





Fanoos: Multi-resolution, Multi-strength, Interactive Explanations for Learned Systems

David Bayani^(✉)  and Stefan Mitsch^(✉) 

Computer Science Department, Carnegie Mellon University,
Pittsburgh, PA 15213, USA
dcbayani@alumni.cmu.edu, smitsch@cs.cmu.edu

Abstract. Machine learning is becoming increasingly important to control the behavior of safety and financially critical components in sophisticated environments, where the inability to understand learned components in general, and neural nets in particular, poses serious obstacles to their adoption. Explainability and interpretability methods for learned systems have gained considerable academic attention, but the focus of current approaches on only one aspect of explanation, at a fixed level of abstraction, and limited if any formal guarantees, prevents those explanations from being digestible by the relevant stakeholders (e.g., end users, certification authorities, engineers) with their diverse backgrounds and situation-specific needs. We introduce Fanoos, a framework for combining formal verification techniques, heuristic search, and user interaction to explore explanations at the desired level of granularity and fidelity. We demonstrate the ability of Fanoos to produce and adjust the abstractness of explanations in response to user requests on a learned controller for an inverted double pendulum and on a learned CPU usage model.

1 Introduction

Explainability and safety in machine learning (ML) are a subject of increasing academic and public concern. As ML continues to grow in success and adoption by wide-ranging industries, the impact of these algorithms' behavior on people's lives is becoming highly non-trivial. Unfortunately, many of the most performant contemporary ML algorithms—neural networks (NNs) in particular—are widely considered black-boxes, with the method by which they perform their duties not being amenable to direct human comprehension. The inability to understand learned components as thoroughly as more traditional software poses serious obstacles to their adoption [1, 5, 13, 28, 30, 52, 88, 89] due to safety concerns,

This material is based upon work supported by the United States Air Force and DARPA under Contract No. FA8750-18-C-0092. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force and DARPA.

difficult debugging and maintenance, and explicit legal requirements (e.g., the “right to an explanation” legislation [24] adopted by the European Union). Symbiotic human-machine interactions can lead to safer and more robust agents, but this task requires effective and versatile communication [66, 79].

Interpretability of learned systems has been studied in the context of computer science intermittently since at least the late 1980s, particularly in the area of formal analysis (e.g., [15, 42, 55, 81, 85, 86]), rule extraction (e.g., [4]), adaptive/non-linear control analysis (e.g., [18]), and various rule-learning paradigms (e.g., inductive logic programming [56], association rule learning [3]). Notwithstanding this long history, main-stream attention has risen only recently due to increased impact on daily life of opaque AI [1] with novel initiatives focused on the problem domain, e.g. [31, 58] and workshops in IJCAI and ICAPS.

Despite this attention, however, most explanatory systems developed for ML lack any formal guarantees with respect to how their descriptions reflect system behavior and are hard-coded to provide a single type of explanation with descriptions at a certain fixed level of abstraction. This not only prevents the explanations generated from being digestible by multiple audiences (the end-user, the intermediate engineers who are non-experts in the ML component, and the ML-engineer for instance) as highlighted by the taxonomy presented in [6], but in fact limits the use by any single audience since the levels of abstraction and formal guarantees needed are situation and goal specific, not just a function of the recipient’s background. When using a microscope, one varies between low and high magnification in order to find what they are looking for and explore samples; these same capabilities are desirable for XAI for much the same reasons.

For example, most consumers of autonomous vehicles may prefer to ask general questions—for instance, “What do you do when you detect a person in front of you?”—and receive a break-down of qualitatively different behaviors for different situations, such as braking when traveling slowly enough, and doing a sharp swerve when traveling too fast to brake. An engineer checking actuator compliance, however, might require greater details, opting to specify precise parameters of the scene and preferring that the car report exact motor commands; the context of use and the audience determine which level of abstraction is best, and supporting multiple types of abstractions in turn supports more use-cases and audiences. Further, the explanations for such a component need to range from formal guarantees to rough tendencies—it may be critical to formally guarantee that the car will always avoid collisions, while it might be sufficient that it usually (but perhaps not always) drives slowly when its battery is low.

The divide between formal and probabilistic explanations also relates to events that are imaginable versus events that may actually occur; formal methods may check every point in a space for conformance to a condition, but if bad behavior only occurs on measure-zero sets, the system would be safe while not being provably so in formalizations lacking knowledge of statistics (e.g., if some criteria demands that a car keep distance >10 cm from obstacles, formally we can get arbitrarily close but not equal; in practice, the difference with ≥ 10 cm

might be irrelevant). Explainable ML systems should enable these sorts of search and smooth variation in need, but at the moment they do not in general.

To address these needs, we introduce Fanoos,¹ an algorithm blending a diverse array of technologies to interactively provide explanations at varying levels of abstraction and fidelity to meet user’s needs. Our algorithm is applicable to currently ubiquitous ML methods, such as feed-forward neural networks (FFNNs) and high-dimensional polynomial kernels. Fanoos offers the following combination of capabilities, which are our contributions:

- Interactivity that allows users to query the learned system they want to understand, and receive explanations characterizing the input requirements, output behavior, or the combination of the two.
- Explanations that can either be formally sound or probabilistic based on the user’s choice. Formal soundness is a capability missing from the vast majority of XAI systems focused on ML, and leveraging verification techniques for ML-related XAI has been underexplored.
- Explanations that can vary in abstraction level.

2 The Methodology of Fanoos

Fanoos is an interactive system that allows users to pose a variety of questions grounded in a domain specification (e.g., asking what environmental conditions cause a robot to swerve left), receive replies from the system, and request that explanations be made more or less abstract. Crucially, Fanoos provides explanations of high fidelity while considering whether the explanation should be formally sound or probabilistically reasonable (which removes the “noise” incurred by measure-zero sets that can plague formal descriptions). To this end, we combine techniques from formal verification, interactive systems, and heuristic search over knowledge domains when responding to user questions and requests.

2.1 Knowledge Domains and User Questions

In the following discussion, let L be the learned system under analysis (which we will assume is piece-wise continuous), q be the question posed by the user, S_I be the (bounded) input space to L , and S_O be the output space for L , $S_{IO} = S_I \cup S_O$ be the joint of the input and output space, and r be the response given by the system. Subscripts I for input, O for output, etc., are simply symbols, not any richer objects. In order to formulate question q and response r , a library listing basic domain information (D) is provided to Fanoos; D lists what S_I and S_O are and provides a set of predicates, P , expressed over the domain symbols in S_{IO} , i.e., for all $p \in P$, the free variables $FV(p)$ are chosen from the variable names $V(S_{IO})$, that is $FV(p) \subseteq V(S_{IO})$. Notably, P is user-extensible and may be generated by automated or semi-automated means.

¹ “Fanoos” (فانوس) means lantern in Farsi. Our approach shines a light on black-box AI. Source code can be found at [7], and an extended exposition is in [8].

Table 1. Description of questions that can be posed to Fanoos

Type q_t	Question content q_c			Description
	Accepts	Illum.	Restrictions	
When do you ^a	Subset s of S_O s.t. $\exists x \in s. q_c(x)$. Found with SAT-solver	S_I	No variables from S_I	Tell the user all sets (formal consideration of all cases) in the input space S_I that have the potential to cause q_c
What do you do when ^b	Subset s of S_I s.t. $\exists x \in s. q_c(x)$. Found with SAT-solver	S_O	No variables from S_O	Tell user all possible learner responses in the collection of input states that q_c accepts
What are the circumstances in which ^c	Subset s of S_{IO} s.t. $\exists x \in s. q_c(x)$. Found with SAT-solver	S_{IO}	None	Tell the user information about what input-output pairs occur in the subset of input-outputs accepted by q_c
... Usually ^d	Subsets over which q_c is true at least once via statistical sampling			Statistical tendency. Avoids measure-zero sets that are unlikely seen in practice

^a when_do_you_move_at_high_speed?

Predicate $pinD$

^b what_do_you_do_when *and* (close_to_target_orientation, close_to_target_position)?

^c what_are_the_circumstances_in_which

and (close_to_target_position, steer_to_right) *or* move_at_low_speed?

^d when_do_you_usually move_at_low_speed *or* steer_to_left?

For queries that formally guarantee behavior (see the first three rows in Table 1), the relevant predicates in P need to expose their internals as first-order formulas; this enables us to guarantee they are satisfied over all members of sets we provide via typical SAT-solvers (such as Z3 [19]). Probabilistic queries require only being able to evaluate question q on a variable assignment provided.

The members of P can be generated in a variety of ways, e.g., by forming most predicates through procedural generation and then using a few hand-tailored predicates to capture particular cases. Notably, since the semantics of the predicates are grounded, they have the potential to be generated from demonstration. For example, operational definitions of “high”, “low”, etc., might be derived from sample data by setting thresholds on quantile values—e.g., 90% or higher might be considered “high” (see, for instance, Sect. 5); further resources and considerations on predicate generation can be found in [8].

2.2 Reachability Analysis of the Learned System

Having established what knowledge Fanoos is given, we proceed to explain our process. First, users select a question type q_t and the content of the question q_c to query the system. That is, $q = (q_t, q_c)$, where q_t is a member of the first column of Table 1 and q_c is a sentence in disjunctive normal form (DNF) over a subset of P that obeys the restrictions listed in Table 1. To ease discussion, we will refer to variables and sets of variable assignments that q accepts (AC_q) and those that q illuminates (IL_q), with the intuition being that the user wants to know what configuration of illuminated variables result in (or result from) the variable

configurations accepted by q_c ; see Table 1 for example queries. When a user asks a question, Fanoos answers by describing a collection of situations that necessarily include those related to the user’s question; this answer is conservative in that it may include additional situations, but never excludes cases.

With question q provided, we analyze the learned system L to find subsets in the inputs S_I and outputs S_O that agree with configuration q_c and the (overapproximated) behavior of L . Specifically, we use CEGAR [16] with boxes (hyper-cubes) as abstractions and a random choice between a bisection or trisection along the longest normalized axis as the refinement process to find the collect of box tuples, B , specified below:

$$B = \{(B_I^{(i)}, B_O^{(i)}) \in \mathcal{B}(S_I) \times \mathcal{B}(S_O) \mid B_O^{(i)} \supseteq L(B_I^{(i)}) \\ \wedge \exists (c, d) \in T. (AC_q(B_c^{(i)}) \wedge IL_q(B_d^{(i)}))\}$$

where $\mathcal{B}(X)$ is the set of boxes over space X and $T = \{(O, I), (I, O), (IO, IO)\}$. For feed-forward neural nets with non-decreasing activation functions, B can be found by (i) covering the input space, (ii) propagating boxes through the network, (iii) testing membership to B of the resulting input- and output-boxes, and (iv) refining abstract states as needed over input-boxes that produce output-boxes overlapping with B ; we detail this process further below.

Covering the Input Space. We cover the input space via iterative dissection informed by properties of the problem, avoiding a naïve gridding of the entire space unless repeated refinement has revealed that to be necessary. The exact sizes of the boxes found by CEGAR are determined by a series of hyper-parameters, which Fanoos maintains in *states*. Hyper-parameters include, e.g., the maximum number of refinement iterations or the minimal size abstractions; an overview of typical hyper-parameters to CEGAR can be found in [10, 15, 16].

Prior to proceeding, B may undergo some limited merging, particularly when an increase of abstraction level is sought. Our merging process is closed over the family of abstract states we have selected; up to a numerical precision threshold, boxes may only merge together to form larger boxes, and only if the smaller boxes formed a partition of the larger box. Value differences within the merging threshold are considered a match (i.e., a soft-match), and allow the pertinent sets of boxes to merge into larger boxes with slightly larger net volumes. Note that enlarging boxes only makes our estimates conservative, and thus continues to ensure the soundness of Fanoos. On exact matches, merging increases the size of abstract states without anywhere increasing the volume of their union—this is not necessarily what would occur if one attempted the CEGAR analysis again with parameters promoting higher granularity. Essentially, merging here is one strategy of increasing abstraction level while retaining some finer-resolution details that might otherwise be lost in a larger volume superset. As before, the state maintains parameters to control the extent of this stage’s merging. Optimal box-merging itself is an NP-hard task, so we adopted a roughly greedy approximation scheme interlaced with hand-written heuristics for accelerating

match-finding (e.g., feasibility checks via shared-vertex lookups) and parameters bounding the extent of computation.

Propagating Boxes Through Networks. In this subsection, we discuss how we conduct our abstract interpretation domains (AIDs) analysis on an FFNN.

Here, we leverage the fact that we are using a pre-trained, fixed-weight feed-forward neural net, that has a typical MLP-like (multi-layer perceptron-like) structure: the network consists of layers of units, each unit being comprised of a scalar-valued affine transformation of the previous layer’s output that is then passed through a non-decreasing (and typically non-linear) activation function, such as a tanh, sigmoidal, or piecewise linear function. For analyzing recurrent neural nets or other systems with loops, more sophisticated mechanisms, such as reachable-set fixed-point calculations, would be necessary in general (see [17]).

As introduced above, we use boxes as the abstract domain, which facilitate a basic implementation since they are easier to manipulate and check for membership than more complex convex polytopes, at the price of typically being less precise per unit volume;² more complex AIDs can be added to Fanoos.

We first examine how boxes are transformed when passing through a single unit, before extending the process to the entire network. Let $u : \mathbb{R}^{I_u} \rightarrow \mathbb{R}^{O_u}$ be a unit of the network with input dimension I_u and output dimension O_u ($I_u, O_u \in \mathbb{N} \setminus \{0\}$), and \mathcal{S}_u be an input box $\times_{i \in [I_u]} [a_i, b_i]$ to unit u (Cartesian product of closed, real intervals $[a_i, b_i]$, and where $[n] = \{k \in \mathbb{N} \setminus \{0\} \mid k \leq n\}$). We want to calculate $u(\mathcal{S}_u)$. Further, let $w \in \mathbb{R}^{I_u}$ be the weights of the unit u , $\beta \in \mathbb{R}$ be the bias, $x \in \mathbb{R}^{I_u}$ be the input value and ρ be a non-decreasing activation function. We have that:

$$u_{linear}(x) = \langle w, x \rangle + \beta, \quad u_\rho(x) = \rho(\langle w, x \rangle + \beta) = \rho(u_{linear}(x))$$

where, $\langle \cdot, \cdot \rangle$ is the L^2 inner product. Since ρ is a non-decreasing function, the extrema of $u_\rho(x)$ and $u_{linear}(x)$ occur at the same arguments. Thus, to find all relevant extreme values over the input space, it suffices to find the values in \mathcal{S}_u that maximize or minimize $\langle w, x \rangle$ as follows:

$$\operatorname{argmin}_{x \in \mathcal{S}_u} \langle w, x \rangle = \langle b_i \mathbb{1}(\{w\}_i \leq 0) + a_i \mathbb{1}(\{w\}_i > 0) \mid i \in [I_u] \rangle$$

where $\langle \cdot \mid i \rangle$ is sequence construction, $\{\cdot\}_i$ accesses the i -th component of a vector, and $\mathbb{1}(\cdot)$ is an indicator function ($\mathbb{1}(\top) = 1$, $\mathbb{1}(\perp) = 0$). The argmax can be found in a similar fashion by swapping the roles of a_i and b_i . With this, we compute the images of the input space under the activation functions as follows:

$$u(\mathcal{S}_u) = [u(\operatorname{argmin}_{x \in \mathcal{S}_u} \langle x, w \rangle), u(\operatorname{argmax}_{x \in \mathcal{S}_u} \langle x, w \rangle)] \quad \text{where } u \in \{u_{linear}, u_\rho\}.$$

Having established how a box should be propagated through a unit in the network, propagation through the entire network follows immediately. Let $u_{i,j}$ be

² In the case of interval arithmetic, this over-approximation and inclusion of additional elements is often called the “wrapping effect” [42].

the i^{th} unit on the j^{th} layer, M_j be the j^{th} layer’s size, $\mathcal{I}_{i,j}$ be the input box to unit $u_{i,j}$, and $m_{i,j} \subseteq [M_j]$ s.t. $|m_{i,j}| = I_{u_{i,j}}$: we simply feed the output box from one layer into the next similar to the usual feed-forward operation:

$$u_{i,j+1}(\mathcal{I}_{i,j+1}) = u_{i,j+1}\left(\bigtimes_{h \in m_{i,j+1}} u_{h,j}(\mathcal{I}_{h,j})\right) \quad (1)$$

Finally, induction shows that these arguments together establish that this process produces a set which contains the image of the network over the box. Notice that approximations creep in during this recursive process; consider, for instance, the bounding rectangle formed for a NN with a 2-d inner-layer whose output exists on a diagonal line whenever the network processes instances in S_I .

Various extensions exist, such as to handle common featurization pre- and post-processings that preserve vector partial-orderings, as well as to aid efficiency; see [8] for more details.

Refining Abstract States. CEGAR [16] is a well-regarded model checking technique for soundly ensuring a system meets desirable properties. In short, the approach uses abstract states carefully discovered through trial and error to attempt verification or refutation; if the desirable property cannot be proven, the algorithm iteratively refines the abstraction based on where the property is in doubt, stopping when the property is either provable or has been disproven by a discovered counterexample. When applied to certain families of discrete programs, results returned by CEGAR are both sound and complete, at the cost of unknown termination of CEGAR in the general case, when no approximations are used. In practice, approximations used with CEGAR tend to err on the safe side: if CEGAR indicates a property holds, then it is true, but the converse might not hold. This flexibility has allowed for extensions of the technique to many domains, including in hybrid system analysis [15], where the state space is necessarily uncountably infinite and system dynamics do not typically have exact numerical representations.

We now overview our CEGAR-like³ abstract state refinement, using boxes as the abstraction domain. As before, we let L be a learned system $L : S_I \rightarrow S_O$ with $S_I \subseteq \mathbb{R}^{I_L}$ and $S_O \subseteq \mathbb{R}^{O_L}$; further, suppose S_I is a box $\bigtimes_{i \in [I_L]} [a_{L,i}, b_{L,i}]$.⁴ Let $\phi : \mathbb{R}^{I_L} \times \mathbb{R}^{O_L} \rightarrow \{\top, \perp\}$ be a formula which we would like to characterize L ’s conformance to over S_I (i.e., find $\{(w, y) \in S_I \times S_O \mid \phi(w, y) \wedge (y = L(w))\}$). Notice that ϕ need not use all of its arguments—so, for instance, the value of ϕ might only vary with changes to input-space variables, thus specifying conditions

³ Elements of our abstract state refinement algorithm may be analogous to CEGAR and its standard extensions—for instance, we perform sampling-based feasibility checks prior to SAT-checks, which may be comparable to spuriousness checks in CEGAR. However, to avoid implying a stringent adherence to canon (i.e., [16] verbatim), we use a different name.

⁴ Strictly speaking, we could discuss a box containing S_I (i.e., a superset), but introducing an auxiliary, potentially larger definition domain might add confusion while giving little benefit.

over the input space but none over the output space. Since CEGAR is not generally guaranteed to terminate, we introduce a function **STOP** : $S_I \rightarrow \{\top, \perp\}$ which will be used to prevent unbounded depth exploration of volumes whose members have mixed truth values under ϕ .

We first form initial abstraction states over the input space; for this, our implementation uses states that do not leverage any expert impressions as to what starting sets would be informative for the circumstances. Instead, we opted for the simple, broadly-applicable strategy of forming high-dimensional “quadrants”: 2^{I_L} hyper-cubes formed by bisecting the input space along each of its axes; we could have just as easily used the universal bounding box undivided to start. The algorithm takes an input-abstraction, w , that has yet to be tried and generates an abstract state, \tilde{o} , that contains $L(w)$ (notice that w and $L(w)$ are both sets). If no member of $w \times \tilde{o}$ is of interest (i.e., meets the condition specified by ϕ), the algorithm returns the empty set. On the other hand, if $w \times \tilde{o}$ has the potential to contain elements of interest then the algorithm continues, attempting to find the smallest allowed abstract states that potentially include interesting elements. In general, further examination is performed by refining the input abstraction, then recursing on the refinements; for efficiency, we also check whether the entire abstract state satisfies ϕ , in which case we are then free to partition it into smaller abstractions without further checks.

Given a box, we refine by splitting along its longest “scaled” axis, h :

$$h = \underset{i \in [I_L]}{\operatorname{argmax}} \frac{b'_i - a'_i}{b_{L,i} - a_{L,i}}$$

We then either bisect ($k = 2$) or trisect ($k = 3$) the chosen axis with probability 0.8 or 0.2 respectively, a design choice balancing between faster analysis, further exploration of diverse abstract states, and keeping boxes of reasonable size:

$$\operatorname{refine}_k(\times_{i \in [I_L]} [a'_i, b'_i]) = \bigcup_{j=0}^{k-1} \left\{ \times_{i \in [I_L]} [a'_i + \mathbb{1}(i = h)jC_k, b'_i + \mathbb{1}(i = h)(j + 1 - k)C_k] \right\},$$

where $C_k = \frac{b'_h - a'_h}{k}$. The use of $b_{L,i} - a_{L,i}$ in the denominator for h is an attempt to control for differences in scaling and meaning among the variables comprising the input space. For instance, 20 mm is not commiserate with 20 radians, and our sensitivity to 3 cm of difference may be different given a quality that is typically on par of kilometers versus one never exceeding a decimeter. Our refinement strategy allows for efficient caching and reloading of refinement results by storing the refinement paths, as opposed to encoding entire boxes. Parameters in the state determine if cached results are reused; reuse improves efficiency and may help reduce uncalled-for volatility in descriptions reported to users, while regenerating results may produce different AIDs which could lead to a better outcome. Our analysis used the following **STOP** function:

$$\mathbf{STOP}(\times_{i \in [I_L]} [a'_i, b'_i]) = (b'_h - a'_h \leq \epsilon(b_{L,i} - a_{L,i})). \quad (2)$$

Algorithm 1: Pseudocode for CEGAR-like abstract state refinement, b is an AID element over the input space (i.e., $b \subseteq S_I$)

```

1 Function RefineAbstractState( $b$ , STOP,  $\phi$ ,  $L$ ):
2    $\tilde{o} \leftarrow \text{approxImage}_L(b)$ ; // AIDs-based image approx., see Eq. (1)
3    $\text{verdict}_1 \leftarrow \text{sat}(\forall x \in b \times \tilde{o}. \neg \phi(x))$ ;
4   if  $\text{verdict}_1$  then
5      $\perp$  return  $\{\}$ ;
6   if STOP( $b$ ) then
7      $\perp$  return  $\{b\}$ ;
8    $\text{verdict}_2 \leftarrow \text{sat}(\forall x \in b \times \tilde{o}. \phi(x))$ ;
9   if  $\text{verdict}_2$  then
10     $\text{boxesToRefine} \leftarrow \{b\}$ ;    $\text{result} \leftarrow \{\}$ ;
11    while  $\text{boxesToRefine}$  is not empty do
12       $c \leftarrow \text{boxesToRefine.pop}()$ ;
13      if STOP( $c$ ) then
14         $\perp$   $\text{result} \leftarrow \text{result} \cup \{c\}$ ;
15      else
16         $\perp$   $\text{boxesToRefine} \leftarrow \text{boxesToRefine} \cup \text{refine}(c)$ ;
17    return  $\text{result}$ ;
18  return  $\bigcup_{r \in \text{refine}(b)} \text{RefineAbstractState}(r, \text{STOP}, \phi, L)$ ;

```

Here, ϵ is the refinement parameter initially specified by the user, but which is then automatically adjusted by operators acting on the state as the user interactions proceed. Similar to the choice of AID, our approach is amenable to more sophisticated refinement and stopping strategies than presented here.

Algorithm 1 addresses formally sound question types; for probabilistic question types (i.e., those denoted with “...usually”), verdict_1 is determined by repeated random sampling, and verdict_2 is fixed as \perp . In our implementation, feasibility checks are done prior to calling the SAT-solver when handling a formally sound question type.

2.3 Generating Descriptions

Having generated B , we produce an initial response, r_0 , to the user’s query in three steps as follows: (i) for each member of B , we extract the box tuple members that were illuminated by q (in the case where S_{IO} is illuminated, we produce a joint box over both tuple members), forming a set of joint boxes, B' ; (ii) next, we heuristically search over predicates P for members that describe box B' and compute a set of predicates covering all boxes; (iii) finally, we format the box covering for user presentation. A sample result answer is shown in Fig. 1 (a), and details on steps (ii) and (iii) follow below.

Producing a Covering of B' . Our search over P for members covering B' is largely based around the greedy construction of a set covering that uses a carefully designed candidate evaluation score.

For each member $b \in B'$, we want to find a set of candidate predicates capable of describing the box to form a larger covering. We find a subset $P_b \subseteq P$ that is consistent with b in that each member of P_b passes the checks called for by q_i when evaluated on b (see the Description column of Table 1). This process is expedited by a feasibility check of each member of P on a vector randomly sampled from b , prior to the expensive check for inclusion in P_b . Having P_b , we filter the candidate set further to P'_b : members of P_b that appear most specific to b ; notice that in our setting, where predicates of varying abstraction level co-mingle in P , P_b may contain many members that only loosely fit b . The subset P'_b is formed by sampling outside of b at increasing radii (in the ℓ_∞ sense) and collecting those members of P_b that fail to hold true at the earliest radius. Importantly, looking ahead to forming a full covering of B , if none of the predicates fail prior to exhausting this sampling, we report P'_b as empty, allowing us to handle b downstream as we will detail in a moment; this avoids having “difficult” boxes force the use of weak predicates that would “wash out” more granular details. The operational meaning of “exhausting”, as well as the radii sampled, are all parameters stored in the state. Generally speaking, we try to be specific at this phase under the assumption that the desired description granularity was determined earlier, primarily during the abstract state refinement. For instance, if we want a subset of P_b that was less specific to b than P'_b , we might reperform the abstract state refinement so to produce larger abstract states. In extensions of our approach, granularity can also be determined earlier by altering P ; our current implementation has first steps in this direction, allowing users to enable an optional operator that filters P based on estimates of a model trained on previous interaction data. We comment further on this extension in Sect. 2.4 and indicate why this operator is left as optional in Sect. 5.

To handle boxes for which P'_b was empty, in general we insert into P'_b a box-range predicate: a *new* atomic predicate that simply lists the variable ranges in the box (e.g., “Box(x : [-1, 0], y : [0.5, 0.3])”). As a result of providing cover for only one box, such predicates will only be retained by the (second) covering we perform in a moment if no other predicates selected are capable of covering the box’s axes. When a request to increase the abstraction level initially finds P'_b empty, we may (as determined by state parameters) set P'_b equal to P_b as opposed to introducing a box-range predicate. If P_b is empty as well, we are forced to add the novel predicate.

We next leverage the P'_b sets to construct a covering of B' , proceeding in an iterative greedy fashion. Specifically, we form an *initial* covering

$$K_f = \mathcal{C}_f \left(\bigcup_{b \in B'} \bigcup_{p \in P'_b} \{(p, b)\}, P \right)$$

where $\mathcal{C}_i(R, H)$ is the covering established at iteration i , incrementing to

$$\mathcal{C}_{i+1}(R, H) = \mathcal{C}_i(R, H) \cup \left\{ \operatorname{argmax}_{p \in H \setminus \mathcal{C}_i(R, H)} \mu(p, \mathcal{C}_i(R, H), R) \right\}$$

where $\mathcal{C}_0(R, H) = \emptyset$, f is the iteration of convergence, and the cover score μ is

$$\mu(p, \mathcal{C}_i(R, H), R) = \sum_{b \in B'} \mathbb{1}(|\text{UV}(b, \mathcal{C}_i(R, H)) \cap \text{FV}(p)| > 0) \mathbb{1}((p, b) \in R)$$

and $\text{UV}(b, \mathcal{C}_i(R, H))$ is the set of variables in b that are not constrained by $\mathcal{C}_i(R, H) \cap P_b$; since the boxes are multivariate and our predicates typically constrain only a subset of the variables, we select predicates based on how many boxes would have open variables covered by them. Notice that K_f is not necessarily an approximately minimal covering of B with respect to members of P . By forcing $p \in P'_b$ when calculating the cover score μ , we enforce additional specificity criteria that the covering should adhere to. At this stage, due to the nature of P'_b being more specific than P_b , it is possible that some members of K_f cover one another: there may exist $p \in K_f$ such that $K_f \setminus \{p\}$ still covers as much of B' as K_f did. By forming K_f , we have found a collection of predicates that can cover B' to the largest extent possible, selected based on how much of B' they were *specific* over (given by the first argument to \mathcal{C}_f when forming K_f). We now remove predicates that are dominated by other (potentially less-specific) predicates that we had to include by performing a second covering:

$$C_F = \mathcal{C}_F \left(\bigcup_{b \in B'} \bigcup_{p \in P_b} \{(p, b)\}, K_f \right).$$

Cleaning and Formatting Output for User. Having produced C_F , we collect the covering's content into a formula in DNF. If $b \in B'$ and s is a maximal, non-singleton subset of $C_F \cap P_b$, then we form a conjunction over the members of s , excluding conjuncts that are implied by others. Concretely, for $A = \bigcup_{b \in B'} \{P_b \cap C_F\}$, we construct:

$$d_0 = \left\{ \bigwedge_{p \in s} p \mid s \in A \wedge \neg(\exists s' \in A. s \subsetneq s') \right\}.$$

The filtering done in d_0 is only to aid efficiency; in a moment, we do a final redundancy check that would achieve similar results even without the filtering in d_0 . Ultimately, the members of d_0 are conjunctions of predicates, with their membership to the set being a disjunction. Prior to actually converting d_0 to DNF, we form d'_0 by: (i) removing any $c \in d_0$ that are redundant given the rest of d_0 (in practice, d_0 is small enough to simply do full one-vs-rest comparison and determine results with a SAT-solver); (ii) attempting to merge any remaining box-range predicates into the minimal number necessary to cover the sets they are responsible for. Note that this redundancy check is distinct from forming C_F out of K_f , which worked at the abstract-state level (and so is unable to tell if a disjunction of predicates covered a box when no individual predicate covered it fully) and attempted to select predicates by maximizing a score.

Finally, r_0 is constructed by listing each c that exists in d'_0 sorted by two relevance scores: first, the approximate proportion of the volume in B' uniquely covered by c , and second by the approximate proportion of total volume c covers in B' . These sorting-scores can be thought of similarly to recall measures.

Specificity is more difficult to tackle, since it would require determining the volume covered by each predicate (which may be an arbitrary first-order formula) across the box bounding the universe, not just the hyper-cubes at hand; this can be approximated for each predicate using set-inversion, but requires non-trivial additional computation for each condition.

2.4 User Feedback and Revaluation

Based on the initial response r_0 , users can request a more abstract or less abstract explanation. We view this alternate explanation generation as another heuristic search, where the system searches over a series of states to find those that are deemed acceptable by the user (consecutive user requests can be viewed in analogy to paths in a tree of Fanoos’s states). The states primarily include algorithm hyper-parameters, the history of interaction, the question to be answered, and the set B . Abstraction and refinement operators take a current state and produce a new one, often by adjusting the system hyper-parameters and recomputing B . This state-operator model of user response allows for rich styles of interaction with the user, beyond and alongside of the three-valued responses of acceptance, increase, or decrease of the abstraction level shown in Fig. 1(b).

For instance, a history-travel operator allows the state (and thus r) to return to an earlier point in the interaction process, if the user feels that response was more informative; from there, the user may investigate an alternate path of abstractions. Other implemented operators allow for refinements of specified parts of explanations as opposed to the entire reply; the simplest form of this is by regenerating the explanation without using a predicate that the user specified be ignored, while a more sophisticated operator determines the predicates to filter out automatically by learning from past interaction. Underlying the discussion of these mechanisms is the utilization of a concept of abstractness, a notion we further comment on in the next subsection.

As future work, we are exploring the use of active learning leveraging user interactions to select operators, with particular interest in bootstrapping the learning process using operationally defined oracles to approximate users.

2.5 Capturing the Concept of Abstractness

The criteria to judge degree-of-abstractness in the lay sense are often difficult to capture. We consider abstractness a diverse set of relations that subsume the part-of-whole relation, and thus also generally includes the subset relation. For our purposes, defining this notion is not necessary, since we simply wish to utilize the fact of its existence. We understand abstractness to be a semantic concept that shows itself by producing a partial ordering over semantic states (their “abstractness” level) which is in turn reflected in the lower-order semantics of the input-output boxes, and ultimately is reflected in our syntax via explanations of different granularity. Discussions of representative formalisms most relevant to computer science can be found in [17, 38, 48, 49, 72, 74]: [17] features abstraction in verification, [74] features abstraction at play in interpreting programs, [72]

is an excellent example of interfaces providing a notion of abstractness in network communications, [48, 49] discuss notions of abstractness relevant for type systems in object-oriented programming languages, and [38] shows an adaptive application in reinforcement learning. An excellent discussion of the philosophical underpinnings and extensions can be found in [26].

In this work, the primary method of producing explanations at desired levels of abstraction is entirely implicit, without explicitly tracking what boxes or predicates are considered more or less abstract (note that an operator that attempts to learn such relations is invoked optionally by human users, and is not used in the evaluations we present here). Instead, we leverage the groundedness of our predicates to naturally form partial orderings over semantic states (their “abstractness” level) which in turn are appropriately reflected in syntax.

On the opposite end of the spectrum is explicit expert tuning of abstraction orderings. Fanoos can easily be adapted to leverage expert labels (e.g., taxonomies as in [71], or even simply type/grouping-labels without explicit hierarchical information) to preference subsets of predicates conditionally on user responses, but for the sake of this paper, we reserve agreement with expert labels as an independent metric of performance in our evaluation, prohibiting the free use of such knowledge by the algorithm during testing. As a side benefit, by forgoing direct supervision, we demonstrate that the concept of abstractness is recoverable from the semantics and structure of the problem itself.

3 Fanoos Interaction Example

We present a user interaction example with our system in Fig. 1. Predicate definitions of the example can be found with the code at [7]. In practice, if users want to know more about the operational meaning of predicates (e.g., the exact conditions each tests), open-on-click hyperlinks and hover text showing the relevant content from the domain definition can be added to the user interface.

Limited text is shown on screen until a user requests more, similar in spirit to the Unix `more` command. Auto-complete is triggered by hitting tab, finishing tokens when unambiguous and listing options available in the context. For instance, suggestions and completions for predicates obey restrictions imposed by Table 1 based on the question type specified by the user.

In Fig. 1, we show the user posing two questions on the IDP domain (see Sect. 5). The initial question in Fig. 1(a) asks for which the situations *typically* result in the NN outputting a low torque and high state value estimate (Line 1). In order to produce an answer, Fanoos (Lines 2–3) asks for a preference of initial refinement granularity (given relative to S_I ’s side lengths; ϵ in Eq. (2)), and after the user requests 0.125 (Line 4), lists several potential situations (Lines 5–13). The user wants more details, and so requests a less abstract description (Line 16); Fanoos now responds with 18 more detailed situation descriptions (5 listed in Fig. 1(b), Lines 17–23). In the second question in Fig. 1(c), the user (Line 25) wants to know the circumstances in which the learned component outputs a high torque while its inputs (e.g., sensors) indicate that the first pole has a low

```

1 (Fanoos) when_do_you_usually and(
  outputtorque_low ,
  statevalueestimate_high )?
2 Enter a fraction of the universe box
  length to limit refinement to
  at the beginning.
3 Value must be a positive real
  number less than or equal to
  one.

User requests box length ↓ 0.125 4

5 5 of 6 lines to print shown. Press
  enter to show more. Hit ctrl+C
  or enter letter q to break. Hit
  a to list all.
6 =====
7 //Description:
8 (0.45789160, 0.61440409, 'x Near
  Normal Levels')
9 (0.31030792, 0.51991449, '
  pole2Angle_rateOfChange Near
  Normal Levels')
10 (0.12008841, 0.37943400, '
  pole1Angle_rateOfChange High')
11 (0.06128723, 0.22426058, 'pole2Angle
  Low')
12 (0.02395519, 0.13633780, 'vx Low')a
13 (0.01147175, 0.01359231, 'pole1Angle
  Low')
14 type letter followed by enter key: b
  - break and ask a different
  question,
15 l - less abstract, m - more
  abstract, h - history travel

User requests less abstract,
  continue at (b) ↓ 1 16

17 5 of 18 lines to print shown. Press
  enter to //[...]
18 =====
19 (0.16153820, 0.31093854, 'And(
  endOfPole2_x Near Normal Levels
  , pole1Angle Low,
  pole1Angle_rateOfChange High,
  pole2Angle Near Normal Levels,
  pole2Angle_rateOfChange High, x
  High)')
20 (0.14268581, 0.18653883, 'And(
  endOfPole2_x Near Normal Levels
  , pole1Angle Low,
  pole1Angle_rateOfChange High,
  pole2Angle Near Normal Levels,
  pole2Angle_rateOfChange Near
  Normal Levels, x High)')
21 (0.11771033, 0.12043966, 'And(
  pole1Angle Near Normal Levels,
  pole1Angle_rateOfChange Near
  Normal Levels, pole2Angle High,
  pole2Angle_rateOfChange Low,
  vx Low)')
22 (0.06948142, 0.07269412, 'And(
  pole1Angle High,
  pole1Angle_rateOfChange Near
  Normal Levels,
  pole2Angle_rateOfChange High,
  vx Low, x Near Normal Levels)')
23 (0.04513659, 0.06282974, 'And(
  endOfPole2_x Near Normal Levels
  , pole1Angle Low,
  pole1Angle_rateOfChange High,
  pole2Angle High,
  pole2Angle_rateOfChange Near
  Normal Levels, x High)')q

User break, continue at (c) ↓ b 24

(a) Initial question response, followed by request for less abstract explanation
(b) Less abstract explanation, user satisfied, continues with different question

25 (Fanoos) what_are_the_circumstances_in_which and(
  pole1angle_rateofchange_low_magnitude , outputtorque_high_magnitude )?

Fanoos answers ↓

26 5 of 32 lines to print shown. Press enter to //[...]
27 =====
29 (0.10147897, 0.17831770, 'And(pole1angle_on_the_left, pole2angle_on_the_left,
  pole2angle_rateofchange_low_magnitude)')
30 (0.09885232, 0.16335186, 'And(pole1angle_on_the_left, pole2angle_on_the_left,
  pole2angle_turning_counterclockwise)')
31 (0.07900125, 0.14467123, 'And(pole1angle_on_the_right, pole2angle_on_the_right,
  pole2angle_turning_clockwise)')
32 (0.06693577, 0.12822191, 'And(pole1angle_down, pole2angle_to_right,
  statevalueestimate_very_low)')q
33 type letter followed by enter key: b - break and ask a different question,
34 l - less abstract, m - more abstract, h - history travel

User requests more abstract ↓ m 35

36 3 of 3 lines to print shown.
37 =====
38 (0.44378316, 0.48588134, 'pole2 not near target position')
39 (0.33605014, 0.36551887, 'pole2angle_rateofchange_high_magnitude')
40 (0.22016670, 0.23739381, 'And(pole2angle_to_right, statevalueestimate_very_low)
  ')

```

(c) Next question, initial response, and user request to make more abstract

Fig. 1. Fanoos user session on the inverted double pendulum example

rotational speed; Fanoos finds 32 descriptions (5 listed, Lines 26–34). The user requests a more abstract summary (Line 35), which condenses the explanation down to 3 situations (Lines 36–40). We see that in both cases—the first request for less abstractness, and the second for greater—that the explanations adjusted

as one would expect, both with respect to the verbosity of the descriptions returned and the verbiage used.

Our focus while developing Fanoos has been to ensure that the desired information can be generated. In application, a user-facing front-end can provide a more aesthetically pleasing presentation, and we elaborate options in [8].

4 Related Work and Discussion

Many methods are closely related to XAI, stemming from a diverse body of literature and various application domains, e.g., [3, 4, 9, 18, 35, 41, 63, 70, 83]. Numerous taxonomies of explanation families have been proposed [1, 4, 5, 11, 13, 14, 27, 30, 32, 44, 47, 51, 59, 64, 65, 80], with popular divisions being (i) between explanations that leverage internal mechanics of systems to generate descriptions (decompositional, a.k.a. “introspective”, approaches) versus those that exclusively leverage input-output relations (pedagogical, a.k.a. “rationalization”) [4, 44] (ii) the medium that comprises the explanation (such as with most-predictive-features [63], summaries of internal states via finite-state-machines [45], natural language descriptions [35, 44] or even visual representations [39, 44]), (iii) theoretical criteria for a good explanation (see, for instance, [52]), and (iv) specificity and fidelity of explanation. Of note, the vast majority of XAI methods for ML lack any formal guarantees regarding the correspondence between the explanations and the learned component’s true behavior (e.g., [25]).

Related to our work are approaches to formally analyze neural networks to certify or verify them as well as to compositionally extract rules from them. Techniques related to our inner-loop reachability analysis have been used for stability and reachability analysis in systems that are otherwise hard to analyze analytically, often in the interest of ensuring safety. Reachability analysis for FFNNs based on abstract interpretation domains, interval arithmetic, or set inversion has been used in rule extraction and neural net stability analysis [4, 20, 75, 84] and continues to be relevant, e.g., for verification of MLPs [29, 53, 61], estimating the reachable states of closed-loop systems with MLPs in the loop [88], estimating the domain of validity of NNs [2], and analyzing security of NNs [82]. A similar variety of motivations and applications exist for approaches to NN verification and rule extraction that are based on symbolic decomposition of a network’s units followed by constraint solving or optimization over the formulas extracted [12, 21–23, 40, 41, 57, 68, 69, 73, 76, 77, 87]. While these works provide methods to extract descriptions that faithfully reflect behavior of the network, they do not generally consider end-user comprehension of descriptions, do not consider varying description abstraction, and do not explore the practice of strengthening descriptions by ignoring the effects of measure-zero sets. Also, many such techniques are only designed to characterize output behavior given particular input sets, whereas we capture relations in multiple directions (i.e., input to output, output to input, and both simultaneously).

Rule-based systems such as expert systems, and work in the (high-level) planning community have a long history of producing explanations in various

forms. Notably, hierarchical planning [35, 54] naturally lends itself to explanations of multiple abstraction levels. All these methods, however, canonically work on the symbolic level, making them inapplicable to most modern ML methods. High fidelity, comprehensible rules describing data points can also be discovered with weakly-consistent inductive logic programming [56] or association rule learning [3, 37] typical in data-mining. However, these approaches are typically pedagogical—not designed to leverage access to the internals of the system—do not offer a variety of descriptions abstractions or strengths, and are typically not interactive. While extensions of association rule learning (e.g., [33, 34, 71]) do consider multiple abstraction levels, they are still pedagogical and non-interactive. Further, they describe only subsets of the analyzed data⁵ and only understand abstractness syntactically, requiring complete taxonomies be provided explicitly and up-front. Our approach, by contrast, leverages semantic information, attempts to efficiently describe all relevant data instances, and produces descriptions that necessarily reflect the mechanism under study.

The high-level components of our approach can be compared to [36], where hand-tunable rule-based methods with natural language interfaces encapsulate a module responsible for extracting information about the ML system, with explanation generation in part relying on minimal set-covering methods to find predicates capturing the model’s states. Extending this approach to generate more varying-resolution descriptions, however, does not seem like a trivial endeavor, since (i) it is not clear that the system can appropriately handle predicates that are not logically independent, and expecting experts to explicitly know and encode all possible dependencies can be unrealistic, (ii) the system described does not have a method to vary the type of explanation provided for a given query when its initial response is unsatisfactory, and (iii) the method produces explanations by first learning simpler models via Markov decision processes (MDPs). Learning simpler models by sampling behavior of more sophisticated models is an often-utilized, widely applicable method to bootstrap human understanding (e.g. [11, 31, 45]), but it comes at the cost of failing to leverage substantial information from the internals of the targeted learned system. Crucially, such a technique cannot guarantee the fidelity of their explanations with respect to the learned system being explained, in contrast to our approach.

In [60], the authors develop vocabularies and circumstance-specific human models to determine the parameters of the desired levels of abstraction, specificity and location in robot-provided explanations about the robot’s specific, previous experiences in terms of trajectories in a specific environment, as opposed to the more generally applicable conditional explanations about the internals of the learned component generated by Fanoos. The particular notions of abstraction and granularity from multiple, distinct, unmixable vocabularies of [60] evaluate explanations in the context of their specific application and are not immediately applicable nor easily transferable to other domains. Fanoos, by contrast, does

⁵ Setting thresholds low enough to ensure each transaction is described would result in a deluge of highly redundant, low-precision rules lacking most practical value, a phenomena know as the “rare itemset problem” [50].

not require separate vocabularies and enables descriptions to include multiple abstraction levels (for example, mixing them as in the sentence “House X and a 6m large patch on house Y both need to be painted”).

Closest in spirit to our work are the planning-related explanations of [70], providing multiple levels of abstraction with a user-in-the-loop refinement process, but with a focus on markedly different search spaces, models of human interaction, algorithms for description generation and extraction, and experiments. Further, we attempt to tackle the difficult problem of extracting high-level symbolic knowledge from systems where such concepts are not natively embedded, in contrast to [70], who consider purely symbolic systems.

In summary, current approaches focus on single aspects of explanations, fixed levels of abstraction, or provide inflexible guarantees (if any) about the explanations given.

5 Experiments and Results

We analyze learned systems from robotics control and more traditional ML predictors to demonstrate the applicability of Fanoos to diverse domains. Code and other supporting information (e.g., predicate definitions) can be found in [7] and at <https://github.com/DBay-ani/Fanoos>.

Inverted Double Pendulum (IDP). The control policy for an inverted double-pendulum is tasked to keep a pole steady and upright. The pole consists of two under-actuated segments attached end-to-end, rotationally free in the same plane; the only actuated component is a cart with the pivot point of the lower segment attached. Even though similar to the basic inverted single pendulum example in control, this setting is substantially more complicated, since multi-pendulum systems are known to exhibit chaotic behavior [43,46]. The trained policy was taken from reinforcement learning literature [62,67]. The seven-dimensional observation space includes the segment’s angles, the cart x-position, their time derivatives, and the y-coordinate of the second pole. The output is a torque in $[-1, 1]$ Nm and a state-value estimate, which is not a priori bounded. The values chosen for the input space bounding box were inspired by the 5% and 95% quantile values over simulated runs. We expanded the input box beyond this range to consider rare inputs and observations the model was not necessarily trained on; whether the analysis stays in the region trained-for depends on the user’s question. For instance, the train and test environments exited whenever the end of the second segment was below a certain height. In real applications, users may want to ensure recovery is attempted.

CPU Usage (CPU). We also analyze a more traditional ML algorithm, a polynomial kernel regression for modeling CPU usage. Specifically, we use a three-degree fully polynomial basis over a 5-dimensional input space (which includes cross-terms and the zero-degree element—e.g., x^2y and 1 are members) to linearly regress out a three-dimensional vector. We trained our model

using the publicly available data from [78].⁶ The observations are [*lread*, *scall*, *sread*, *freemem*, *freeswap*], which are normalized with respect to the training set min and max prior to featurization, and the response variables we predict are [*lwrite*, *swrite*, *usr*]. We opted to analyze an algorithm with this featurization since it achieved the highest performance—over 90% accuracy—on a 90%-10% train-test split of the data compared to similar models with 1, 2, or 4 degree kernels. While the kernel weights may be interpreted in some sense (e.g., indicating which individual feature is, by itself, most influential), the joint correlation between the features and non-linear transformations of the input values makes it far from clear how the model behaves over the original input space. For Fanoos, the input space bounding box was determined from the 5% and 95% quantiles for each input variable over the full, normalized dataset.

5.1 Experiment Design

Tests were conducted using synthetically generated interactions, with the goal of determining whether our approach properly changes the description abstractness in response to the user request. The domain and question type were randomly chosen, the latter selected among the options listed in Table 1. The questions themselves were randomly generated to have up to four disjuncts, each with conjuncts of length no more than four; conjuncts were ensured to be distinct, and only predicates respecting the constraints of the question type were used. After posing an initial question, interaction with Fanoos was randomly selected from four alternatives (here, MA means “more abstract” and LA means “less abstract”): (i *or* ii) initial refinement of 0.25 *or* 0.20 → make LA → make MA → exit; (iii *or* iv) initial refinement of 0.125 *or* 0.10 → make MA → make LA → exit. For the results presented here, over 130 interactions were held, resulting in several hundred question-answer-descriptions.

5.2 Metrics

We evaluated the abstractness of Fanoos’s responses using metrics in the categories of reachability analysis, description structure, and human word labeling.

Reachability Analysis. We compare the reachability analysis results produced during the interactions: we record statistics about the distribution of volumes of input-boxes generated during the abstract state refinement, normalized to the input space bounding box so that each axis is in $[0, 1]$, yielding results comparable across domains. The values provide a rough sense of the abstractness notion implicit in the size of boxes and how they relate to descriptions. For brevity, we only report volume, but we note that the distribution of sum-of-side-lengths showed similar trends.

⁶ Dataset at <https://www.openml.org/api/v1/json/data/562>.

Description Structure. Fanoos responds to users with a multi-weighted DNF description. This structure is summarized as follows to give a rough sense of how specific each description is by itself: number of disjuncts, including atomic predicates; number of non-singleton conjuncts, providing a rough measure of the number of “complex” terms; number of distinct named predicates (atomic, user-defined predicates that occur anywhere in the description, i.e., excludes box-range predicates); number of box-range predicates that occur anywhere (i.e., in conjuncts as well as stand-alone). The Jaccard score and overlap coefficients—classic text analysis measures—are calculated over the set of atomic predicates in the descriptions to measure verbiage similarity.

Human Word Labeling. We apply our intuitive, human understanding of the relative abstractness of the atomic predicates to evaluate Fanoos’s responses based on usage of more vs. less abstract verbiage. For simplicity we choose two classes, more abstract (MA) vs. less abstract (LA), and count the number of predicates both (a) accounting for multiplicity and, (b) accounting for uniqueness; if an atomic predicate q has label MA (resp., LA) and occurs twice in a sentence, it contributes twice to the (a) score, and only once to (b).

5.3 Results

Summary statistics of our results are listed in Table 2. We are chiefly interested in how a description changes in response to a user-requested abstraction change. Specifically, for pre-interaction state S_t and post-interaction state S_{t+1} , we collect metrics $m(S_{t+1}) - m(S_t)$ that describe *relative* change for each domain-response combination (for the Jaccard and overlap coefficients, the computation is simply $m(S_{t+1}, S_t)$). The medians of these distributions are in Table 2.

In summary, the reachability and structural metrics follow the desired trends: when the user requests greater abstraction (MA), the boxes become larger, and the sentences become structurally less complex—namely, they become shorter (fewer disjuncts), have disjuncts that are less complicated (fewer explicit conjuncts, hence more atomic predicates), use fewer unique terms overall (reduction in named predicates) and refer less often to the exact values of a box (reduction in box-range predicates). Symmetric statements can be made for less abstraction (LA) requests. From the overlap and Jaccard scores, we can see that the changes in response complexity are not simply due to increased verbosity—simply adding or removing phrases to the descriptions from the prior steps—but also the result of changes in the verbiage used.

Trends for the human word-labels are similar, though more subtle. We see that use of LA-related terms follows the trend of user requests with respect to multiplicity and uniqueness counts (increases for LA-requests, decreases for MA-requests). We see that the MA counts, when taken relative to the same measures for LA terms, are correlated with user requests in the expected fashion. Specifically, when a user requests greater abstraction (MA), the counts for LA terms decrease far more than those of MA terms, and the symmetric situation occurs

Table 2. Median *relative* change in description before and after Fanoos adjusts the abstraction in the requested direction

			CPU	CPU	IDP	IDP
		Request	LA	MA	LA	MA
Reachability	Boxes	Number	8417.5	-8678.0	2.0	-16.0
	Volume	Max	-0.015	0.015	-0.004	0.004
		Median	-0.003	0.003	-0.004	0.004
		Min	-0.001	0.001	-0.003	0.003
		Sum	-0.03	0.03	-0.168	0.166
Structural	Jaccard		0.106	0.211	0.056	0.056
	Overlap coefficient		0.5	0.714	0.25	0.25
	Non-singleton conjuncts		1.0	-2.0	0.5	-2.5
	Disjuncts		7.0	-7.5	2.0	-2.5
	Named predicates		1.0	-1.0	1.0	-4.5
	Box-range predicates		2.0	-2.0	1.5	-1.5
Words	MA terms	Multiplicity	3.0	-3.0	24.0	-20.0
		Uniqueness	0.0	0.0	1.0	-1.5
	LA terms	Multiplicity	20.0	-21.5	68.5	-86.0
		Uniqueness	2.0	-2.0	12.0	-14.0

for requests of lower abstraction (LA), as expected. These results—labelings coupled with the structural trends—lend solid support that Fanoos can recover elements of a human’s notion about abstractness by leveraging the grounded semantics of the predicates.

6 Conclusions and Future Work

Fanoos is an explanatory framework for ML systems that mixes technologies ranging from classical verification to heuristic search. Our experiments support that Fanoos can produce and navigate explanations at multiple granularities and strengths. We are investigating operator-selection learning and accelerating knowledge base construction via further data-driven predicate generation.

We will continue to explore Fanoos’s potential, and hope that the community finds inspiration in both the methodology and philosophical underpinnings presented here. Additional content, such as pseudo-code, summary statistics, extended descriptions and further pointers, can be found in [8].

Acknowledgments. We thank: Nicholay Topin for supporting our spirits at some key junctures of this work; David Held for pointing us to the rl-baselines-zoo repository; David Eckhardt for his proof-reading of earlier versions of this document; the anonymous reviewers for their thoughtful feedback.

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE Access* **6**, 52138–52160 (2018)
2. Adam, S.P., Karras, D.A., Magoulas, G.D., Vrahatis, M.N.: Reliable estimation of a neural network’s domain of validity through interval analysis based inversion. In: 2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, 12–17 July 2015, pp. 1–8 (2015). <https://doi.org/10.1109/IJCNN.2015.7280794>
3. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, 26–28 May 1993, vol. 22, pp. 207–216. ACM (1993)
4. Andrews, R., Diederich, J., Tickle, A.: Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowl.-Based Syst.* **6**, 373–389 (1995). [https://doi.org/10.1016/0950-7051\(96\)81920-4](https://doi.org/10.1016/0950-7051(96)81920-4)
5. Anjomshoae, S., Najjar, A., Calvaresi, D., Främling, K.: Explainable agents and robots: results from a systematic literature review. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, pp. 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems (2019)
6. Arya, V., et al.: One explanation does not fit all: a toolkit and taxonomy of AI explainability techniques. CoRR abs/1909.03012 (2019)
7. Bayani, D.: Code for Fanoos: multi-resolution, multi-strength, interactive explanations for learned systems (2021). <https://doi.org/10.5281/zenodo.5513079>. Method of distribution is Zenodo, distributed in October, 2021
8. Bayani, D., Mitsch, S.: Fanoos: multi-resolution, multi-strength, interactive explanations for learned systems. CoRR abs/2006.12453 (2020)
9. Benz, A., Jäger, G., Van Rooij, R.: *Game Theory and Pragmatics*. Springer, Heidelberg (2005). <https://doi.org/10.1057/9780230285897>
10. Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y., et al.: Bounded model checking. *Adv. Comput.* **58**(11), 117–148 (2003)
11. Biran, O., Cotton, C.: Explanation and justification in machine learning: a survey. In: IJCAI-17 Workshop on Explainable AI (XAI), vol. 8, p. 1 (2017)
12. Bunel, R., Turkaslan, I., Torr, P.H.S., Kohli, P., Mudigonda, P.K.: A unified view of piecewise linear neural network verification. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3–8 December 2018, Montréal, Canada*, pp. 4795–4804 (2018)
13. Chakraborti, T., Kulkarni, A., Sreedharan, S., Smith, D.E., Kambhampati, S.: Explicability? Legibility? Predictability? Transparency? Privacy? Security? The emerging landscape of interpretable agent behavior. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 29, pp. 86–96 (2019)
14. Chuang, J., Ramage, D., Manning, C., Heer, J.: Interpretation and trust: designing model-driven visualizations for text analysis. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 443–452. ACM (2012)

15. Clarke, E., Fehnker, A., Han, Z., Krogh, B., Stursberg, O., Theobald, M.: Verification of hybrid systems based on counterexample-guided abstraction refinement. In: Garavel, H., Hatcliff, J. (eds.) TACAS 2003. LNCS, vol. 2619, pp. 192–207. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36577-X_14
16. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 154–169. Springer, Heidelberg (2000). https://doi.org/10.1007/10722167_15
17. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, pp. 238–252 (1977)
18. David, Q.: Design issues in adaptive control. *IEEE Trans. Autom. Control* **33**(1), 50–58 (1988)
19. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
20. Driescher, A., Korn, U.: Checking stability of neural NARX models: an interval approach. *IFAC Proc. Vol.* **30**(6), 1005–1010 (1997)
21. Dvijotham, K., Stanforth, R., Gowal, S., Mann, T.A., Kohli, P.: A dual approach to scalable verification of deep networks. In: Globerson, A., Silva, R. (eds.) Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, 6–10 August 2018, pp. 550–559. AUAI Press (2018)
22. Ehlers, R.: Formal verification of piece-wise linear feed-forward neural networks. In: D’Souza, D., Narayan Kumar, K. (eds.) ATVA 2017. LNCS, vol. 10482, pp. 269–286. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68167-2_19
23. Etchells, T.A., Lisboa, P.J.: Orthogonal search-based rule extraction (OSRE) for trained neural networks: a practical and efficient approach. *IEEE Trans. Neural Netw.* **17**(2), 374–384 (2006)
24. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (General Data Protection Regulation) (2016)
25. Fern, A.: Don’t get fooled by explanations. Invited Talk, IJCAI-XAI (2020). Recording at: <https://ijcai20.org/w41/>. Schedule at: <https://sites.google.com/view/xai2020/home>
26. Floridi, L.: The method of levels of abstraction. *Mind. Mach.* **18**(3), 303–329 (2008)
27. Friedrich, G., Zanker, M.: A taxonomy for generating explanations in recommender systems. *AI Mag.* **32**(3), 90–98 (2011)
28. Garcia, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **16**(1), 1437–1480 (2015)
29. Gehr, T., Mirman, M., Drachler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.T.: AI2: safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21–23 May 2018, San Francisco, California, USA, pp. 3–18. IEEE Computer Society (2018). <https://doi.org/10.1109/SP.2018.00058>
30. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. *ACM Comput. Surv. (CSUR)* **51**(5), 93 (2019)

31. Gunning, D., Aha, D.: DARPA's explainable artificial intelligence (XAI) program. *AI Mag.* **40**(2), 44–58 (2019). <https://doi.org/10.1609/aimag.v40i2.2850>
32. Hailesilassie, T.: Rule extraction algorithm for deep neural networks: a review. arXiv preprint [arXiv:1610.05267](https://arxiv.org/abs/1610.05267) (2016)
33. Han, J., Fu, Y.: Discovery of multiple-level association rules from large databases. In: *VLDB*, vol. 95, pp. 420–431. Citeseer (1995)
34. Han, J., Fu, Y.: Mining multiple-level association rules in large databases. *IEEE Trans. Knowl. Data Eng.* **11**(5), 798–805 (1999)
35. Hayes, B., Scassellati, B.: Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 5469–5476. IEEE (2016)
36. Hayes, B., Shah, J.A.: Improving robot controller transparency through autonomous policy explanation. In: 2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 303–312. IEEE (2017)
37. Hipp, J., Güntzer, U., Nakhaeizadeh, G.: Algorithms for association rule mining—a general survey and comparison. *SIGKDD Explor.* **2**(1), 58–64 (2000)
38. Hostetler, J., Fern, A., Dietterich, T.G.: Progressive abstraction refinement for sparse sampling. In: Meila, M., Heskes, T. (eds.) *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015, 12–16 July 2015, Amsterdam, The Netherlands*, pp. 365–374. AUAU Press (2015)
39. Huang, S.H., Held, D., Abbeel, P., Dragan, A.D.: Enabling robots to communicate their objectives. *Auton. Robot.* **43**(2), 309–326 (2019). <https://doi.org/10.1007/s10514-018-9771-0>
40. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: Majumdar, R., Kunčák, V. (eds.) *CAV 2017. LNCS*, vol. 10426, pp. 3–29. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_1
41. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: an efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kunčák, V. (eds.) *CAV 2017, Part I. LNCS*, vol. 10426, pp. 97–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_5
42. Kearfott, R.B.: Interval computations: introduction, uses, and resources. *Euromath Bull.* **2**(1), 95–112 (1996)
43. Kellert, S.H.: *In the Wake of Chaos: Unpredictable Order in Dynamical Systems*. University of Chicago Press (1993)
44. Kim, J., Rohrbach, A., Darrell, T., Canny, J.F., Akata, Z.: Textual explanations for self-driving vehicles (2018). https://doi.org/10.1007/978-3-030-01216-8_35
45. Koul, A., Fern, A., Greydanus, S.: Learning finite state representations of recurrent policy networks. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019. OpenReview.net (2019)
46. Levien, R., Tan, S.: Double pendulum: an experiment in chaos. *Am. J. Phys.* **61**(11), 1038–1044 (1993)
47. Lipton, Z.C.: The mythos of model interpretability. arXiv preprint [arXiv:1606.03490](https://arxiv.org/abs/1606.03490) (2016)
48. Liskov, B.: Keynote address—data abstraction and hierarchy. In: *Addendum to the Proceedings on Object-Oriented Programming Systems, Languages and Applications (Addendum)*, pp. 17–34 (1987)
49. Liskov, B.H., Wing, J.M.: A behavioral notion of subtyping. *ACM Trans. Program. Lang. Syst. (TOPLAS)* **16**(6), 1811–1841 (1994)
50. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 337–341 (1999)

51. Miller, T.: Explanation in artificial intelligence: insights from the social sciences. *Artif. Intell.* **267**, 1–38 (2018)
52. Miller, T., Howe, P., Sonenberg, L.: Explainable AI: beware of inmates running the asylum or: how I learnt to stop worrying and love the social and behavioural sciences. arXiv preprint [arXiv:1712.00547](https://arxiv.org/abs/1712.00547) (2017)
53. Mirman, M., Gehr, T., Vechev, M.: Differentiable abstract interpretation for provably robust neural networks. In: International Conference on Machine Learning, pp. 3575–3583 (2018)
54. Mohseni-Kabir, A., Rich, C., Chernova, S., Sidner, C.L., Miller, D.: Interactive hierarchical task learning from a single demonstration. In: Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction, pp. 205–212. ACM (2015)
55. Moore, R.E.: Interval Analysis, vol. 4. Prentice-Hall, Englewood Cliffs (1966)
56. Muggleton, S.: Inductive logic programming: issues, results and the challenge of learning language in logic. *Artif. Intell.* **114**(1–2), 283–296 (1999)
57. Murdoch, W.J., Szlam, A.: Automatic rule extraction from long short term memory networks. arXiv preprint [arXiv:1702.02540](https://arxiv.org/abs/1702.02540) (2017)
58. Neema, S.: Assured autonomy (2017). https://www.darpa.mil/attachments/AssuredAutonomyProposersDay_Program%20Brief.pdf
59. Papadimitriou, A., Symeonidis, P., Manolopoulos, Y.: A generalized taxonomy of explanations styles for traditional and social recommender systems. *Data Min. Knowl. Disc.* **24**(3), 555–583 (2012)
60. Perera, V., Selvaraj, S.P., Rosenthal, S., Veloso, M.M.: Dynamic generation and refinement of robot verbalization. In: 25th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2016, New York, NY, USA, 26–31 August 2016, pp. 212–218. IEEE (2016). <https://doi.org/10.1109/ROMAN.2016.7745133>
61. Pulina, L., Tacchella, A.: An abstraction-refinement approach to verification of artificial neural networks. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 243–257. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_24
62. Raffin, A.: RL baselines zoo (2018). <https://web.archive.org/web/20190524144858/https://github.com/araffin/rl-baselines-zoo>
63. Ribeiro, M.T., Singh, S., Guestrin, C.: “Why should I trust you?”: explaining the predictions of any classifier. In: Krishnapuram, B., Shah, M., Smola, A.J., Aggarwal, C.C., Shen, D., Rastogi, R. (eds.) Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016, pp. 1135–1144. ACM (2016). <https://doi.org/10.1145/2939672.2939778>
64. Richardson, A., Rosenfeld, A.: A survey of interpretability and explainability in human-agent systems. In: XAI Workshop on Explainable Artificial Intelligence, pp. 137–143 (2018)
65. Roberts, M., et al.: What was I planning to do. In: ICAPS Workshop on Explainable Planning, pp. 58–66 (2018)
66. Rosenthal, S., Biswas, J., Veloso, M.: An effective personal mobile robot agent through symbiotic human-robot interaction. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 915–922. International Foundation for Autonomous Agents and Multiagent Systems (2010)
67. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) (2017)

68. Setiono, R., Liu, H.: Understanding neural networks via rule extraction. In: IJCAI, vol. 1, pp. 480–485 (1995)
69. Singh, G., Ganvir, R., Püschel, M., Vechev, M.T.: Beyond the single neuron convex barrier for neural network certification. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, 8–14 December 2019, Vancouver, BC, Canada, pp. 15072–15083 (2019)
70. Sreedharan, S., Madhusoodanan, M.P., Srivastava, S., Kambhampati, S.: Plan explanation through search in an abstract model space. In: *International Conference on Automated Planning and Scheduling (ICAPS) Workshop on Explainable Planning*, pp. 67–75 (2018)
71. Srikant, R., Agrawal, R.: Mining generalized association rules. In: Dayal, U., Gray, P.M.D., Nishio, S. (eds.) *VLDB 1995, Proceedings of 21st International Conference on Very Large Data Bases*, 11–15 September 1995, Zurich, Switzerland, pp. 407–419. Morgan Kaufmann (1995)
72. International Standardization: ISO/IEC 7498–1: 1994 information technology-open systems interconnection-basic reference model: the basic model. *International Standard ISO/IEC 74981*, 59 (1996)
73. Taylor, B.J., Darrah, M.A.: Rule extraction as a formal method for the verification and validation of neural networks. In: *2005 Proceedings of 2005 IEEE International Joint Conference on Neural Networks*, vol. 5, pp. 2915–2920. IEEE (2005)
74. Tennent, R.D.: The denotational semantics of programming languages. *Commun. ACM* **19**(8), 437–453 (1976). <https://doi.org/10.1145/360303.360308>
75. Thrun, S.: Extracting rules from artificial neural networks with distributed representations. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (eds.) *1994 Advances in Neural Information Processing Systems 7, NIPS Conference*, Denver, Colorado, USA, pp. 505–512. MIT Press (1994)
76. Tjeng, V., Xiao, K., Tedrake, R.: Verifying neural networks with mixed integer programming. *CoRR* abs/1711.07356 (2017). <http://arxiv.org/abs/1711.07356>
77. Towell, G.G., Shavlik, J.W.: Extracting refined rules from knowledge-based neural networks. *Mach. Learn.* **13**(1), 71–101 (1993)
78. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. *SIGKDD Explor.* **15**(2), 49–60 (2013). <https://doi.org/10.1145/2641190.2641198>
79. Veloso, M.M., Biswas, J., Coltin, B., Rosenthal, S.: CoBots: robust symbiotic autonomous mobile service robots. In: IJCAI, p. 4423 (2015)
80. Ventocilla, E., et al.: Towards a taxonomy for interpretable and interactive machine learning. In: *XAI Workshop on Explainable Artificial Intelligence*, pp. 151–157 (2018)
81. Walter, E., Jaulin, L.: Guaranteed characterization of stability domains via set inversion. *IEEE Trans. Autom. Control* **39**(4), 886–889 (1994). <https://doi.org/10.1109/9.286277>
82. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. In: *27th USENIX Security Symposium, USENIX Security 2018*, Baltimore, MD, USA, 15–17 August 2018, pp. 1599–1614 (2018)
83. Wellman, H.M., Lagattuta, K.H.: Theory of mind for learning and teaching: the nature and role of explanation. *Cogn. Dev.* **19**(4), 479–497 (2004)

84. Wen, W., Callahan, J.: Neuralware engineering: develop verifiable ANN-based systems. In: Proceedings IEEE International Joint Symposia on Intelligence and Systems, pp. 60–66. IEEE (1996)
85. Wen, W., Callahan, J., Napolitano, M.: Towards developing verifiable neural network controller. Technical report (1996)
86. Wen, W., Callahan, J., Napolitano, M.: Verifying stability of dynamic soft-computing systems. Technical report NASA-IVV-97-002, WVU-CS-TR-97-005, NASA/CR-97-207032, WVU-IVV-97-002 (1997)
87. Weng, L., et al.: Towards fast computation of certified robustness for ReLU networks. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 5276–5285. PMLR, 10–15 July 2018
88. Xiang, W., Johnson, T.T.: Reachability analysis and safety verification for neural network control systems. CoRR abs/1805.09944 (2018)
89. Yasmin, M., Sharif, M., Mohsin, S.: Neural networks in medical imaging applications: a survey. World Appl. Sci. J. **22**(1), 85–96 (2013)