Cornell Bowers C·IS
College of Computing and Information Science

Toward More Robust Hyperparameter Optimization

A. Feder Cooper
Ph.D. Student, Department of Computer Science

Brief bio

Current work **Accountable ML algorithms**

Future work **Accountable ML systems**

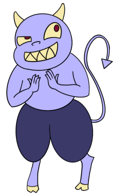
AF Cooper, et. al. [EAAMO '21]
AF Cooper and K Levy [CTLJ '22]

Reliable model selection

JZ Forde*, **AF Cooper***, et. al. [SEDL@ICLR '21]

AF Cooper and E Abrams [AIES '21]

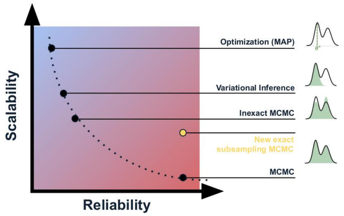
AF Cooper, et. al. [NeurIPS '21]



Reliable, scalable MCMC

R Zhang, **AF Cooper**, et. al. [AISTATS '20]

R Zhang*, **AF Cooper***, et. al. [NeurIPS '20]



Brief bio

Accountable ML algorithms

Accountable ML systems

AF Cooper, et. al. [EAAMO '21]
AF Cooper and K Levy [CTLJ '22]

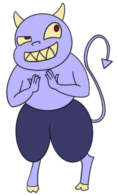
Reliable model selection

Reliable, scalable MCMC

JZ Forde*, **AF Cooper***, et. al. [SEDL@ICLR '21]

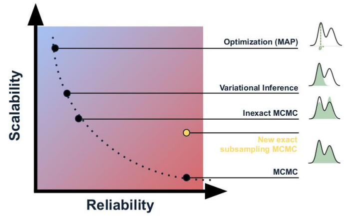
AF Cooper and E Abrams [AIES '21]

AF Cooper, et. al. [NeurIPS '21]



R Zhang, **AF Cooper**, et. al. [AISTATS '20]

R Zhang*, **AF Cooper***, et. al. [NeurIPS '20]

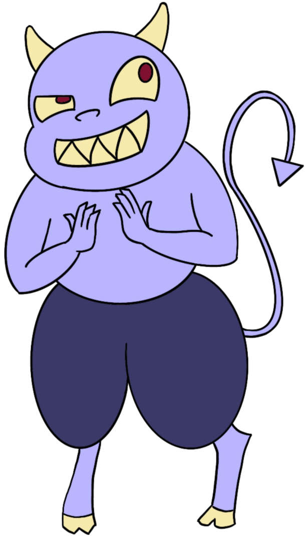


Hyperparameter Optimization Is Deceiving Us, and How to Stop It

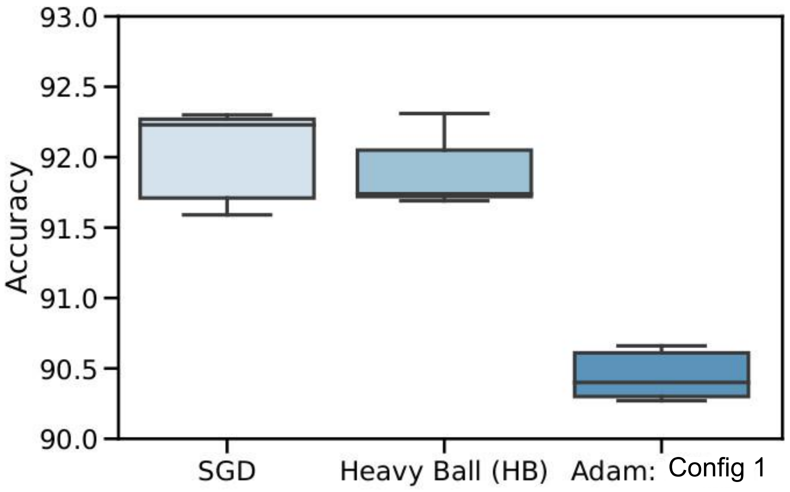
AF Cooper, et. al. [NeurIPS ‘21]

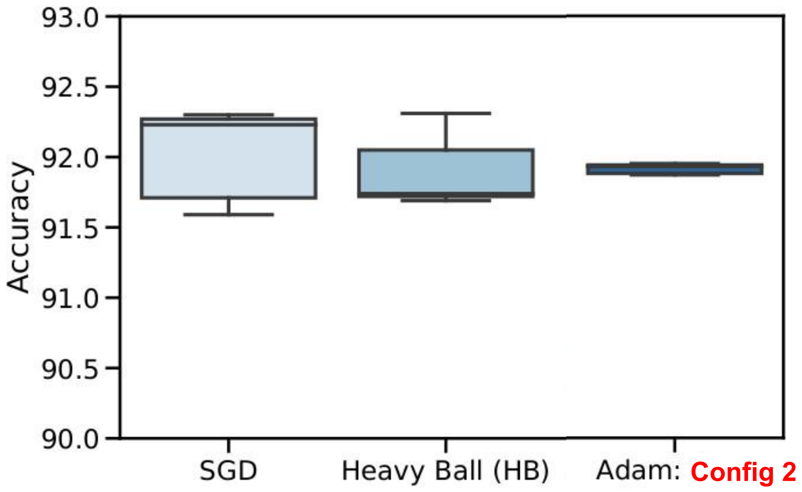
Joint work with

Yucheng Lu, **Jessica Zosa Forde**, and **Chris De Sa**

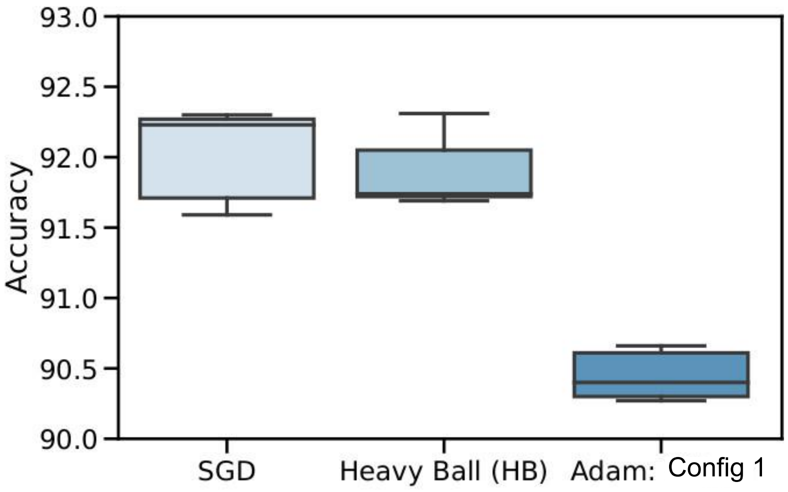


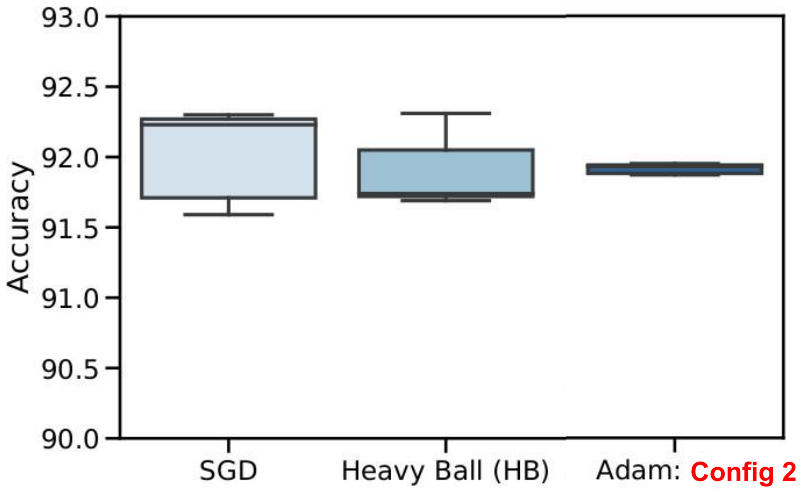
We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform





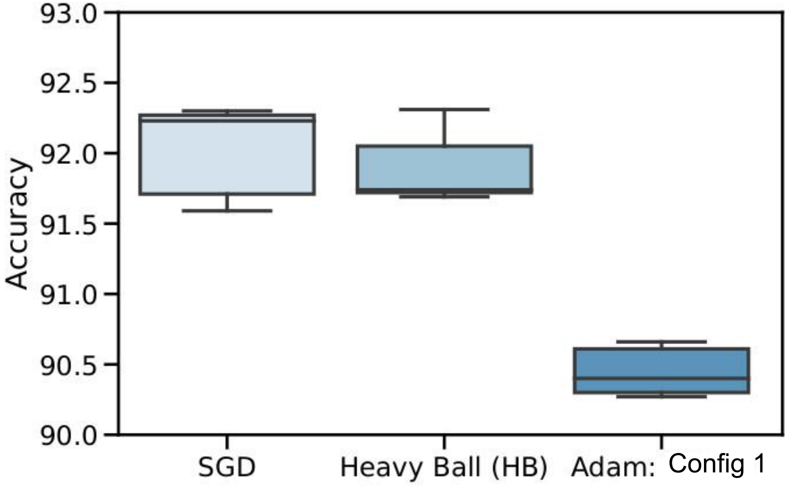
We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform

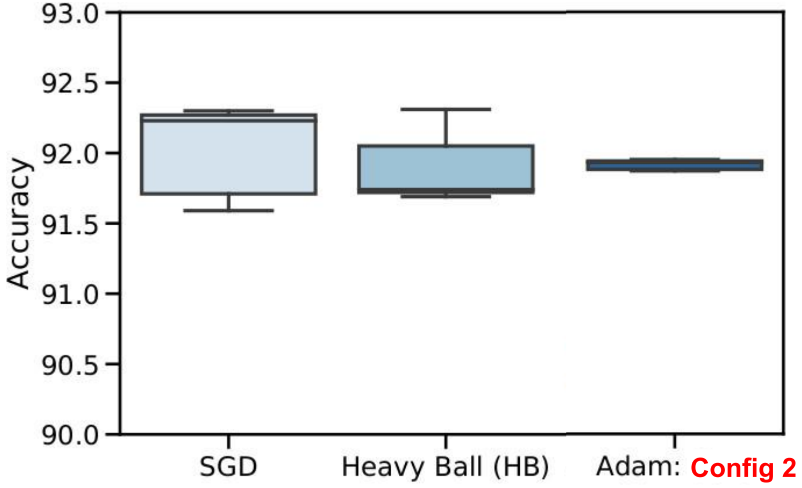




non-adaptive optimizers
outperform adaptive optimizers

We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform

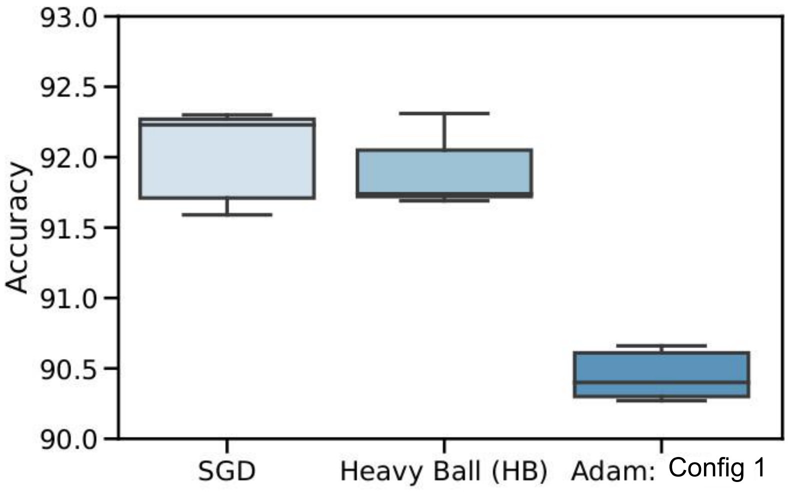




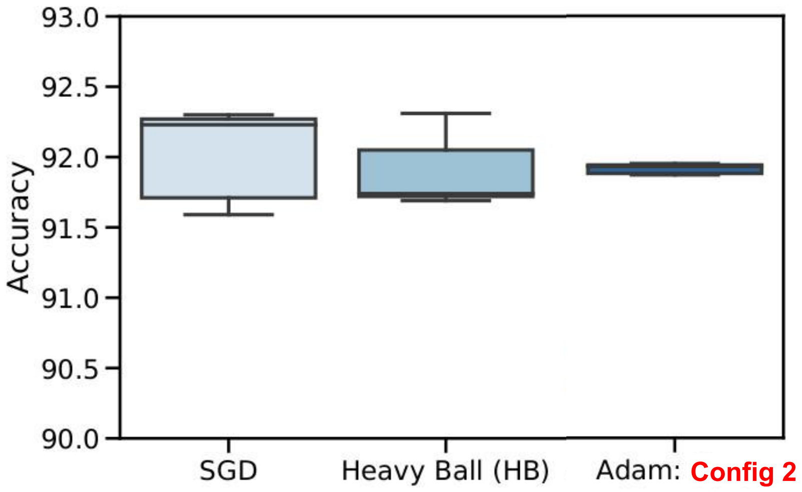
non-adaptive optimizers outperform adaptive optimizers

non-adaptive optimizers do not outperform adaptive optimizers

We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform



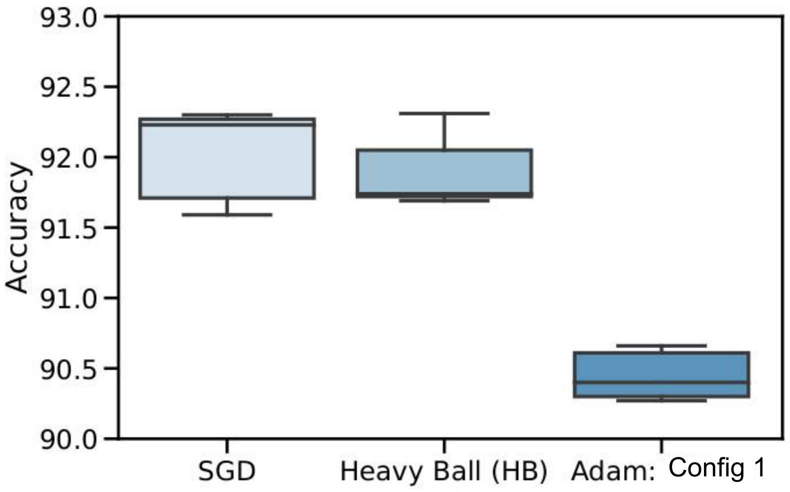
non-adaptive optimizers
outperform adaptive optimizers



non-adaptive optimizers
do not outperform adaptive optimizers

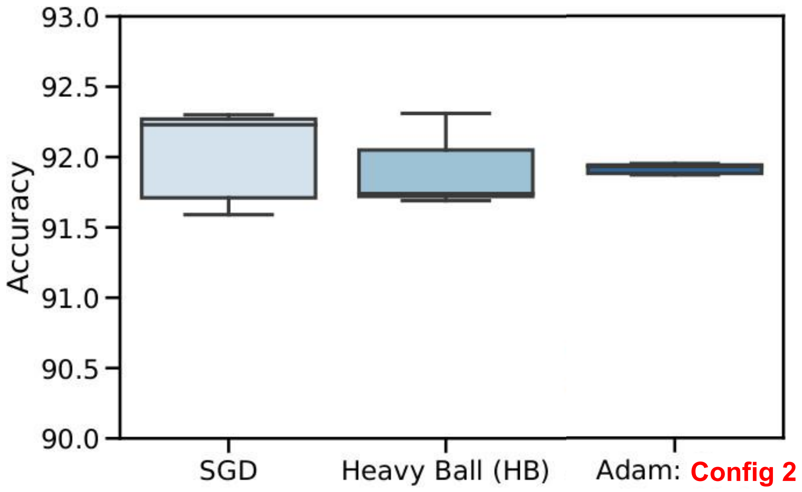
p

We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform



non-adaptive optimizers outperform adaptive optimizers

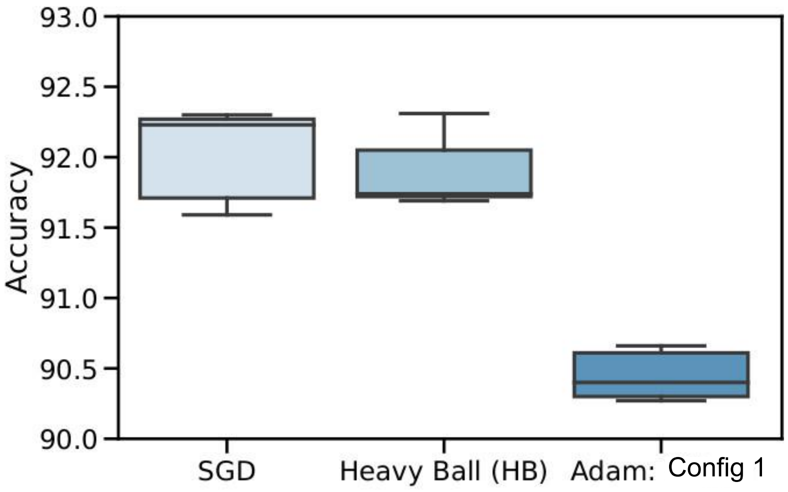
p



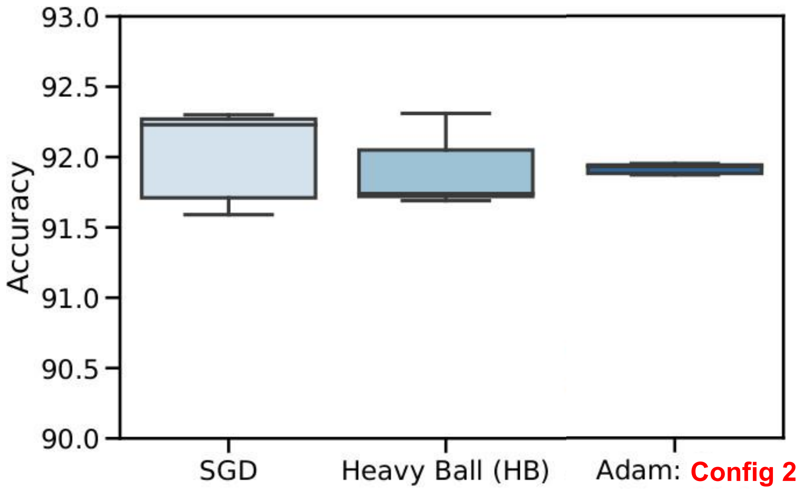
non-adaptive optimizers do not outperform adaptive optimizers

$\neg p$

We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform



non-adaptive optimizers outperform adaptive optimizers

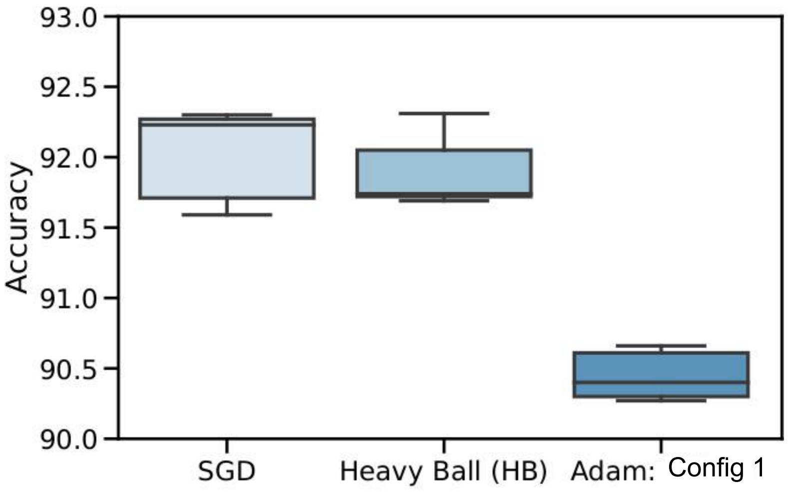


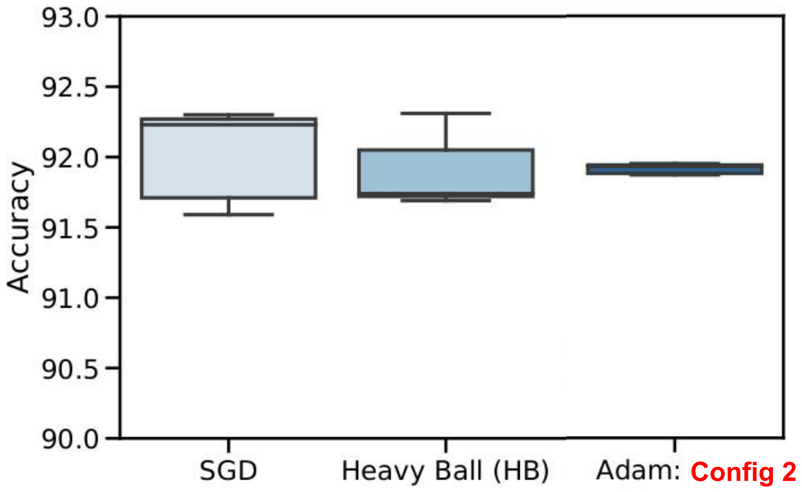
non-adaptive optimizers do not outperform adaptive optimizers

p AND $\neg p$ $\rightarrow$ FALSE

We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform

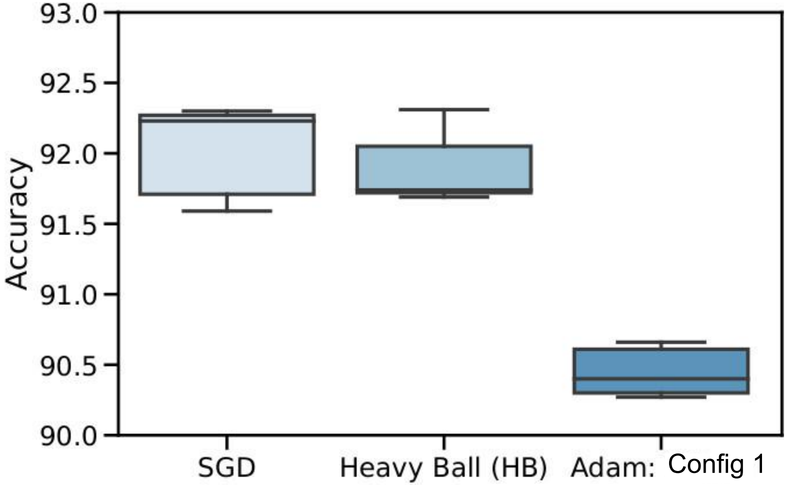
We do not know the ground truth

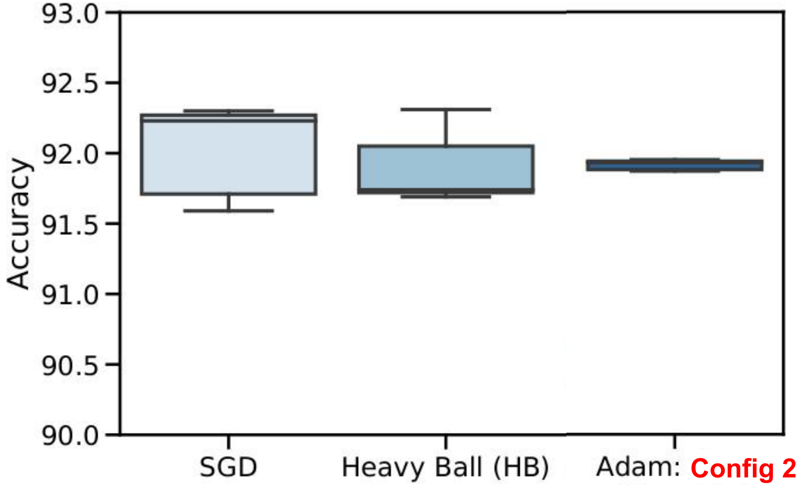




We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform

We do not know the ground truth

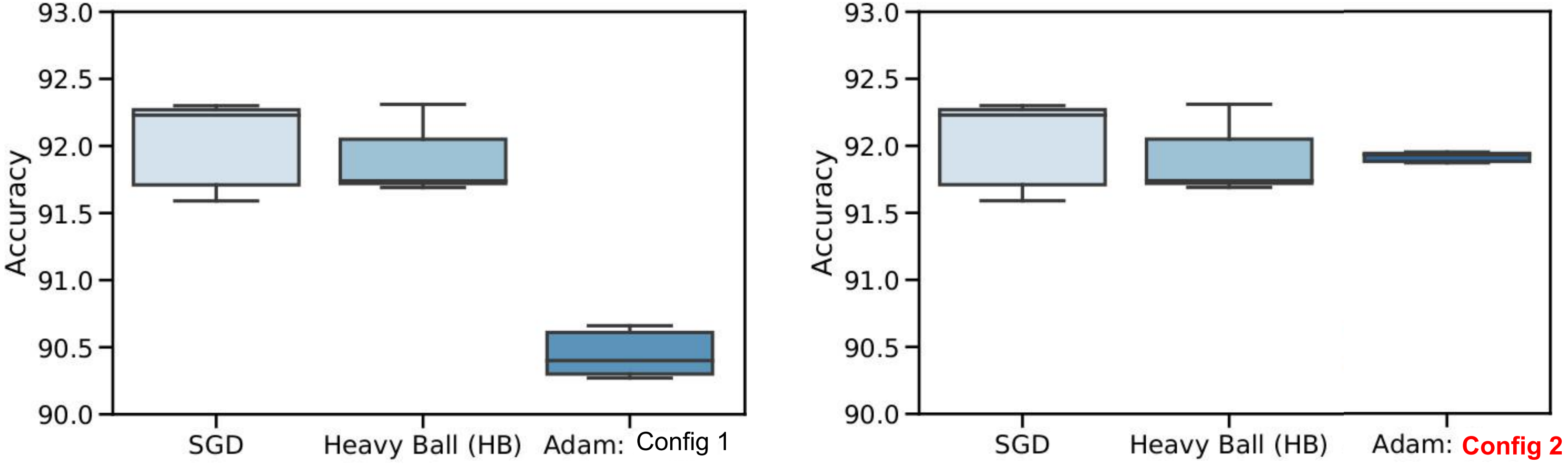




It is **fine** for the ML community to accept **either** to form **conclusions**

We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform

We do not know the ground truth

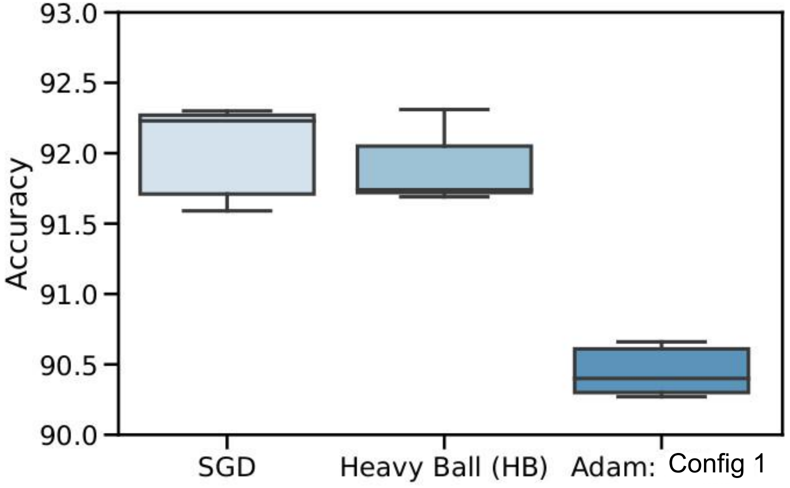


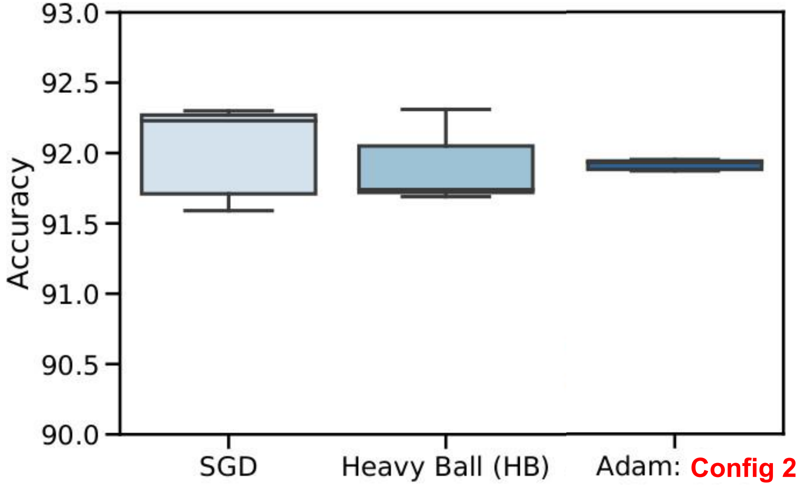
It is **fine** for the ML community to accept **either** to form **conclusions**

It is **fine** for the ML community to accept **neither** to form **no conclusions**

We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform

We do not know the ground truth





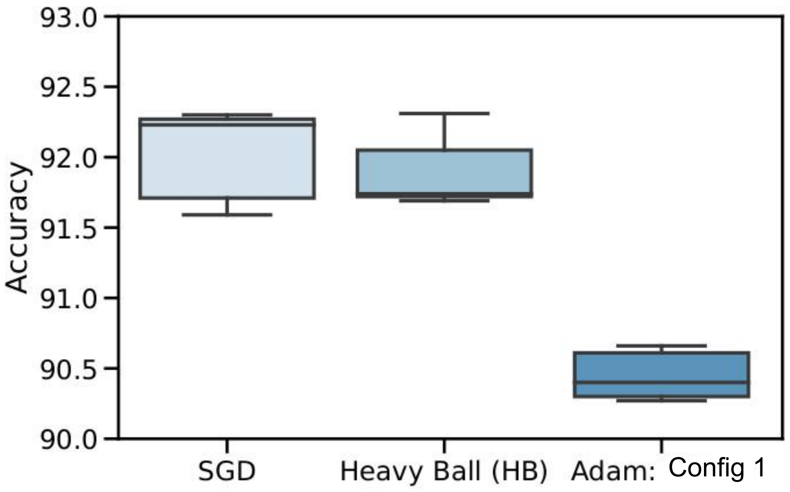
It is **fine** for the ML community to accept **either** to form **conclusions**

It is **fine** for the ML community to accept **neither** to form **no conclusions**

It is **not fine** for the ML community to accept **both** to form **inconsistent conclusions**

We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform

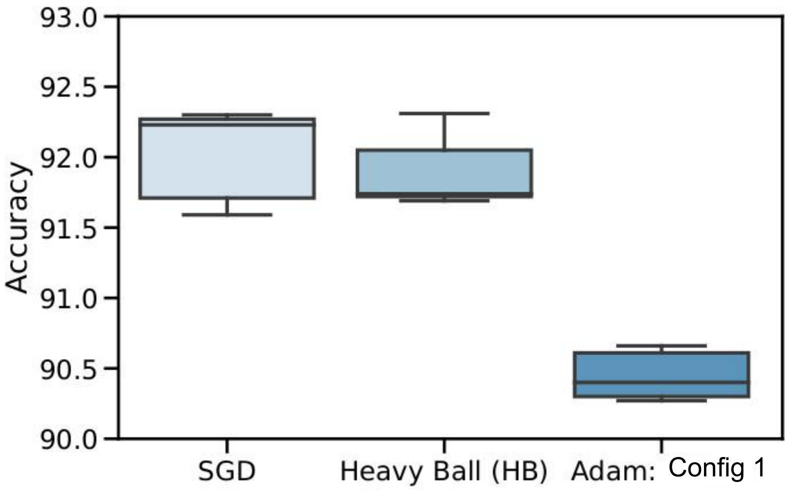
We do not know the ground truth



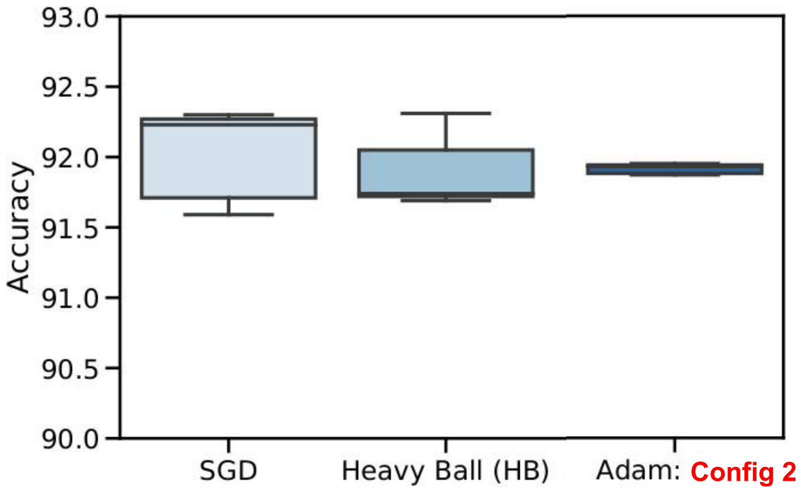
p

We want to prevent our conclusions from depending on the underlying configuration of the HPO we perform

We do not know the ground truth



p



$\neg p$

We do not want this to be *possible*

We call this phenomenon “hyperparameter deception”

No one has studied HPO at this meta level of “possible” outcomes before

For **ML practitioners**, the **goal is to deploy a specific model**

pick a few hyperparameters to test, compare results, and deploy the model that achieved the best performance

For **ML research**, the **goal is to derive general knowledge**

pick a few hyperparameters to test, compare results, and deploy the model that achieved the best performance

We call this phenomenon "hyperparameter deception"

No one has studied HPO at this meta level of "possible" outcomes before

For **ML practitioners**, the **goal is to deploy a specific model**

> **pick a few hyperparameters to test, compare results, and deploy the model that achieved the best performance**

For **ML research**, the **goal is to derive general knowledge**

> **pick a few hyperparameters to test, compare results, and deploy the model that achieved the best performance**

Doing the same thing to achieve very different goals

The process of picking HPO configurations to test is vague

Whether we believe our conclusions or not is also vague

We formally study “hyperparameter deception”

The paper formalizes the problem in 2 phases:

1) We remove the vagueness from the problem by pinning down
    a) a concrete formalization for the **possible outcomes of running hyperparameter optimization**
    b) a concrete formalization of our **belief in conclusions**

2) Proving non-trivial theorems about whether a hyperparameter optimization procedure is defended against “deception” (and validating this empirically)

We formally study “hyperparameter deception”

The paper formalizes the problem in 2 phases:

1) We remove the vagueness from the problem by pinning down
   a) a concrete formalization for the **possible outcomes of running hyperparameter optimization**
   b) a concrete formalization of our **belief in conclusions**

2) Proving non-trivial theorems about whether a hyperparameter optimization procedure is defended against “deception” (and validating this empirically)

We formalize the process of drawing conclusions from empirical studies using HPO

We formalize all of the randomness in HPO in terms of a random seed *r* and a pseudo-random number generator (PRNG) *G*. Given a seed, *G* deterministically produces a sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^{\infty}$.

We formalize the process of drawing conclusions from empirical studies using HPO

We formalize all of the randomness in HPO in terms of a random seed r and a pseudo-random number generator (PRNG) G. Given a seed, G deterministically produces a sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^{\infty}$. With this, we can now define HPO formally:

Definition 2. *An HPO procedure H is a tuple $(H_*, \mathcal{C}, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where H_* is a randomized algorithm, $\mathcal{C}$ is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), Λ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\mathcal{M}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in \mathcal{C}$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[3] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple (λ^*, ℓ), where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

We formalize the process of drawing conclusions from empirical studies using HPO

We formalize all of the randomness in HPO in terms of a random seed r and a pseudo-random number generator (PRNG) G. Given a seed, G deterministically produces a sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^\infty$. With this, we can now define HPO formally:

Definition 2. *An HPO procedure H is a tuple $(H_*, \mathcal{C}, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where $\boxed{H_*}$ is a randomized algorithm, $\mathcal{C}$ is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), Λ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\mathcal{M}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in \mathcal{C}$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[3] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple (λ^*, ℓ), where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

We formalize the process of drawing conclusions from empirical studies using HPO

We formalize all of the randomness in HPO in terms of a random seed *r* and a pseudo-random number generator (PRNG) *G*. Given a seed, *G* deterministically produces a sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^{\infty}$. With this, we can now define HPO formally:

Definition 2. *An HPO procedure H is a tuple $(H_*, \mathcal{C}, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where H_* is a randomized algorithm, $\mathcal{C}$ is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), Λ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\mathcal{M}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in \mathcal{C}$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[3] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple (λ^*, ℓ), where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

We formalize the process of drawing conclusions from empirical studies using HPO

We formalize all of the randomness in HPO in terms of a random seed *r* and a pseudo-random number generator (PRNG) G. Given a seed, G deterministically produces a sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^\infty$. With this, we can now define HPO formally:

Definition 2. *An HPO procedure H is a tuple $(H_*, \mathcal{C}, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where H_* is a randomized algorithm, $\mathcal{C}$ is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), $\boxed{\Lambda}$ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\mathcal{M}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in \mathcal{C}$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[3] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple (λ^*, ℓ), where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

We formalize the process of drawing conclusions from empirical studies using HPO

We formalize all of the randomness in HPO in terms of a random seed *r* and a pseudo-random number generator (PRNG) G. Given a seed, G deterministically produces a sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^\infty$. With this, we can now define HPO formally:

Definition 2. *An HPO procedure H is a tuple $(H_*, \mathcal{C}, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where H_* is a randomized algorithm, $\mathcal{C}$ is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), Λ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\mathcal{M}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in \mathcal{C}$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[3] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple (λ^*, ℓ), where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

We formalize the process of drawing conclusions from empirical studies using HPO

We formalize all of the randomness in HPO in terms of a random seed r and a pseudo-random number generator (PRNG) G. Given a seed, G deterministically produces a sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^\infty$. With this, we can now define HPO formally:

Definition 2. *An HPO procedure H is a tuple $(H_*, \mathcal{C}, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where H_* is a randomized algorithm, $\mathcal{C}$ is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), Λ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\boxed{\mathcal{M}}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in \mathcal{C}$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[3] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple (λ^*, ℓ), where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

We formalize the process of drawing conclusions from empirical studies using HPO

We formalize all of the randomness in HPO in terms of a random seed *r* and a pseudo-random number generator (PRNG) G. Given a seed, G deterministically produces a

sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^{\infty}$. With this, we can now define HPO formally:

Definition 2. *An HPO procedure H is a tuple $(H_*, \mathcal{C}, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where H_* is a randomized algorithm, $\mathcal{C}$ is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), Λ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\mathcal{M}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in \mathcal{C}$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[3] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple (λ^*, ℓ), where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

We formalize the process of drawing conclusions from empirical studies using HPO

We formalize all of the randomness in HPO in terms of a random seed *r* and a pseudo-random number generator (PRNG) *G*. Given a seed, *G* deterministically produces a sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^{\infty}$. With this, we can now define HPO formally:

Definition 2. *An HPO procedure H is a tuple $(H_*, \mathcal{C}, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where H_* is a randomized algorithm, $\mathcal{C}$ is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), Λ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\mathcal{M}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in \mathcal{C}$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[3] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple (λ^*, ℓ), where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

We formalize the process of drawing conclusions from empirical studies using HPO

We formalize all of the randomness in HPO in terms of a random seed *r* and a pseudo-random number generator (PRNG) *G*. Given a seed, *G* deterministically produces a

sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^{\infty}$. With this, we can now define HPO formally:

Definition 2. *An HPO procedure H is a tuple $(H_*, \mathcal{C}, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where H_* is a randomized algorithm, $\mathcal{C}$ is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), Λ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\mathcal{M}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in \mathcal{C}$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[3] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple (λ^*, ℓ), where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

We formalize the process of drawing conclusions from empirical studies using HPO

We formalize all of the randomness in HPO in terms of a random seed *r* and a pseudo-random number generator (PRNG) *G*. Given a seed, *G* deterministically produces a

sequence of pseudo-random numbers: all numbers lie in some set $\mathcal{I}$ (typically 64-bit integers), i.e. $r \in \mathcal{I}$ and PRNG $G : \mathcal{I} \to \mathcal{I}^\infty$. With this, we can now define HPO formally:

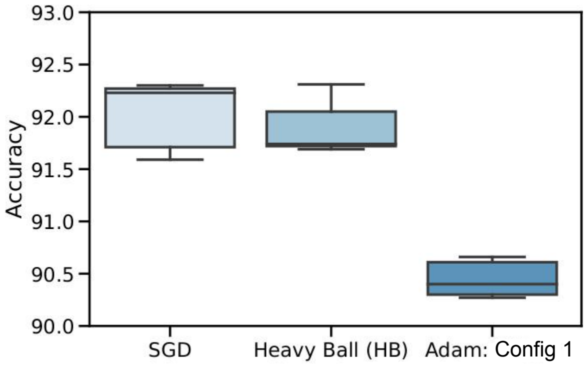
Definition 2. *An HPO procedure H is a tuple $(H_*, \mathcal{C}, \Lambda, \mathcal{A}, \mathcal{M}, G, X)$ where H_* is a randomized algorithm, $\mathcal{C}$ is a set of allowable hyper-HPs (i.e., allowable configurations for H_*), Λ is a set of allowable HPs (i.e., of HP sets λ), $\mathcal{A}$ is a training algorithm (e.g. SGD), $\mathcal{M}$ is a model (e.g. VGG16), G is a PRNG, and X is some dataset (usually split into train and validation sets). When run, H_* takes as input a hyper-HP configuration $c \in \mathcal{C}$ and a random seed $r \in \mathcal{I}$, then proceeds to run $\mathcal{A}_\lambda$ (on $\mathcal{M}_\lambda$ using $G(r)$ and data[3] from X) some number of times for different HPs $\lambda \in \Lambda$. Finally, H_* outputs a tuple $\boxed{(\lambda^*, \ell)}$, where λ^* is the HP configuration chosen by HPO and ℓ is the log documenting the run.*

We formalize the process of drawing conclusions from empirical studies using HPO

Definition 1. *A log ℓ records all the choices and measurements made during an HPO run, including the total time T it took to run. It has all necessary information to make the HPO run reproducible.*

We formalize the process of drawing conclusions from empirical studies using HPO

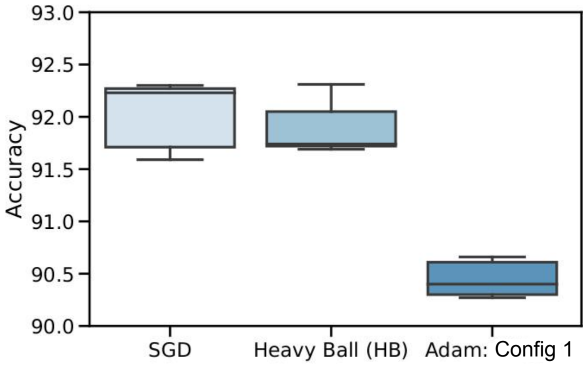
Definition 1. *A log ℓ records all the choices and measurements made during an HPO run, including the total time T it took to run. It has all necessary information to make the HPO run reproducible.*



3 HPO procedures;
Each box plot corresponds to a log

We formalize the process of drawing conclusions from empirical studies using HPO

Definition 1. *A log ℓ records all the choices and measurements made during an HPO run, including the total time T it took to run. It has all necessary information to make the HPO run reproducible.*



3 HPO procedures;
Each box plot corresponds to a log

We then formalize the process of drawing conclusions from empirical studies using HPO

Definition 3. *An* ***epistemic hyperparameter optimization procedure (EHPO)*** *is a tuple* $(\mathcal{H}, \mathcal{F})$ *where* $\mathcal{H}$ *is a set of HPO procedures* H *(Definition* $\boxed{2}$*) and* $\mathcal{F}$ *is a function that maps a set of HPO logs* $\mathcal{L}$ *(Definition* $\boxed{1}$*) to a set of logical formulas* $\mathcal{P}$*, i.e.* $\mathcal{F}(\mathcal{L}) = \mathcal{P}$*. An execution of EHPO involves running each* $H \in \mathcal{H}$ *some number of times (each run produces a log* ℓ*), and then evaluating* $\mathcal{F}$ *on the logs* $\mathcal{L}$ *produced in order to output the conclusions* $\mathcal{F}(\mathcal{L})$ *we draw from all of the HPO runs.*

We then formalize the process of drawing conclusions from empirical studies using HPO

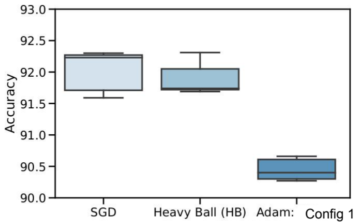
Definition 3. *An **epistemic hyperparameter optimization procedure (EHPO)** is a tuple $(\mathcal{H}, \mathcal{F})$ where $\mathcal{H}$ is a set of HPO procedures H (Definition $\boxed{2}$) and $\mathcal{F}$ is a function that maps a set of HPO logs $\mathcal{L}$ (Definition $\boxed{1}$) to a set of logical formulas $\mathcal{P}$, i.e. $\mathcal{F}(\mathcal{L}) = \mathcal{P}$. An execution of EHPO involves running each $H \in \mathcal{H}$ some number of times (each run produces a log ℓ), and then evaluating $\mathcal{F}$ on the logs $\mathcal{L}$ produced in order to output the conclusions $\mathcal{F}(\mathcal{L})$ we draw from all of the HPO runs.*

Epistemic Hyperparameter Optimization (EHPO) takes a **set of HPO procedures** and a function **F**, which maps a set of HPO procedure **logs** to **conclusions** about algorithm performance

We then formalize the process of drawing conclusions from empirical studies using HPO

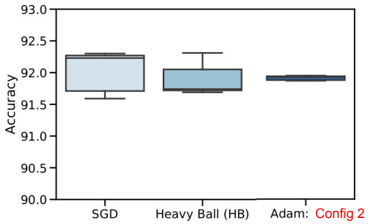
Epistemic Hyperparameter Optimization (EHPO) takes a **set of HPO procedures** and a function **F**, which maps a set of HPO procedure **logs** to **conclusions** about algorithm performance

Log set L1
(|L1| = 3)



F(L1) → "Non-adaptive optimizers outperform adaptive ones" = p

Log set L2
(|L2| = 3)



F(L2) → "Non-adaptive optimizers do not outperform adaptive ones" = $\neg p$

We frame drawing inconsistent conclusions from EHPO in terms of an adversary who can deceive us

"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."

We frame drawing inconsistent conclusions from EHPO in terms of an adversary who can deceive us

"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."

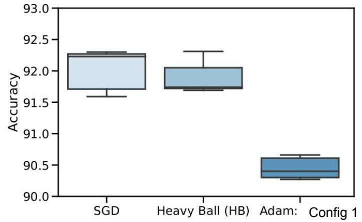
Recall:

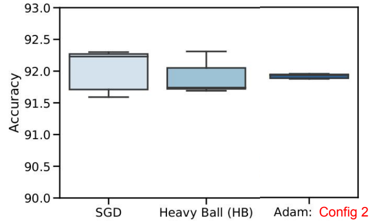
We do not know the ground truth

It is **fine** to accept **either**

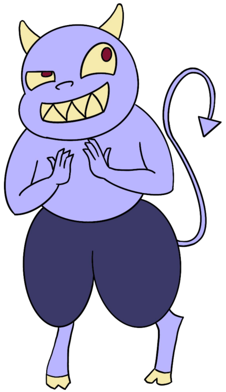
It is **fine** to accept **neither** (form **no conclusions**)

It is **not fine** to accept **both** (inconsistent conclusions are false)





We frame drawing inconsistent conclusions from EHPO in terms of an adversary who can deceive us



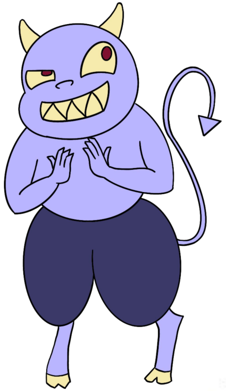
Imagine an evil demon who

is trying to deceive us about relative algorithm performance via EHPO

maintains a set of log HPO logs which it can modify

presents us with a final log set, from which we can draw conclusions

We frame drawing inconsistent conclusions from EHPO in terms of an adversary who can deceive us



Imagine an **evil demon** who

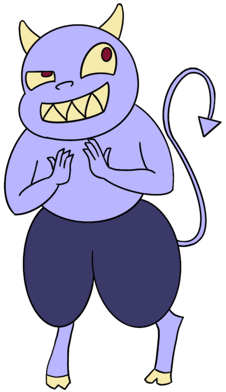
is trying to deceive us about relative algorithm performance via EHPO

maintains a set of log HPO logs which it can modify

presents us with a final log set, from which we can draw conclusions

Definition 4. *A randomized* **strategy** $\boldsymbol{\sigma}$ *is a function that specifies which action the demon will take. Given* $\mathcal{L}$*, its current set of logs,* $\boldsymbol{\sigma}(\boldsymbol{\mathcal{L}})$ *gives a distribution over concrete actions, where each action is either 1) running a new* H *with its choice of hyper-HPs* c *and seed* r *2) erasing some logs, or 3) returning. We let* Σ *denote the set of all such strategies.*

We frame drawing inconsistent conclusions from EHPO in terms of an adversary who can deceive us



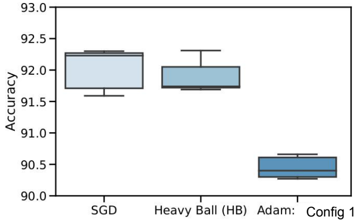
Imagine an evil demon who

is trying to deceive us about relative algorithm performance via EHPO

maintains a set of HPO logs which it can modify

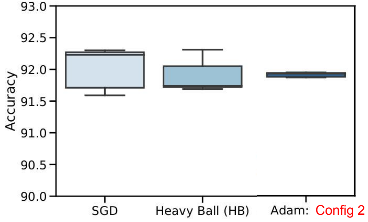
presents us with a final log set, from which we can draw conclusions

could produce L1



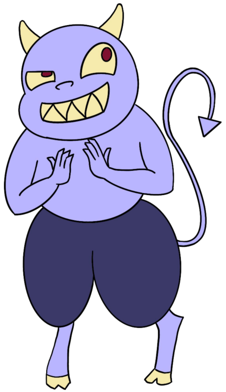
$\text{F(L1)} \rightarrow p$

or *could* produce L2



$\text{F(L2)} \rightarrow \neg p$

We frame drawing inconsistent conclusions from EHPO in terms of an adversary who can deceive us



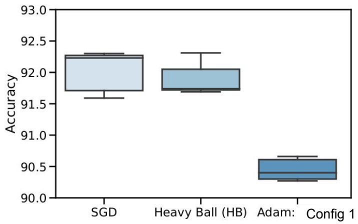
Imagine an evil demon who

is trying to deceive us about relative algorithm performance via EHPO

maintains a set of HPO logs which it can modify

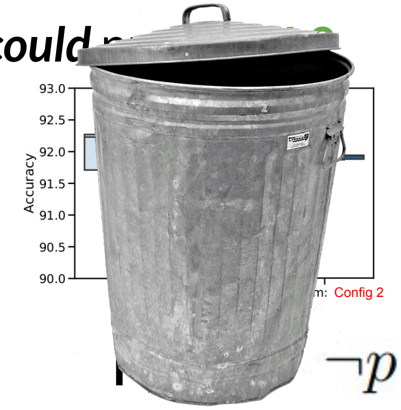
presents us with a final log set, from which we can draw conclusions

could produce L1



F(L1) $\rightarrow \quad p$

or *could*



$\neg p$

Modal logic is the standard way to formalize "could"

Modal logic extends propositional logic to allow us to reason about **possibility** introducing an additional operator

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

Modal logic is the standard way to formalize “could”

Modal logic extends propositional logic to allow us to reason about **possibility** introducing an additional operator

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

$\Diamond\phi$ eads, “It is **possible** that ϕ.”

Modal logic is the standard way to formalize "could"

Modal logic extends propositional logic to allow us to reason about **possibility** introducing an additional operator

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

$\Diamond\phi$ eads, "It is **possible** that ϕ."

We combine two modal operators so that we can capture the idea that **it is not possible to adopt inconsistent beliefs** about experimental outcomes

Modal logic is the standard way to formalize “could”

Modal logic extends propositional logic to allow us to reason about **possibility** introducing an additional operator

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

$\Diamond\phi$ eads, “It is **possible** that ϕ.”

We combine two modal operators so that we can capture the idea that **it is not possible to adopt inconsistent beliefs** about experimental outcomes

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

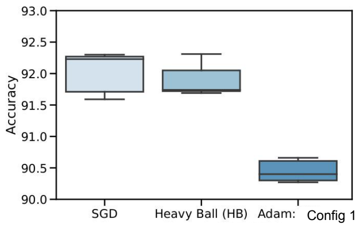
Expressing the possible outcomes of EHPO with $\Diamond_t p$

Syntax $\Diamond_t p$

Intuition The demon could a adopt a strategy for running EHPO that is guaranteed to cause their desired outcome p in at most time *t* in expectation

Semantics A set of EHPO output logs **models** that it is possible p in time t

$\Diamond_t p$



Expressing the possible outcomes of EHPO with $\Diamond_t p$

Syntax $\Diamond_t p$

Definition. *A randomized **strategy** $\boldsymbol{\sigma}$ is a function that specifies which action the demon will take. Given $\mathcal{L}$, its current set of logs, $\boldsymbol{\sigma}(\mathcal{L})$ gives a distribution over concrete actions, where each action is either 1) running a new H with its choice of hyper-HPs c and seed r 2) erasing some logs, or 3) returning. We let Σ denote the set of all such strategies.*

Expressing the possible outcomes of EHPO with $\Diamond_t p$

Syntax $\Diamond_t p$

Definition. *A randomized **strategy** σ is a function that specifies which action the demon will take. Given $\mathcal{L}$, its current set of logs, $\sigma(\mathcal{L})$ gives a distribution over concrete actions, where each action is either 1) running a new H with its choice of hyper-HPs c and seed r 2) erasing some logs, or 3) returning. We let Σ denote the set of all such strategies.*

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

Expressing the possible outcomes of EHPO with $\Diamond_t p$

Syntax $\Diamond_t p$

Definition. *A randomized* ***strategy*** $\boldsymbol{\sigma}$ *is a function that specifies which action the demon will take. Given* $\mathcal{L}$*, its current set of logs,* $\boldsymbol{\sigma}(\mathcal{L})$ *gives a distribution over concrete actions, where each action is either 1) running a new* H *with its choice of hyper-HPs* c *and seed* r *2) erasing some logs, or 3) returning. We let* Σ *denote the set of all such strategies.*

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

Definition. *Let* $\boldsymbol{\sigma}[\mathcal{L}]$ *denote the logs output from executing strategy* σ *on logs* $\mathcal{L}$*, and let* $\boldsymbol{\tau_\sigma}(\mathcal{L})$ *denote the total time spent during execution.* $\tau_\sigma(\mathcal{L})$ *is equivalent to the sum of the times* T *it took each HPO procedure* $H \in \mathcal{H}$ *executed in strategy* σ *to run. Note that both* $\sigma[\mathcal{L}]$ *and* $\tau_\sigma(\mathcal{L})$ *are random variables, as a function of the randomness of selecting* G *and the actions sampled from* $\sigma(\mathcal{L})$*. For any formula* p *and any* $t \in \mathbb{R}_{>0}$*, we say* $\mathcal{L} \models \Diamond_t p$*, i.e. "*$\mathcal{L}$ *models that it is possible* p *in time* t*," if*

$$\textit{there exists a strategy } \sigma \in \Sigma, \textit{ such that } \quad \mathbb{P}(\sigma[\mathcal{L}] \models p) = 1 \;\; \textit{and} \;\; \mathbb{E}[\tau_\sigma(\mathcal{L})] \leq t.$$

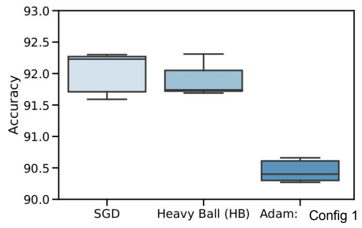
Expressing the possible outcomes of EHPO with $\Diamond_t p$

Syntax $\Diamond_t p$

Intuition The demon could a adopt a strategy for running EHPO that is guaranteed to cause their desired outcome p in at most time *t* in expectation

Semantics A set of EHPO output logs **models** that it is possible p in time t

$\Diamond_t p$



Expressing our belief with $\mathcal{B}p$

Syntax $\boxed{\mathcal{B}p}$

Intuition We believe/ conclude p

Semantics The set of EHPO output logs models our belief in p

Expressing our belief with $\boxed{\mathcal{B}p}$

Syntax $\boxed{\mathcal{B}p}$

Intuition We believe/ conclude p

Semantics The set of EHPO output logs models our belief in p

Definition 6. *For any formula* p*, we say* $\mathcal{L} \models \mathcal{B}p$*, "*$\mathcal{L}$ *models our belief in* p*", if* $p \in \mathcal{F}(\mathcal{L})$.

With both of these operators, we can formalize the problem of *hyperparameter deception*

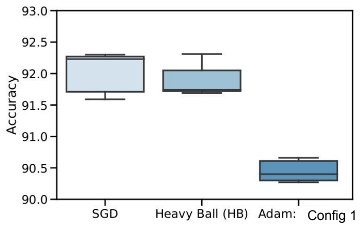
t-non-deceptive axiom $\neg(\Diamond_t \mathcal{B}p \ \wedge \ \Diamond_t \mathcal{B}\neg p)$

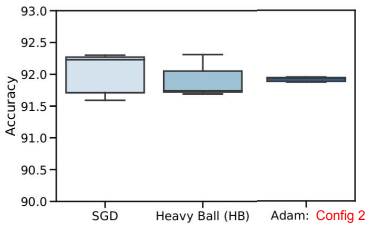
If it is **possible** for the **demon** can get us to **believe** p in time ***t***, then it is **not possible** for the **demon** to get us to **believe** $\neg p$ me ***t***

With both of these operators, we can formalize the problem of *hyperparameter deception*

t-non-deceptive axiom $\neg(\Diamond_t \mathcal{B}p \wedge \Diamond_t \mathcal{B}\neg p)$

If it is **possible** for the **demon** can get us to **believe** p in time ***t***, then it is **not possible** for the **demon** to get us to **believe** $\neg p$me ***t***





Our motivating example is ***t*-deceptive**

We formally study “hyperparameter deception”

The paper formalizes the problem in 2 phases:

1) We remove the vagueness from the problem by pinning down
   a) a concrete formalization for the possible outcomes of running hyperparameter optimization
   b) a concrete formalization of our belief in conclusions

2) **Proving non-trivial theorems about whether a hyperparameter optimization procedure is defended against “deception” (and validating this empirically)**

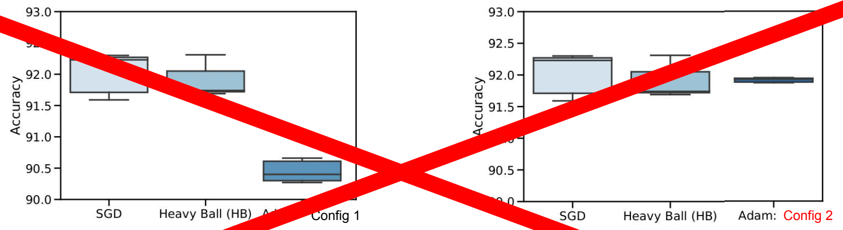
Using this formalization, we can prove *t*-non-deceptive EHPO

We can

suggest **concrete** EHPO

prove that this EHPO satisfies ***t-non-deceptiveness***

Which means, for this EHPO, in time *t* deception **is not possible**



~~Our motivating example is ***t*-deceptive**~~

Using this formalization, we can prove *t*-non-deceptive EHPO

Defense intuition:

Given some naive reasoner, we construct a defended reasoner that is always **more skeptical** than the naive reasoner

If the naive reasoner is *t*-non-deceptive, then any more skeptical reasoner is also *t*-non-deceptive

Using this formalization, we can prove *t*-non-deceptive EHPO

construct $\mathcal{B}_*$ such that for any p, $\mathcal{B}_* p \equiv \mathcal{B}_\text{n} p \wedge \neg \Diamond_t \mathcal{B}_\text{n} \neg p$ (1).

Directly from our axioms (Section 4), we can now prove $\mathcal{B}_*$ is defended. We will suppose it is possible for $\mathcal{B}_*$ to be deceived, demonstrate a contradiction, and thereby guarantee that $\mathcal{B}_*$ is t-non-deceptive. Suppose $\mathcal{B}_*$ can be deceived in time t, i.e. $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p$ is True. Starting with the left, $\Diamond_t \mathcal{B}_* p$:

			Rule
$\Diamond_t \mathcal{B}_* p$	$\equiv$	$\Diamond_t (\mathcal{B}_\text{n} p \wedge \neg \Diamond_t \mathcal{B}_\text{n} \neg p)$	Applying $\Diamond_t$ to the definition of $\mathcal{B}_* p$ (1)
	$\rightarrow$	$\Diamond_t (\neg \Diamond_t \mathcal{B}_\text{n} \neg p)$	Reducing a conjunction to either of its terms: $(a \wedge b) \rightarrow b$
	$\rightarrow$	$\neg \Diamond_t \mathcal{B}_\text{n} \neg p$	Symmetry; dropping all but the right-most operator: $\Diamond_t(\Diamond_t a) \rightarrow \Diamond_t a$

We then pause to apply our axioms to the right side of the conjunction, $\Diamond_t \mathcal{B}_* \neg p$:

			Rule
$\Diamond_t \mathcal{B}_* \neg p$	$\equiv$	$\Diamond_t (\mathcal{B}_\text{n} \neg p \wedge \neg \Diamond_t \mathcal{B}_\text{n} p)$	Applying $\Diamond_t$ to the definition of $\mathcal{B}_* \neg p$ (1)
	$\rightarrow$	$\Diamond_t \mathcal{B}_\text{n} \neg p \wedge \Diamond_t \neg \Diamond_t \mathcal{B}_\text{n} p$	Distributing $\Diamond_t$ over $\wedge$: $\Diamond_t(a \wedge b) \rightarrow (\Diamond_t a \wedge \Diamond_t b)$
	$\rightarrow$	$\Diamond_t \mathcal{B}_\text{n} \neg p$	Reducing a conjunction to either of its terms: $(a \wedge b) \rightarrow a$

We now bring both sides of the conjunction back together: $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p \equiv \neg \Diamond_t \mathcal{B}_\text{n} \neg p \wedge \Diamond_t \mathcal{B}_\text{n} \neg p$. The right-hand side is of the form $\neg a \wedge a$, which must be False. This contradicts our initial assumption that $\mathcal{B}_*$ is t-deceptive (i.e., $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p$ is True). Therefore, $\mathcal{B}_*$ is t-non-deceptive.

Using this formalization, we can prove *t*-non-deceptive EHPO

construct $\mathcal{B}_*$ such that for any p, $\mathcal{B}_* p \equiv \mathcal{B}_\text{n} p \wedge \neg \Diamond_t \mathcal{B}_\text{n} \neg p$ (1).

Directly from our axioms (Section 4), we can now prove $\mathcal{B}_*$ is defended. We will suppose it is possible for $\mathcal{B}_*$ to be deceived, demonstrate a contradiction, and thereby guarantee that $\mathcal{B}_*$ is t-non-deceptive. Suppose $\mathcal{B}_*$ can be deceived in time t, i.e. $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p$ is True. Starting with the left, $\Diamond_t \mathcal{B}_* p$:

Defense takeaway:

It is **always possible** to construct a t-defended EHPO

We then

			Rule
$\Diamond_t \mathcal{B}_* \neg p$	$\equiv$	$\Diamond_t (\mathcal{B}_\text{n} \neg p \wedge \neg \Diamond_t \mathcal{B}_\text{n} p)$	Applying $\Diamond_t$ to the definition of $\mathcal{B}_* \neg p$ (1)
	$\rightarrow$	$\Diamond_t \mathcal{B}_\text{n} \neg p \wedge \Diamond_t \neg \Diamond_t \mathcal{B}_\text{n} p$	Distributing $\Diamond_t$ over $\wedge$: $\Diamond_t (a \wedge b) \rightarrow (\Diamond_t a \wedge \Diamond_t b)$
	$\rightarrow$	$\Diamond_t \mathcal{B}_\text{n} \neg p$	Reducing a conjunction to either of its terms: $(a \wedge b) \rightarrow a$

We now bring both sides of the conjunction back together: $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p \equiv \neg \Diamond_t \mathcal{B}_\text{n} \neg p \wedge \Diamond_t \mathcal{B}_\text{n} \neg p$. The right-hand side is of the form $\neg a \wedge a$, which must be False. This contradicts our initial assumption that $\mathcal{B}_*$ is t-deceptive (i.e., $\Diamond_t \mathcal{B}_* p \wedge \Diamond_t \mathcal{B}_* \neg p$ is True). Therefore, $\mathcal{B}_*$ is t-non-deceptive.

Our *t*-defended random-search EHPO

A defended random search EHPO. Random search takes two hyper-HPs, a distribution μ over the HP space and a number of trials $K \in \mathbb{N}$ to run. HPO consists of K independent trials of training algorithms $\mathcal{A}_{\lambda_1}, \mathcal{A}_{\lambda_2}, \ldots, \mathcal{A}_{\lambda_K}$, where the HPs λ_k are independently drawn from μ, taking expected time proportional to K. When drawing conclusions, we usually look at the "best" run for each algorithm. For simplicity, we suppose there is only one algorithm, $\mathcal{A}$. We bound how much the choice of hyper-HPs can affect the HPs, and define a defended EHPO based on a variant of random search.

Our *t*-defended random-search EHPO

A defended random search EHPO. Random search takes two hyper-HPs, a distribution μ over the HP space and a number of trials $K \in \mathbb{N}$ to run. HPO consists of K independent trials of training algorithms $\mathcal{A}_{\lambda_1}, \mathcal{A}_{\lambda_2}, \ldots, \mathcal{A}_{\lambda_K}$, where the HPs λ_k are independently drawn from μ, taking expected time proportional to K. When drawing conclusions, we usually look at the "best" run for each algorithm. For simplicity, we suppose there is only one algorithm, $\mathcal{A}$. We bound how much the choice of hyper-HPs can affect the HPs, and define a defended EHPO based on a variant of random search.

Our *t*-defended random-search EHPO

A defended random search EHPO. Random search takes two hyper-HPs, a distribution μ over the HP space and a number of trials $K \in \mathbb{N}$ to run. HPO consists of K independent trials of training algorithms $\mathcal{A}_{\lambda_1}, \mathcal{A}_{\lambda_2}, \ldots, \mathcal{A}_{\lambda_K}$, where the HPs λ_k are independently drawn from μ, taking expected time proportional to K. When drawing conclusions, we usually look at the "best" run for each algorithm. For simplicity, we suppose there is only one algorithm, $\mathcal{A}$. We bound how much the choice of hyper-HPs can affect the HPs, and define a defended EHPO based on a variant of random search.

Our *t*-defended random-search EHPO

A defended random search EHPO. Random search takes two hyper-HPs, a distribution μ over the HP space and a number of trials $K \in \mathbb{N}$ to run. HPO consists of K independent trials of training algorithms $\mathcal{A}_{\lambda_1}, \mathcal{A}_{\lambda_2}, \ldots, \mathcal{A}_{\lambda_K}$, where the HPs λ_k are independently drawn from μ, taking expected time proportional to K. When drawing conclusions, we usually look at the "best" run for each algorithm. For simplicity, we suppose there is only one algorithm, $\mathcal{A}$. We bound how much the choice of hyper-HPs can affect the HPs, and define a defended EHPO based on a variant of random search.

Our *t*-defended random-search EHPO

Definition 7. *Suppose that we are given a naive EHPO procedure $(\{H\}, \mathcal{F}_n)$, in which H is random search and is the only HPO in our EHPO, and $\mathcal{F}_n$ is a "naive" belief function associated with a naive reasoner $\mathcal{B}_n$. For any $K, R \in \mathbb{N}$, we define the "(K, R)-defended" belief function $\mathcal{F}_*$ for a skeptical reasoner $\mathcal{B}_*$ as the following conclusion-drawing procedure. First, $\mathcal{F}_*$ only makes conclusion set $\mathcal{P}_*$ from a single log $\hat{\ell}$ with $K * R$ trials; otherwise, it concludes nothing, outputting $\emptyset$. Second, $\mathcal{F}_*$ splits the single $\hat{\ell}$ into R logs $\ell_1, \ell_2, \ldots, \ell_R$, each containing K independent-random-search trials.*[8] *Finally, $\mathcal{F}_*$ outputs the intersection of what the naive reasoner would have output on each log ℓ_i,*

$$\mathcal{F}_*(\{\hat{\ell}\}) = \mathcal{P}_* \equiv \mathcal{F}_n(\{\ell_1\}) \cap \mathcal{F}_n(\{\ell_2\}) \cap \cdots \cap \mathcal{F}_n(\{\ell_R\}).$$

Equivalently, $\{\hat{\ell}\} \models \mathcal{B}_* p$ only if $\{\ell_i\} \models \mathcal{B}_n p$ for all i.

Our *t*-defended random-search EHPO

Definition 7. *Suppose that we are given a naive EHPO procedure* $(\{H\}, \mathcal{F}_n)$*, in which* H *is random search and is the only HPO in our EHPO, and* $\mathcal{F}_n$ *is a "naive" belief function associated with a naive reasoner* $\mathcal{B}_n$*. For any* $K, R \in \mathbb{N}$*, we define the "*(K, R)*-defended" belief function* $\mathcal{F}_*$ *for a skeptical reasoner* $\mathcal{B}_*$ *as the following conclusion-drawing procedure. First,* $\mathcal{F}_*$ *only makes conclusion set* $\mathcal{P}_*$ *from a single log* $\hat{\ell}$ *with* $K * R$ *trials; otherwise, it concludes nothing, outputting* $\emptyset$*. Second,* $\mathcal{F}_*$ *splits the single* $\hat{\ell}$ *into* R *logs* $\ell_1, \ell_2, \ldots, \ell_R$*, each containing* K *independent-random-search trials.*[8] *Finally,* $\mathcal{F}_*$ *outputs the intersection of what the naive reasoner would have output on each log* ℓ_i*,*

$$\mathcal{F}_*(\{\hat{\ell}\}) = \mathcal{P}_* \equiv \mathcal{F}_n(\{\ell_1\}) \cap \mathcal{F}_n(\{\ell_2\}) \cap \cdots \cap \mathcal{F}_n(\{\ell_R\}).$$

Equivalently, $\{\hat{\ell}\} \models \mathcal{B}_* p$ *only if* $\{\ell_i\} \models \mathcal{B}_n p$ *for all* i*.*

Our *t*-defended random-search EHPO

Definition 7. *Suppose that we are given a naive EHPO procedure* $(\{H\}, \mathcal{F}_n)$*, in which* H *is random search and is the only HPO in our EHPO, and* $\mathcal{F}_n$ *is a "naive" belief function associated with a naive reasoner* $\mathcal{B}_n$*. For any* $K, R \in \mathbb{N}$*, we define the "*(K, R)*-defended" belief function* $\mathcal{F}_*$ *for a skeptical reasoner* $\mathcal{B}_*$ *as the following conclusion-drawing procedure. First,* $\mathcal{F}_*$ *only makes conclusion set* $\mathcal{P}_*$ *from a single log* $\hat{\ell}$ *with* $K * R$ *trials; otherwise, it concludes nothing, outputting* $\emptyset$*. Second,* $\mathcal{F}_*$ *splits the single* $\hat{\ell}$ *into* R *logs* $\ell_1, \ell_2, \ldots, \ell_R$*, each containing* K *independent-random-search trials.*[8] *Finally,* $\mathcal{F}_*$ *outputs the intersection of what the naive reasoner would have output on each log* ℓ_i*,*

$$\mathcal{F}_*(\{\hat{\ell}\}) = \mathcal{P}_* \equiv \mathcal{F}_n(\{\ell_1\}) \cap \mathcal{F}_n(\{\ell_2\}) \cap \cdots \cap \mathcal{F}_n(\{\ell_R\}).$$

Equivalently, $\{\hat{\ell}\} \models \mathcal{B}_* p$ *only if* $\{\ell_i\} \models \mathcal{B}_n p$ *for all* i*.*

Our t-defended random-search EHPO

Definition 7. *Suppose that we are given a naive EHPO procedure $(\{H\}, \mathcal{F}_n)$, in which H is random search and is the only HPO in our EHPO, and $\mathcal{F}_n$ is a "naive" belief function associated with a naive reasoner $\mathcal{B}_n$. For any $K, R \in \mathbb{N}$, we define the "(K, R)-defended" belief function $\mathcal{F}_*$ for a skeptical reasoner $\mathcal{B}_*$ as the following conclusion-drawing procedure. First, $\mathcal{F}_*$ only makes conclusion set $\mathcal{P}_*$ from a single log $\hat{\ell}$ with $K * R$ trials; otherwise, it concludes nothing, outputting $\emptyset$. Second, $\mathcal{F}_*$ splits the single $\hat{\ell}$ into R logs $\ell_1, \ell_2, \ldots, \ell_R$, each containing K independent-random-search trials.* $\boxed{8}$ *Finally, $\mathcal{F}_*$ outputs the intersection of what the naive reasoner would have output on each log ℓ_i,*

$$\mathcal{F}_*(\{\hat{\ell}\}) = \mathcal{P}_* \equiv \mathcal{F}_n(\{\ell_1\}) \cap \mathcal{F}_n(\{\ell_2\}) \cap \cdots \cap \mathcal{F}_n(\{\ell_R\}).$$

Equivalently, $\{\hat{\ell}\} \models \mathcal{B}_* p$ only if $\{\ell_i\} \models \mathcal{B}_n p$ for all i.

Our *t*-defended random-search EHPO

Definition 7. *Suppose that we are given a naive EHPO procedure $(\{H\}, \mathcal{F}_n)$, in which H is random search and is the only HPO in our EHPO, and $\mathcal{F}_n$ is a "naive" belief function associated with a naive reasoner $\mathcal{B}_n$. For any $K, R \in \mathbb{N}$, we define the "(K, R)-defended" belief function $\mathcal{F}_*$ for a skeptical reasoner $\mathcal{B}_*$ as the following conclusion-drawing procedure. First, $\mathcal{F}_*$ only makes conclusion set $\mathcal{P}_*$ from a single log $\hat{\ell}$ with $K * R$ trials; otherwise, it concludes nothing, outputting $\emptyset$. Second, $\mathcal{F}_*$ splits the single $\hat{\ell}$ into R logs $\ell_1, \ell_2, \ldots, \ell_R$, each containing K independent-random-search trials.*[8] *Finally, $\mathcal{F}_*$ outputs the intersection of what the naive reasoner would have output on each log ℓ_i,*

$$\mathcal{F}_*(\{\hat{\ell}\}) = \mathcal{P}_* \equiv \mathcal{F}_n(\{\ell_1\}) \cap \mathcal{F}_n(\{\ell_2\}) \cap \cdots \cap \mathcal{F}_n(\{\ell_R\}).$$

Equivalently, $\{\hat{\ell}\} \models \mathcal{B}_* p$ only if $\{\ell_i\} \models \mathcal{B}_n p$ for all i.

Our *t*-defended random-search EHPO

Informally, to draw a conclusion using this EHPO, $\mathcal{B}_*$ splits a random-search-trial log of size $K * R$ into R groups of K-trial logs, passing each K-trial log to one of an ensemble of R naive reasoners $\mathcal{B}_{\mathrm{n}}$. $\mathcal{B}_*$ only concludes p if all R naive reasoners unanimously agree on p. We can guarantee this EHPO to be t-non-deceptive by assuming a bound on how much the hyper-HPs can affect the HPs.

Our *t*-defended random-search EHPO

Informally, to draw a conclusion using this EHPO, $\mathcal{B}_*$ splits a random-search-trial log of size $K * R$ into R groups of K-trial logs, passing each K-trial log to one of an ensemble of R naive reasoners $\mathcal{B}_\text{n}$. $\mathcal{B}_*$ only concludes p if all R naive reasoners unanimously agree on p. We can guarantee this EHPO to be t-non-deceptive by assuming a bound on how much the hyper-HPs can affect the HPs.

Our *t*-defended random-search EHPO

Informally, to draw a conclusion using this EHPO, $\mathcal{B}_*$ splits a random-search-trial log of size $K * R$ into R groups of K-trial logs, passing each K-trial log to one of an ensemble of R naive reasoners $\mathcal{B}_\mathrm{n}$. $\mathcal{B}_*$ only concludes p if all R naive reasoners unanimously agree on p. We can guarantee this EHPO to be t-non-deceptive by assuming a bound on how much the hyper-HPs can affect the HPs.

Theorem 1. *Suppose that the set of allowable hyper-HPs $\mathcal{C}$ of H is constrained, such that any two allowable random-search distributions μ and ν have Renyi-∞-divergence at most a constant, i.e. $D_\infty(\mu \| \nu) \leq \gamma$. The (K, R)-defended random-search EHPO of Definition $\boxed{7}$ is guaranteed to be t-non-deceptive if we set $R \geq \sqrt{t \exp(\gamma K)/K} = O(\sqrt{t})$.*

Our *t*-defended EHPO can make conclusions using fewer than *t* resources

We describe a defended variation of random search

Our defended-random-search EHPO is defended against deception **for time budget *t*,** but it is able to draw conclusions **using up compute budgets that are O(√*t*)**

In short, our algorithm can make conclusions **using much fewer resources** than the total compute budget for which it is defended against deception (i.e., *t*-defended in runtime on the order of √t)

Our *t*-defended random-search EHPO

Algorithm 1 Defense with Random Search

Require: Set of $K * R$ random-search logs $\{\mathcal{L}_i\}_{i=1}^{KR}$, defense subsampling budget M, criterion constant δ, subsample size κ.

```
1: for m = 1, ··· , M do
2:     Subsample κ logs: {L_i}_{i=1}^κ ~ {L_i}_{i=1}^{KR}.
3:     Obtain conclusions {P_i}_{i=1}^κ from {L_i}_{i=1}^κ.
4:     Obtain output conclusion for m:
           P^(m) ← Majority({P_i}_{i=1}^κ)
5: end for
6: if ∃p s.t. ≥ (1 − δ)M of {P^(m)}_{i=1}^M conclude p then
7:     Conclude p.
8: else
9:     Conclude nothing.
10: end if
```

K∗*R*-size super log

Our *t*-defended random-search EHPO

Algorithm 1 Defense with Random Search

Require: Set of $K * R$ random-search logs $\{\mathcal{L}_i\}_{i=1}^{KR}$,
defense subsampling budget M,
criterion constant δ, subsample size κ.

1: **for** $m = 1, \cdots, M$ **do**
2: Subsample κ logs: $\{\mathcal{L}_i\}_{i=1}^{\kappa} \sim \{\mathcal{L}_i\}_{i=1}^{KR}$.
3: Obtain conclusions $\{\mathcal{P}_i\}_{i=1}^{\kappa}$ from $\{\mathcal{L}_i\}_{i=1}^{\kappa}$.
4: Obtain output conclusion for m:
$\mathcal{P}^{(m)} \leftarrow \text{Majority}(\{\mathcal{P}_i\}_{i=1}^{\kappa})$
5: **end for**
6: **if** $\exists p$ s.t. $\geq (1-\delta)M$ of $\{\mathcal{P}^{(m)}\}_{i=1}^{M}$ conclude p **then**
7: Conclude p.
8: **else**
9: Conclude nothing.
10: **end if**

For *M iterations,*
we subsample *κ* logs,
and pass them to
κ naive reasoners B_n

Each iteration *m* concludes
the **majority conclusion**
of the *κ*-sized B_n ensemble

Our *t*-defended random-search EHPO

Algorithm 1 Defense with Random Search

Require: Set of $K * R$ random-search logs $\{\mathcal{L}_i\}_{i=1}^{KR}$,
defense subsampling budget M,
criterion constant δ, subsample size κ.

```
for m = 1, ..., M do
    Subsample κ logs: {L_i}_{i=1}^{κ} ~ {L_i}_{i=1}^{KR}.
    Obtain conclusions {P_i}_{i=1}^{κ} from {L_i}_{i=1}^{κ}.
    Obtain output conclusion for m:
        P^(m) ← Majority({P_i}_{i=1}^{κ})
end for
if ∃p s.t. ≥ (1 − δ)M of {P^(m)}_{i=1}^{M} conclude p then
    Conclude p.
else
    Conclude nothing.
end if
```

If we a meet a certain number of *M*-majority conclusions,

we conclude *p*

Our *t*-defended random-search EHPO

Algorithm 1 Defense with Random Search

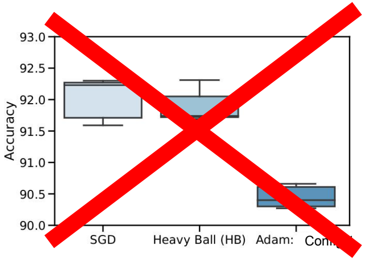
Require: Set of $K * R$ random-search logs $\{\mathcal{L}_i\}_{i=1}^{KR}$, defense subsampling budget M, criterion constant δ, subsample size κ.

for $m = 1, \cdots, M$ **do**
  Subsample κ logs: $\{\mathcal{L}_i\}_{i=1}^{\kappa} \sim \{\mathcal{L}_i\}_{i=1}^{KR}$.
  Obtain conclusions $\{\mathcal{P}_i\}_{i=1}^{\kappa}$ from $\{\mathcal{L}_i\}_{i=1}^{\kappa}$.
  Obtain output conclusion for m:
    $\mathcal{P}^{(m)} \leftarrow \text{Majority}(\{\mathcal{P}_i\}_{i=1}^{\kappa})$
end for
if $\exists p$ s.t. $\geq (1-\delta)M$ of $\{\mathcal{P}^{(m)}\}_{i=1}^{M}$ conclude p **then**
  Conclude p.
else
  Conclude nothing.
end if

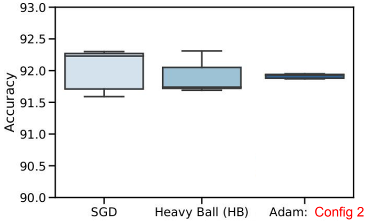
Table 1: Results from repeating our Section 2 experiment, using Algorithm 1 instead of grid search. p = "Non-adaptive optimizers (SGD and HB) perform better than the adaptive optimizer Adam".

	p	$\neg p$	$1-\delta$	Conclude
SGD vs. Adam	0.213	0.788	0.75	$\neg p$
			0.8	Nothing
			0.9	Nothing
HB vs. Adam	0.168	0.832	0.75	$\neg p$
			0.8	$\neg p$
			0.9	Nothing

We empirically validate our suggested *t*-non-deceptive EHPO



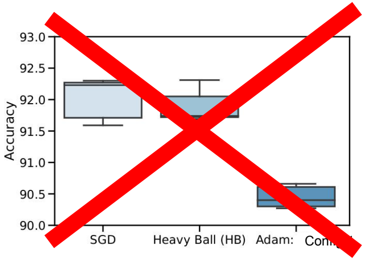
p



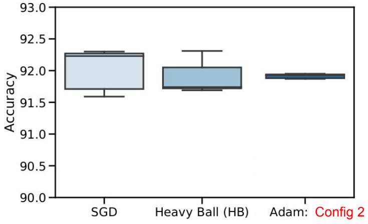
$\neg p$

	p	$\neg p$	$1-\delta$	Conclude
SGD vs. Adam	0.213	0.788	0.75	$\neg p$
			0.8	Nothing
			0.9	Nothing
HB vs. Adam	0.168	0.832	0.75	$\neg p$
			0.8	$\neg p$
			0.9	Nothing

We empirically validate our suggested *t*-non-deceptive EHPO



p

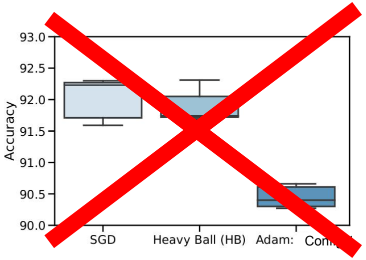


$\neg p$

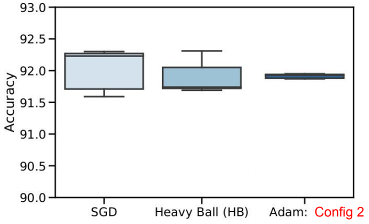
	p	$\neg p$	$1-\delta$	Conclude
SGD vs. Adam	0.213	0.788	0.75	$\neg p$
			0.8	Nothing
			0.9	Nothing
HB vs. Adam	0.168	0.832	0.75	$\neg p$
			0.8	$\neg p$
			0.9	Nothing

Our belief / skepticism

We empirically validate our suggested *t*-non-deceptive EHPO



p



$\neg p$

	p	$\neg p$	$1-\delta$	Conclude
SGD vs. Adam	0.213	0.788	0.75	$\neg p$
			0.8	Nothing
			0.9	Nothing
HB vs. Adam	0.168	0.832	0.75	$\neg p$
			0.8	$\neg p$
			0.9	Nothing

Our belief / skepticism

Takeaways for Robust ML

It is possible to construct t-defended EHPO, such as our *t*-defended random search

Takeaways for Robust ML

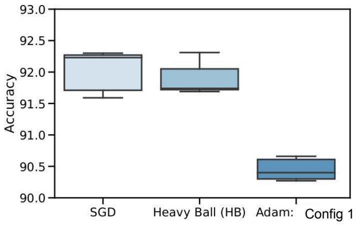
It is possible to construct t-defended EHPO, such as our *t*-defended random search

Any defense will depend on assumptions concerning how we configure underlying HPO, so researchers must be explicit about their configuration choices and their notion of **belief/skepticism**

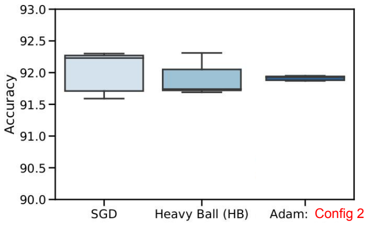
Takeaways for Robust ML

It is possible to construct t-defended EHPO, such as our *t*-defended random search

Any defense will depend on assumptions concerning how we configure underlying HPO, so researchers must be explicit about their configuration choices and their notion of **belief/skepticism**



p



$\neg p$

Takeaways for Robust ML

It is possible to construct t-defended EHPO, such as our *t*-defended random search

Any defense will depend on assumptions concerning how we configure underlying HPO, so researchers must be explicit about their configuration choices and their notion of **belief/skepticism**

Avoiding deception is just as important as ensuring reproducibility, as we want to ensure results are both replicable and correct

Future work

Plug-and-play defended EHPO for a specific class of ML algorithms

- Leverages the logic to suggest a specific, defended grid-search-based hyper-HP selection algorithm (rather than a general possibility proof)
- Takes an adversarial approach to defining our notion of skepticism
- Provides a clear recipe for developing *t*-defended EHPO for different ML domains

Thank you!