-non-deceptive if $R$ is set to be $R = \sqrt{t \exp(\gamma K)/K} = O(\sqrt{t})$. This result both validates the defense and does so with a log size—and compute requirement for good-faith EHPO—that is sublinear in $t$.

**Illustrating a defense empirically.** Now we re-run the experiment from Section 3, using a slightly modified defense: Instead of requiring hits on all $R$ independent groups of trials, we use subsamples of all available trials and require hits on at least a $1 - \epsilon$ fraction of the subsamples.[5] We adopt three candidate optimizers: (1) SGD with Nesterov acceleration (Nes), (2) SGD with Heavy Ball method (HB), and (3) Adam (Kingma & Ba, 2014). The defended reasoner runs random search with 125 trials. We then take 10000 subsamples of size $K = 11$, requiring at least an at least $1 - \epsilon$ fraction of hits. We pass them to the naive reasoner, which makes conclusions based on which algorithm performed best across $K$ trials. We summarize the results in Table 1 and include an algorithm statement in the Appendix.

## 7. Related Work: Toward More Robust ML

We consider our work on formalizing hyperparameter deception to be orthogonal to, but just as urgent as, recent advocacy for better reproducibility in ML research (Henderson et al., 2018; Pineau, 2019; Gundersen & Kjensmo, 2018; Raff, 2019; Bouthillier et al., 2019). Reproducibility is only part of the story for ensuring robustness; it is necessary, but not sufficient. While reproducibility guards against brittle findings through the "good science" practice of replicable results, it does not guarantee that those results are actually correct—that they are reflective of actual knowledge of an algorithm's performance. We address this need, providing a mechanism for reasoning more rigorously about algorithm performance in the context of HPO.

More generally, our work can be understood as a mechanism for dealing with *measurement bias*—the misalignment between what one intends to measure and what they are actually measuring—for overall ML algorithm performance. While alleviating measurement bias is by no means novel to more mature branches of science (Gould, 1981), including other fields of computing (Mytkowicz et al., 2009), it is coming under increased scrutiny with respect to the origins of empirical gains in ML (Musgrave et al., 2020). In current work, it is often difficult to disentangle whether the concluded measured performance gains are due to properties of the training algorithm or to fortuitous hyperparameter selection. Our formalization, rather than allowing hyper-hyperparameter choices to potentially obscure empirical results, provides confidence in the conclusions we can draw about overall algorithm performance.

Our work also highlights how there is a human element, not a just statistical one, to bias in ML pipelines: Practitioners make decisions about hyper-hyperparameters that can heavily influence performance. The human element of biasing solution spaces has been discussed in sociotechnical writing (Friedman & Nissenbaum, 1996), in AI (Mitchell, 1980), in the context of "p-hacking" results that fit a desired pattern (Gelman & Loken, 2019), and, more recently, was also the focus of Isbell (2020)'s NeurIPS keynote. Our push here to formalize conclusions from HPO has the potential to alleviate the effects of such bias.

## 8. Conclusion

After confirming empirically that grid search can lead to inconsistent conclusions about algorithm performance, we focus our attention on how to reason formally about the conclusions we can draw from HPO. We suggest a logic for doing so, and motivate it via the example of an adversary who is trying to deceive us from running an epistemic HPO procedure—a procedure in which such an adversary can run any number of reproducible HPO passes to try to get us to believe a particular notion about comparative algorithm performance. We verify that this logic enables us to formally capture that grid search can deceive us: Modifying the hyper-hyperparameters can easily mislead conclusions about algorithm performance. We offer a defense against deception, analytically showing that random search can be employed to prevent inconsistent conclusions.

More generally, our work is a call to ML scientists to reason more rigorously about their beliefs concerning algorithm performance. In relation to EHPO, this is akin to challenging researchers to reify their notion of $\mathcal{B}$—to justify their belief in the conclusions they make from the HPO they run

[5] Practically speaking, it is easier to subsample than resample.

in their experiments. We believe that doing so will contribute significantly to the ongoing effort of producing a more robust ML scientific discipline.

## Acknowledgements

This work was made possible by generous funding from a Cornell Institute for the Social Sciences (ISS) Team Grant. We would additionally like to thank the following individuals for feedback on earlier ideas, drafts, and iterations of this work: Prof. Rediet Abebe, Harry Auster, Prof. Solon Barocas, Jerry Chee, Jessica Zosa Forde, Dr. Jack Goetz, Kweku Kwegyir-Aggrey, Prof. Karen Levy, and Prof. Helen Nissenbaum. We would also like to thank the Artificial Intelligance Policy & Practice initiative at Cornell University.

## References

Barcan-Marcus, R. *Modalities: Philosophical Essays*. Oxford University Press, 1995.

Bergstra, J. and Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.*, 13: 281–305, February 2012. ISSN 1532-4435.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for Hyper-Parameter Optimization. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 2546–2554. Curran Associates, Inc., 2011.

Bishop, C. M. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., USA, 1995. ISBN 0198538642.

Bouthillier, X., Laurent, C., and Vincent, P. Unreproducible Research is Reproducible. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 725–734. PMLR, 09–15 Jun 2019.

Chellas, B. F. *Modal Logic - An Introduction*. Cambridge University Press, 1980.

Chen, I., Johansson, F. D., and Sontag, D. Why Is My Classifier Discriminatory? In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31, pp. 3539–3550. Curran Associates, Inc., 2018.

Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., and Dahl, G. E. On Empirical Comparisons of Optimizers for Deep Learning, 2019.

Claesen, M. and Moor, B. D. Hyperparameter Search in Machine Learning, 2015.

Descartes, R. *Discourse on Method and Meditations on First Philosophy*. Hackett Publishing Company, Inc., Translator Donald A. Cress, 4th edition, 1998. Meditation One: Concerning Those Things That Can Be Called into Doubt.

Emerson, E. A. *Temporal and Modal Logic*, pp. 995–1072. MIT Press, Cambridge, MA, USA, 1991. ISBN 0444880747.

Feurer, M. and Hutter, F. Hyperparameter Optimization. In Hutter, F., Kotthoff, L., and Vanschoren, J. (eds.), *Automated Machine Learning: Methods, Systems, Challenges*, pp. 3–33. Springer International Publishing, 2019.

Friedman, B. and Nissenbaum, H. Bias in Computer Systems. *ACM Trans. Inf. Syst.*, 14(3):330–347, July 1996. ISSN 1046-8188.

Gadat, S., Panloup, F., Saadane, S., et al. Stochastic heavy ball. *Electronic Journal of Statistics*, 12(1):461–529, 2018.

Garson, J. Modal Logic. In *The Stanford Encyclopedia of Philosophy*. Fall 2018 Edition, Edward N. Zalta (ed.), 2018.

Gelman, A. and Loken, E. The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking" and the research hypothesis was posited ahead of time, 2019.

Gettier, E. L. Is Justified True Belief Knowledge? *Analysis*, 23(6):121–123, 06 1963.

Gieryn, T. F. Boundary-Work and the Demarcation of Science from Non-Science: Strains and Interests in Professional Ideologies of Scientists. *American Sociological Review*, 48(6):781–795, 1983.

Gould, S. J. *The Mismeasure of Man*. Norton, New York, 1981.

Gundersen, O. E. and Kjensmo, S. State of the Art: Reproducibility in Artificial Intelligence. In *AAAI*, 2018.

Halpern, J. Y., Samet, D., and Segev, E. Defining Knowledge in Terms of Belief: The Modal Logic Perspective. *Rev. Symb. Log.*, 2(3):469–487, 2009.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep Reinforcement Learning that Matters. In *Thirty-Second AAAI Conference On Artificial Intelligence*, 2018.

Hinton, G. E. A Practical Guide to Training Restricted Boltzmann Machines. In Montavon, G., Orr, G. B., and Müller, K.-R. (eds.), *Neural Networks: Tricks of the Trade: Second Edition*, pp. 599–619. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

Hsu, C., Chang, C., and Lin, C. A Practical Guide to Support Vector Classification, November 2003.

Isbell, C. You Can't Escape Hyperparameters and Latent Variables: Machine Learning as a Software Engineering Enterprise. NeurIPS Keynote, 2020.

John, G. H. Cross-Validated C4.5: Using Error Estimation for Automatic Parameter Selection. Technical report, Stanford University, Stanford, CA, USA, 1994.

Kingma, D. and Ba, J. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 12 2014.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. An Empirical Evaluation of Deep Architectures on Problems with Many factors of Variation. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 473–480, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient BackProp. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pp. 9–50, Berlin, Heidelberg, 1998. Springer-Verlag. ISBN 3540653112.

Lehrer, K. The Gettier Problem and the Analysis of Knowledge. In Pappas, G. S. (ed.), *Justification and Knowledge: New Studies in Epistemology*, pp. 65–78. Springer Netherlands, Dordrecht, 1979.

Lipton, Z. C. and Steinhardt, J. Troubling Trends in Machine Learning Scholarship. *ACM Queue*, 2018.

Liu, C. and Belkin, M. Accelerating sgd with momentum for over-parameterized learning. *arXiv preprint arXiv:1810.13395*, 2018.

Melis, G., Dyer, C., and Blunsom, P. On the State of the At of Evaluation in Neural Language models. In *International Conference on Learning Representations*, 2018.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Mitchell, T. M. The Need for Biases in Learning Generalizations. Technical report, Rutgers University, New Brunswick, NJ, 1980. http://www-cgi.cs.cmu.edu/~tom/pubs/NeedForBias_1980.pdf.

Musgrave, K., Belongie, S., and Lim, S.-N. A Metric Learning Reality Check, 2020.

Mytkowicz, T., Diwan, A., Hauswirth, M., and Sweeney, P. F. Producing Wrong Data without Doing Anything Obviously Wrong! In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV, pp. 265–276, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584065.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., and et al. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011. ISSN 1532-4435.

Pineau, J. The Machine Learning Reproducibility Checklist, March 2019. https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf.

Raff, E. A Step Toward Quantifying Independently Reproducible Machine Learning Research. In *NeurIPS*, 2019.

Rendsvig, R. and Symons, J. Epistemic Logic. In *The Stanford Encyclopedia of Philosophy*. Summer 2019 Edition, Edward N. Zalta (ed.), 2019.

Schneider, F., Balles, L., and Hennig, P. DeepOBS: A Deep Learning Optimizer Benchmark Suite. In *7th International Conference on Learning Representations (ICLR)*. ICLR, May 2019.

Scott, D. Advice on modal logic. In Lambert, K. (ed.), *Philosophical Problems in Logic: Some Recent Developments*, pp. 143–173. Springer Netherlands, Dordrecht, 1970.

Sivaprasad, P. T., Mai, F., Vogels, T., Jaggi, M., and Fleuret, F. Optimizer Benchmarking Needs to Account for Hyperparameter Tuning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9036–9045. PMLR, 13–18 Jul 2020.

Smullyan, R. M. Logicians Who Reason about Themselves. In *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge*, TARK '86, pp. 341–352, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc. ISBN 0934613049.

van Benthem, J. Epistemic Logic and Epistemology: The State of Their Affairs. *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition*, 128(1):49–76, 2006.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The Marginal Value of Adaptive Gradient Methods in Machine Learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4148–4158. Curran Associates, Inc., 2017.