NeuroCERIL: Robotic Imitation Learning via Hierarchical Cause-Effect Reasoning in Programmable Attractor Neural Networks

Gregory P. Davis · Garrett E. Katz · Rodolphe J. Gentili · James A. Reggia

Accepted: 21 March 2023 / Published online: 21 April 2023
© The Author(s), under exclusive licence to Springer Nature B.V. 2023

Abstract
Imitation learning allows social robots to learn new skills from human teachers without substantial manual programming, but it is difficult for robotic imitation learning systems to generalize demonstrated skills as well as human learners do. Contemporary neurocomputational approaches to imitation learning achieve limited generalization at the cost of data-intensive training, and often produce opaque models that are difficult to understand and debug. In this study, we explore the viability of developing purely-neural controllers for social robots that learn to imitate by reasoning about the underlying intentions of demonstrated behaviors. We present a novel hypothetico-deductive reasoning algorithm that combines bottom-up abductive inference with top-down predictive verification and captures important aspects of human causal reasoning that are relevant to a broad range of cognitive domains. We also present NeuroCERIL, a neurocognitive architecture that implements this algorithm using only neural computations, and produces generalizable and human-readable explanations for demonstrated behavior. Our empirical results demonstrate that NeuroCERIL can learn various procedural skills in a simulated robotic imitation learning domain. We also show that its causal reasoning procedure is computationally efficient, and that its memory use is dominated by highly transient short-term memories, much like human working memory. We conclude that NeuroCERIL is a viable neural model of human-like imitation learning that can improve human-robot collaboration and contribute to investigations of the neurocomputational basis of human cognition.

Keywords Imitation learning · Causal reasoning · Programmable neural networks · Cognitive control · Working memory · Symbolic processing

1 Introduction
Humans readily teach and learn using demonstration and imitation. The ability to imitate emerges at an early age and plays a crucial role in early human development, but remains a natural and intuitive method for acquiring new skills throughout the lifespan [1, 2]. Crucially, human-level imitation involves not only replicating observable motor behavior, but also inferring the underlying goals and intentions of the demonstrator. This allows learners to generalize demonstrated skills to novel environments by abstracting away details that are circumstantial to the demonstration environment.

Programming social robots to carry out complex tasks in a human-like fashion is difficult and typically requires laborious programming by an experienced roboticist. One promising solution to this problem is to develop robots that learn from demonstrations (i.e., robotic imitation learning) [3–6]. This eases the burden of robotic programming, making it accessible to non-experts. However, most work in robotic imitation learning focuses on reproducing overt motor activity, which affords only limited generalization [3]. Developing more human-like imitation in robots requires algorithms for reasoning about observed actions to construct a deeper under-
standing of the demonstrator’s goals and intentions that can be adapted to novel environments. This approach also provides a common framework for reasoning about human and robot behavior, which facilitates an understanding of roles and perspectives that promotes seamless human-robot collaboration [7].

Inferring intentions during imitation learning can be viewed as a process of causal reasoning, in which observable behaviors are treated as the effects of hidden causal intentions. Reasoning backwards from effects to possible causes is known as abductive inference, and is a crucial aspect of human diagnostic reasoning and general problem-solving. Algorithms for causal imitation learning may therefore be broadly relevant to cognitive domains beyond overt motor planning, such as language comprehension and visual scene understanding. In addition, insight into the neural basis of these cognitive processes may be gained through the development of robotic imitation learning systems that respect the constraints and neurobiological foundations of human cognition.

In previous work, we developed CERIL, a robotic imitation learning system that uses abductive inference to construct causal interpretations of demonstrated motor behavior and generalizes them for imitation in novel environments [8]. While effective and provably correct, CERIL’s algorithms are implemented with traditional non-neural symbolic programming and have a limited degree of cognitive plausibility. Specifically, CERIL’s inference algorithm involves exhaustive enumeration of plausible causal explanations, which places unrealistic demands on working memory. It also processes demonstrations in an offline fashion, rather than iteratively as humans do.

In this paper, we present NeuroCERIL, a purely-neural imitation learning system that reproduces CERIL’s ability to explain demonstrated behavior during imitation learning. NeuroCERIL is a programmable neural network that implements a novel causal inference algorithm based on the hypothetico-deductive approach, combining bottom-up abductive inference with top-down deductive prediction and verification. Notably, NeuroCERIL processes demonstrations in an online fashion by iteratively constructing efficient data structures in memory that can be used to generate plausible explanations for observed behavior. In other words, NeuroCERIL’s cognitive processes are much more human-like than CERIL’s, and they are supported by neuro-computational mechanisms that more closely resemble those used by people during cause-effect reasoning.

Our empirical results show that NeuroCERIL is potentially an effective neurocognitive controller for robotic imitation learning systems, as it is able to reproduce CERIL’s performance on a battery of demonstrations of procedural maintenance tasks. We examine NeuroCERIL’s runtime and memory usage during causal inference and show that they scale roughly linearly with the length of the demonstration. Further, many of its memories have very short lifetimes, and are only accessed during a narrow window of processing. Thus, like human working memory, many of its short-term memories are rapidly abandoned, and only a small fraction of its memories need to be maintained through the duration of a demonstration.

2 Related Work

In imitation learning, or learning from demonstration, an agent learns new skills by observing a teacher’s demonstrations. While imitation can be as simple as reproducing motor trajectories, humans are capable of a higher form of imitation that involves reasoning about a teacher’s goals and intentions. This form of imitation, which we refer to as “cognitive-level” imitation, allows learners to grasp the underlying purpose of the demonstrated behaviors and generalize them to novel circumstances. Although its origin remains unclear, cognitive-level imitation emerges in early childhood and plays a crucial role in human cognitive development [1, 9–12]. Imitation is thought to be supported by neural mechanisms that establish shared representations for perceptually observable behavior and cognitive-motor control processes (i.e., the mirror neuron system), facilitating perspective-taking and interpersonal collaboration [13–16].

Robotic imitation learning has been proposed as a solution to the complexity and limited accessibility of robotic programming, and has been approached in a variety of ways [3–6, 17, 18]. Much past work has focused on low-level tasks such as manual coordination and navigation in simple environments [19–23], often taking the form of behavioral cloning, an approach that is limited by its focus on replicating motor control. In contrast, inverse reinforcement learning approaches model demonstrated behavior as the output of a goal-directed process, and attempt to infer the value function of the demonstrator and use it to guide subsequent planning during imitation [24–26]. This focus on demonstrator goals improves generalization, but it remains a significant challenge to develop systems that can reason about learned behavior and adapt it to situations that deviate from the demonstration environment. Furthermore, contemporary imitation learning systems that rely on reinforcement learning and other machine learning techniques require substantial training data and are opaque to users, which makes it difficult to diagnose and debug errors, and creates barriers in trustworthiness and explainability. Safe and effective robotic imitation requires human-like algorithms for understanding demonstrated actions, adapting learned skills to novel environments, and constructing explanations of planned behavior that can be understood by end-users.
In our view, these challenges suggest an emphasis on hierarchical and causal reasoning. Although research in these areas is limited, recent work indicates that exploiting the hierarchical structure of behavior [27–31] and the causal structure of the environment [32–34] leads to significant improvements in the performance and efficiency of imitation learning systems. The combination of these skills in robotic imitation learning is largely unexplored. In previous work, we developed CERIL, a robotic imitation learning system based on hierarchical cause-effect reasoning (Fig. 1) [8, 35].

The basic intuition behind this approach is that demonstrated motor behavior is caused by hidden intentions or goals that must be inferred by the learner. To do so, CERIL reasons backwards from effects to plausible causes, a process known as abductive inference, to construct a hierarchy of cause-effect relations that explains the demonstration in terms of high-level intentions (Fig. 1, arrow A). Because these intentions are abstracted from the concrete demonstration, they can be used during imitation to plan a new sequence of motor behavior that implements the learned skill in a novel environment (Fig. 1, arrows B and C). This approach permits generalization across variations in the imitation environment that demand different low-level implementations of the same high-level intention. For example, opening a door may require different motor sequences depending on if it has a rotating handle, whether it must be pushed or pulled, and if it must be unlocked first. An imitator that knows about these variations can recognize the abstract intention to open a door in a demonstration and generalize it to imitation environments with different doors. In addition, CERIL can use these plans to provide explanations for its motor behavior, which allows end-users to investigate and debug its understanding of demonstrated skills. Thus, CERIL uses cause-effect reasoning to understand demonstrated behavior, generalize it to new environments, and explain its own behavior to human users.

As noted above, CERIL is effective and provably correct, but its algorithms are implemented with traditional non-neural programming techniques that have a limited degree of cognitive plausibility. To infer intentions, CERIL uses a bottom-up dynamic programming algorithm that exhaustively enumerates plausible causal explanations, which places unrealistic demands on working memory. It also requires multiple passes through a demonstration, whereas human imitators reason about demonstrated behavior as it occurs to construct partial explanations before a demonstration is complete. Finally, it is unclear how this approach might be implemented using neural networks to leverage the unique advantages of neural computation, such as its capacity for learning and generalization, and provide insight into the neurobiological foundations of human imitation learning.

Neural approaches to imitation learning typically involve deep neural networks, which require data-intensive training and exhibit a limited degree of generalization in constrained environments [27, 28, 36, 37]. It remains an outstanding challenge to develop neural networks with the high-level cognitive abilities that CERIL exhibits, such as causal inference, compositional modeling, and logical reasoning. Many neural models are incorporated into hybrid systems that delegate these abilities to non-neural algorithms, such as neural-guided search and program synthesis [38–40]. Furthermore, these models often rely on neurobiologically implausible processes that require simultaneous access to a temporal sequence of inputs or activity states, such as temporal convolutions and attention.

In recent work, we have developed neurocognitive systems that learn to represent and evaluate symbolic programs (i.e., programmable neural networks) [41–43]. Most recently, we developed NeuroLISP, a programmable neural network with a compositional working memory that can learn to evaluate programs written in LISP, a language with an extensive history in artificial intelligence research [44]. Several features make NeuroLISP an attractive option for modeling human-level cognition in a neurobiologically plausible manner. NeuroLISP can learn to perform high-level cognitive tasks that are difficult for contemporary deep neural networks, such as compositional sequence manipulation, tree traversals, and symbolic pattern matching. Importantly, it learns these tasks using fast associative learning rules that establish robust algorithmic behavior using minimal training data. Its architecture is program-independent, and it represents programs and data using learned attractor states in recurrent neural regions (i.e., distributed representations) that are controlled by top-down gating of both learning and activation. Finally, programmable neural networks can be integrated with neural models of sensorimotor control, including the complex motor control required in robotic imitation learning [45, 46].

In this paper, we explore whether it is viable to develop purely-neural controllers for social robotic systems that behave in a human-like manner. To this end, we develop NeuroCERIL, a programmable neural network that learns human-like algorithms for causal inference during imitation learning (left side of Fig. 1). We evaluate our model using CERIL as a target system, as it has been demonstrated to be an effective cognitive controller for bimanual robots. Our specific contributions are as follows. First, we introduce a novel causal inference algorithm (Sect. 3.2) based on the hypothetico-deductive approach, an influential model of diagnostic and scientific reasoning [47–51]. Hypothetico-deductive reasoning involves a combination of bottom-up abductive inference and top-down predictive verification, which obviates the need for exhaustive search by focusing cognitive processing on relevant causal knowledge, and addresses CERIL’s limitations in cognitive plausibility. Second, we introduce a neurocognitive architecture (Sect. 3.3)
that realizes this algorithm using only neural computations. This architecture is based on NeuroLISP, but introduces two major innovations that show how high-level programming constructs can improve the efficiency and computational capabilities of programmable neural networks, allowing them to learn complex cognitive behaviors: (1) a class system for constructing typed objects with named attributes, and (2) an exception handling system for responding to errors encountered during program evaluation. Overall, NeuroCERIL serves as a purely-neural controller component for social robots that more closely resembles human cognition and learning, facilitates more seamless human-robot interactions, and provides a neurocomputational framework for modeling cognitive processes that are relevant to a broad range of application domains.

3 Methods

NeuroCERIL\(^1\) is a brain-inspired cognitive model that learns procedural skills from demonstrations using cause-effect reasoning. The model’s architecture is an extension of NeuroLISP, a programmable neural network that can store and evaluate programs written in a subset of the Common LISP programming language [44]. NeuroCERIL is programmed with a novel causal inference algorithm based on hypothetico-deductive reasoning, which combines bottom-up abductive inference with top-down deductive prediction and verification. This approach allows NeuroCERIL to anticipate future behavior and focuses cognitive computations on plausible explanations for observed behavior.

Although NeuroCERIL is implemented using attractor neural networks, its distributed neural computations represent algorithmic procedures performed on symbolic data structures. It is therefore convenient to begin by describing its behavior in terms of symbolic information processing. We first outline the robotic imitation learning domain in which NeuroCERIL operates (Sect. 3.1), and the algorithms and data structures that it uses to implement hypothetico-deductive causal inference (Sect. 3.2). Then, we present the neurocognitive architecture that learns to represent and evaluate these algorithms and data structures using only neural computations (Sect. 3.3). Finally, we describe the empirical experiments that we conducted to validate NeuroCERIL, including a battery of test demonstrations that was used to validate CERIL (Sect. 3.4). We show that NeuroCERIL performs comparably to CERIL, but that its iterative hypothetico-deductive approach is memory efficient and scales well to long demonstrations.

3.1 Robotic Imitation Learning Domain

NeuroCERIL operates in the robotic imitation learning domain designed for CERIL, which involves a bimanual robot (Baxter) learning procedural maintenance tasks [8]. As previously mentioned, a teacher demonstrates these tasks using SMILE, a simulated 3D environment that allows users to interact with virtual objects such as blocks, drawers,
switches, and screw valves \[52\]. SMILE also includes a simulation of the Baxter robot, shown in Fig. 2 with a variety of simulated objects. SMILE greatly simplifies the low-level sensory processing involved in recognizing and segmenting actions and objects, allowing us to focus on the higher level cognitive processing that occurs during imitation. When a user is finished recording a demonstration, SMILE produces a transcript containing the sequence of recorded actions, along with a record of changes that occur in the environment, such as changes in object state or location.

Actions are encoded as discrete events with free parameters that refer to objects or locations in the environment. For example, grasping a red-block with the left-gripper is encoded as:

\[
\text{grasp<red-block, left-gripper>}
\]

The identifier left-gripper refers to the demonstrator’s left hand, and red-block refers to an object in the environment, which is encoded as a collection of named properties:

\[
\{\text{id:red-block, type:block, color:red, location:loc}\}
\]

Once the block is grasped, its location property is updated to left-gripper to indicate that it is currently located in the demonstrator’s left hand. This change is represented as a record containing the object identifier, property name, and the new property value:

(\text{red-block location left-gripper})

Once the block is moved and placed, this property is updated again to reflect its new location. Although locations are encoded as discrete symbols, they can be associated with representations of 3D points in continuous space for use in low-level motor planning.

Like CERIL, NeuroCERIL is pre-programmed with a knowledge-base of cause-effect relations that describe the implementation of abstract intentions. We represent intentions as compound planning operators, in the technical sense of hierarchical task network planning \[53\]. A compound operator is a high-level routine that takes the current environment state as input, and recursively decomposes into state-dependent sequences of sub-routines (themselves compound operators) and eventually observable actions (i.e., primitive operators). Domain authors typically design compound operators with target goal states in mind, although that is not formally required. NeuroCERIL focuses on the structure of observed actions, and uses its causal relations to infer a demonstrator’s intentions (on left side of arrow) from their actions (right side of arrow). For example, the intention to relocate an object (obj) to a target location (loc) causes a sequence of concrete motor actions: grasp
the object, move it to a target location, and release the grasp. This is encoded as a template or schema that can be matched to observed behavior:

\[
\text{relocate} \langle \text{obj}, \text{loc} \rangle \rightarrow \\
\text{grasp} \langle \text{obj}, \text{gripper} \rangle, \\
\text{move} \langle \text{gripper}, \text{loc} \rangle, \\
\text{release} \langle \text{gripper} \rangle
\]

Here, the right arrow represents causation, and indicates that the intention on the left side of the arrow can cause the ordered sequence of actions on the right side. It is important that parameter names (\text{obj}, \text{loc}, \text{gripper}) are repeated in this schema, because this indicates correspondences between the parameters of the intention and the actions that it causes (e.g., the same gripper is used for each action). NeuroCERIL verifies these correspondences when it infers casual intentions in a demonstration. In addition, each schema may include explicit logical predicates that must be satisfied for a cause-effect relation to be plausible. For example, the intention to open a drawer may cause a sequence of grasping, moving, and releasing, but the grasped object must be a drawer handle, and the drawer must be closed prior to opening. These constraints can be encoded as logical statements that NeuroCERIL must verify while inferring causal intentions.

The effects of a causal intention may include other abstract intentions, allowing causes to be chained together to create hierarchies of cause-effect relations. For example, the intention to swap the location of two objects may be implemented as a sequence of \text{relocate} intentions:

\[
\text{swap} \langle \text{obj1}, \text{obj2} \rangle \rightarrow \\
\text{relocate} \langle \text{obj1}, \text{temp} \rangle, \\
\text{relocate} \langle \text{obj2}, \text{loc1} \rangle, \\
\text{relocate} \langle \text{obj1}, \text{loc2} \rangle
\]

A concrete demonstration of this swapping behavior would involve a sequence of \text{grasp}, \text{move}, and \text{release} actions that are caused by intermediate \text{relocate} intentions. Thus, inferring the causes of demonstrated actions requires a recursive inference process: when an intention is recognized as a plausible cause, it is treated as the effect of plausible higher-level causal intentions.

NeuroCERIL’s causal knowledge-base may contain multiple schemas describing different implementations of the same intention. For example, the location of two objects may be swapped without placing one in an intermediate location, by instead keeping one object in hand while relocating the other:

\[
\text{swap} \langle \text{obj1}, \text{obj2} \rangle \rightarrow \\
\text{grasp} \langle \text{obj1}, \text{gripper} \rangle, \\
\text{move} \langle \text{gripper}, \text{temp} \rangle, \\
\text{relocate} \langle \text{obj2}, \text{loc1} \rangle,
\]

\[
\text{move} \langle \text{gripper}, \text{loc2} \rangle, \\
\text{release} \langle \text{gripper} \rangle
\]

Here, \text{temp} refers to a location in the air to which the demonstrator lifts \text{obj1} to, holding it there while \text{obj2} is relocated to the original location of \text{obj1} (\text{loc1}). A key feature of NeuroCERIL’s knowledge-base is that causal relations are agnostic to the implementation of their effects: a higher-level intention that is implemented using \text{swap} does not specify which implementation of \text{swap} to use. This flexibility affords generalization during imitation; a demonstration involving one implementation of \text{swap} can be imitated using the other implementation. Thus, causal inference allows the imitator to abstract away circumstantial details of the demonstration environment and adapt learned skills to novel circumstances. This may also be necessary if the embodiment of the imitator differs from that of the demonstrator (e.g., number of arms, dexterity, range of motion), requiring the imitator to implement learned skills in a different but equivalent way.

Finally, a sequence of demonstrated actions may have more than one plausible explanation. This may occur if two sequences of cause-effect relations share the same sequence of effects. For example, given the following three cause-effect relations (shown without parameters for simplicity):

\[
X \rightarrow A, B \\
Y \rightarrow C \\
Z \rightarrow A, B, C
\]

a sequence of actions \((A, B, C)\) may be caused by the sequence of intentions \((X, Y)\), or the single intention \(Z\). In this case, the most parsimonious (i.e., simplest or shortest) explanation is usually preferred: \((A, B, C)\) was caused by \(Z\).

In the next subsection, we describe the new hypothetico-deductive causal inference algorithm that NeuroCERIL uses to identify the most parsimonious explanation for a demonstration. NeuroCERIL is provided with a pre-programmed knowledge-base of cause-effect relations with optional logical constraints, as described above. The initial state of the virtual environment is provided as a list of objects encoded as collections of named properties, which may change during the demonstration (e.g., location). The demonstration is encoded as a sequence of LISP-like expressions encoding parameterized actions, each paired with a list of changes that occur to objects in the environment. The output of this algorithm is an expression encoding the causal hierarchy relating observed actions to inferred intentions, which serves as an explanation of the demonstration as well as an encoding of the demonstrated skill. For example, given a knowledge-base that describes how to relocate an object (by grasping it, moving it to a target location, and releasing it) and discard it (by relocating it to a discard bin), a simple demonstra-
tion of discarding a block (block1) with the right gripper (right_gripper) would involve the following inputs and outputs:

Inputs:
- ((grasp right_gripper block1) (right_gripper gripping block1))
- (move right_gripper discard-bin)
- ((release right_gripper) (right_gripper gripping none))

Output:
- ((discard block1) (relocate block1 discard-bin) (grasp right_gripper block1) (move right_gripper discard-bin) (release right_gripper))

The output’s nested structure shows the hierarchical organization of the causal explanation: the top-level intention to discard block1 causes the intention to discard block1 to the discard-bin, which in turn causes the observed sequence of grasp, move, and release actions.

### 3.2 Hypothetico-Deductive Causal Inference

NeuroCERIL’s approach to causal inference differs from CERIL’s in a way that is more cognitively plausible and memory efficient. Whereas CERIL conducts an exhaustive bottom-up search that makes multiple passes through an entire demonstration, NeuroCERIL uses a more human-like hypothetico-deductive approach that involves a combination of bottom-up and top-down reasoning to iteratively construct a causal explanation for a demonstration as it occurs. When an action is observed, NeuroCERIL consults its cause-effect knowledge-base to generate explicit hypotheses about the demonstrator’s causal intentions (bottom-up), and uses them to deduce testable predictions about subsequent actions (top-down). NeuroCERIL’s cognitive processing is focused on evaluating these predictions to verify or falsify hypotheses. By organizing active hypotheses based on their predictions, NeuroCERIL can efficiently access those that are relevant to an observation, and avoid considering those that are not.

When all of the predictions of a hypothesis are verified by observations, the hypothesized causal intention is treated as an observation and processed recursively to identify plausible higher-level intentions that may have caused it. In this way, NeuroCERIL constructs hierarchies of cause-effect relations that are supported by observations, and that represent plausible explanations for sequences of demonstrated behavior. As plausible intentions are identified, NeuroCERIL updates parsimony pointers that indicate the shortest sequence of intentions that covers the actions observed so far. At the end of the demonstration, these pointers are traced back to identify the most parsimonious explanation for the entire demonstration. This process is illustrated in Fig. 3, outlined as pseudocode in Algorithm 1, and described in more detail below.

NeuroCERIL uses several different data structures to keep track of observed actions, their relative timing, and hypotheses about their causal explanations. These data structures are organized around a timeline, represented in memory as a chain of discrete time-points that delimit observed actions (circles connected by solid arrows in Fig. 3). Each action contains pointers to the timepoints immediately before and after it (i.e., start and end points). For concrete primitive actions that are directly observed (e.g., grasping and releasing), these timepoints are adjacent in the timeline (Action: A in Fig. 3a). However, inferred high-level causal intentions can be implemented with multiple lower-level actions, and can therefore span several timepoints (Intention: X in Fig. 3c).
Hypotheses originate from a bottom-up abductive reasoning process that we call evocation; when an action/intention is observed, NeuroCERIL consults its causal knowledge-base to identify relevant cause-effect schemas that might explain it (top right of Fig. 3a, c, and first loop of PROCESS_ACTION procedure in Algorithm 1). These schemas are stored in an associative array that maps each action/intention type to a list of schemas that predict it as their first effect. For example, the knowledge-base may contain a schema describing a cause-effect relation between the relocate intention and a sequence of grasp, move, and release actions. This schema is stored in the grasp list of the knowledge-base, and can be retrieved to evoke a hypothesis that an observed grasp action was caused by the intention to relocate the grasped object. This hypothesis must be evaluated to determine if the observed action satisfies the constraints of the schema, including correspondences between parameters with shared names as well as explicit logical predicates that must be true for the causal relation to be plausible (VERIFY_HYPOTHESIS procedure in Algorithm 1).

| Algorithm 1 Pseudocode for hypothetico-deductive causal inference algorithm |
|---------------------------------------------------------------|
| **procedure** EXPLAIN(demo, init_env)                          |
| curr_env ← init_env                                           |
| prev_time ← create timepoint with init_env                    |
| for each action and record of environment changes in demo do   |
| curr_env ← new environment with changes chained off prev_time |
| curr_time ← new timepoint with curr_env chained off prev_time |
| set action start and end timepoints to prev_time and curr_time |
| PROCESS_ACTION(action)                                         |
| prev_time ← curr_time                                          |
| end for                                                        |
| return TRACE(curr_time)                                        |
| end procedure                                                  |
| **procedure** PROCESS_ACTION(action)                           |
| if action has shorter path to initial timepoint then            |
| update parsimony pointer for action’s end timepoint            |
| end if                                                         |
| for each schema predicting action type as first effect do      |
| hypothesis ← generate hypothesis from schema                   |
| VERIFY_HYPOTHESIS(hypothesis, action)                         |
| end for                                                        |
| for each hypothesis predicting action type at action end timepoint do |
| VERIFY_HYPOTHESIS(hypothesis, action)                         |
| end for                                                        |
| end procedure                                                  |
| **procedure** VERIFY_HYPOTHESIS(hypothesis, action)             |
| if action matches hypothesis prediction then                   |
| if hypothesis is fully matched then                            |
| intent ← generate causal intention from hypothesis             |
| PROCESS_ACTION(intent)                                         |
| else                                                          |
| update hypothesis prediction                                   |
| add hypothesis to action’s end timepoint                       |
| end if                                                        |
| end if                                                        |
| end procedure                                                  |
| **procedure** TRACE(curr_time)                                 |
| intent ← parsimony pointer of curr_time                       |
| prev_time ← start timepoint of intent                         |
| if prev_time is initial timepoint then                         |
| return list containing intent                                  |
| else                                                          |
| prior_intents ← TRACE(prev_time)                              |
| append intents to prior_intents                               |
| return prior_intents                                          |
| end if                                                        |
| end procedure                                                  |
Corresponding parameters are matched with a symbolic pattern matching procedure (unification) that we have previously implemented using neural computations [44]. If these constraints are not satisfied, the hypothesis is immediately abandoned. Otherwise, it is added to the timeline and used to make predictions about subsequent actions, as described below. In Fig. 3a, the knowledge-base contains two schemas indicating causal relations that might explain the observed action of type A (top right). The evoked hypotheses predict a subsequent action of type B₁ and B₂, respectively (bottom right).

Each timepoint contains a set of hypotheses that make predictions about actions or causal intentions that might occur immediately after it (boxes labeled “Hyp”, bottom halves of Fig. 3a–c). Like cause-effect schemas in the knowledge-base, these hypotheses are stored in an associative array that maps the predicted action/intention type to the hypotheses that predict it at that timepoint. When an action/intention is observed, the hypothesis set for its starting timepoint is consulted to retrieve the hypotheses that predicted it (second loop of PROCESS_ACTION procedure in Algorithm 1). For the relocate example above, the evoked hypothesis predicts that the demonstrator will move the grasping arm immediately after the grasp action occurred. When move is observed, this hypothesis is retrieved and evaluated to determine if its prediction was satisfied. This involves verifying the schema’s logical constraints, as described above. If these constraints are satisfied and the hypothesis predicts further actions, the next prediction is retrieved, and the hypothesis is advanced to the next timepoint. When all of the predictions for a hypothesis are verified, it is used to generate a plausible causal intention that is added to the timeline. This intention is then processed recursively in order to generate and verify further hypotheses about its underlying cause (call to PROCESS_ACTION procedure within VERIFY_HYPOTHESIS procedure of Algorithm 1). In Fig. 3b, the observed action of type B₁ matches the prediction of a hypothesis at t₁, and the corresponding causal intention of type X is generated (bottom right). This intention is then processed as an observation in Fig. 3c, and a new hypothesis is evoked proposing that an intention of type Z is the underlying cause. This new hypothesis predicts an intention of type Y at time t₂, and is added to the timeline accordingly (bottom right).

Each timepoint also contains a representation of the state of the environment that is consulted during hypothesis verification (Fig. 4). The environment contains several objects with named properties that can change over the course of a demonstration. At the beginning of the demonstration, NeuroCERIL is provided with a full specification of the initial state of the environment (left side of Fig. 4), stored as a nested associative array mapping symbolic names of objects to sets of object properties (e.g., the location of drive₁ is initially loc₁). When an action is observed, NeuroCERIL is also provided with a list of changes to object properties that were caused by the action. Rather than maintaining full copies of the environment state at each timepoint, which would require substantial memory, NeuroCERIL stores a record of these changes that can be consulted to determine the state of the environment at a given timepoint (“Change₁” and “Change₂”, center and right of Fig. 4). To query the state of an object property at a given timepoint, NeuroCERIL retrieves the most recent change that occurred to that property prior to that
NeuroCERIL’s neurocognitive architecture that learns to perform hypothetico-deductive causal inference. This architecture is an extension of NeuroLISP [44], and is made up of several recurrent neural regions (boxes) with inter-regional connections (solid arrows) that are divided into sub-networks (grey background boxes). Like NeuroLISP, NeuroCERIL implements an interpreter for a LISP-like programming language that is used to implement high-level algorithms.

To support this operation, each timepoint stores a nested associative array, where the outer array stores entries for changed objects, and each inner array stores entries for a particular object’s changed properties. Importantly, the inner arrays representing changes to the same object at different timepoints are chained together to allow an efficient search for the most recent change to a specific property (dotted lines in Fig. 4). This compact representation uses minimal memory, but affords access to the state of the environment at each timepoint.

Finally, NeuroCERIL maintains pointers in memory that can be used to retrieve the most parsimonious explanation for the actions observed so far. The best explanation is the shortest sequence of intentions that covers all directly observed primitive actions without gaps or overlaps. This is represented by a chain of alternating timepoints and intentions that leads from the last timepoint to the first timepoint. Thus, each timepoint maintains a parsimony pointer to the intention that provides the shortest path back to the first timepoint in the demonstration. Whenever a plausible intention is identified, it is compared with the current best intention for the intention’s end timepoint (beginning of PROCESS_ACTION procedure in Algorithm 1). NeuroCERIL performs this comparison by iterating through the paths simultaneously until the initial timepoint is reached. If the newly identified intention provides a shorter path to the initial timepoint, it is replaced as the current best intention for the end timepoint. When the demonstration is complete, the best explanation for the full sequence of observed actions can be reconstructed by following the chain of parsimony pointers from the final timepoint back to the initial timepoint (TRACE procedure in Algorithm 1).

### 3.3 Neural Implementation

NeuroCERIL’s architecture (shown in Fig. 5) is an extension of NeuroLISP, a programmable neural network that learns to store and evaluate programs written in a subset of the Common LISP programming language. As mentioned in Sect. 2, this approach offers a number of advantages for neurobiologically plausible cognitive modeling, including the ability to learn robust algorithms that operate on compositional data structures, such as those described in Sect. 3.2, fast learning from minimal training data, and ready integration with neural models of sensory and motor processing. Many of the details of NeuroCERIL’s functionality are shared with NeuroLISP and can be found in [44]. Here we provide a brief overview and highlight the novel features of NeuroCERIL’s...
architecture that extend its computational capabilities beyond NeuroLISP.

Like NeuroLISP, NeuroCERIL represents programs and other symbolic data structures as learned systems of dynamical attractor states and associative transitions between attractor states [41]. Programs and other data are stored as structured associative memories [41] in the mem region (center of Fig. 5). These memories are retrieved during program evaluation and used to orchestrate top-down control of gated connectivity between and within neural regions (regional gating, bottom left of Fig. 5). This guides the flow of activity according to instructions retrieved from neural memory, much like a conventional computer architecture controls data flow according to instruction opcodes. Importantly, NeuroCERIL also controls its own learning in this way, allowing it to construct, access, and modify data structures stored in memory during program evaluation. Inputs and outputs are mediated by gated connectivity between the outer environment and a special region that represents discrete symbols as unique patterns of activity (\texttt{lex}, center of Fig. 5). These symbolic representations can be associated with programs and data structures stored in the mem region via connectivity between mem and \texttt{lex}. These connections allow NeuroCERIL to read symbolic inputs, including representations of programs, and output the results of program evaluation. During imitation learning, a demonstration recorded in SMILE is provided as a sequence of symbolic inputs, and NeuroCERIL outputs a sequence of symbolic outputs that encodes the inferred causal explanation.

Dynamics in NeuroCERIL’s neural regions are governed by a common set of mathematical formulas described briefly below. Neurons in a region \( r \) receive several inputs based on gating signals that determine when each input source is active during model execution:

\[
s_r(t) = \sum_{q, \ell} g_{r,q}^{\text{context}}(t) W_{r,q}(t) v_q(t) + g_{r}^{\text{bias}}(t) b_r + g_{r}^{\text{noise}}(t) n_r(t) + g_{r}^{\text{read}}(t) l_r(t) + g_{r}^{\text{saturate}}(t) (v_r(t)) \sigma_r^{-1}(v_r(t))
\]

\[
x_r(t) = \prod_{q} \left\{ \begin{array}{ll} v_q(t) > 0, & \text{if } g_{r,q}^{\text{context}}(t) = 1 \\ 1, & \text{otherwise} \end{array} \right.
\]

\[
v_r(t + 1) = \sigma_r(x_r(t) \odot s_r(t))
\]

where \( v_r(t) \) is a vector of neural activity in region \( r \) at time \( t \), \( \sigma_r \) is the activation function of region \( r \) (typically sign/sgn), and \( x_r(t) \) is an optional vector of multiplicative inputs to region \( r \) at time \( t \). Multiplicative inputs are aggregated from the activity state \( v_q(t) \) of each region \( q \) when the corresponding gate is dynamically enabled (\( g_{r,q}^{\text{context}}(t) = 1 \)), and provide a mechanism for context-dependent associative learning and memory retrieval for representations of compositional data structures [41]. \( s_r(t) \) is a vector of cumulative synaptic inputs to region \( r \) aggregated from several sources:

- inputs from connected regions (solid lines with arrow heads in Fig. 5).
- random noise. When \( g_{r}^{\text{noise}}(t) = 1 \), a vector of random inputs \( n_r(t) \) with density \( \lambda_r \) [41] is generated in region \( r \).
- external inputs. When \( g_{r}^{\text{read}}(t) = 1 \), region \( r \) “reads” an input pattern \( l_r(t) \) from the external environment.
- activity maintenance. When \( g_{r}^{\text{saturate}}(t) = 1 \), activity \( v_r(t) \) in region \( r \) is cycled back into the region’s inputs to maintain it over time. \( \sigma_r^{-1} \) is the inverse of the region’s activation function.

Learning in the model is also controlled by regional gating, allowing for online construction of new memories during program execution. Pathway-specific weight matrices are updated via a combination of Hebbian and anti-Hebbian learning [42, 44]:

\[
\Delta W_{r,q}(t) = g_{r,q}^{\text{learn}}(t) \left( \frac{1}{v_q(t)} \right)^{\text{learning gate}} \left( v_r(t) - (x_r(t) \odot W_{r,q}(t) v_q(t)) \right) v_q(t)^T
\]

where \( W_{r,q}(t) \) is the weight matrix for connection \( \ell \) from region \( q \) to region \( r \) at time \( t \), \( v_r(t) \) and \( v_q(t) \) are the current activity patterns in regions \( r \) and \( q \) at time \( t \) (Eq. 3), and \( \sigma_r^{-1} \) is the inverse of the activation function for region \( r \). Weight updates are distributed across the weight matrix and are normalized according to the magnitude of the source pattern. The weight matrix is only updated during model execution when \( g_{r,q}^{\text{learn}}(t) = 1 \), a regional gating signal computed during program execution.

This learning rule is also used to initialize NeuroCERIL with a learned program-independent virtual machine composed of procedures that implement the primitive operations of its programming language [42], and to program NeuroCERIL with the causal inference algorithm described in
Sect. 3.2, which is expressed in the language of NeuroCERIL’s virtual machine. The details of initialization and program learning can be found in [44].

NeuroCERIL’s virtual machine supports two major innovations that extend its computational capabilities beyond NeuroLISP and ease the implementation of its causal inference algorithm: a class system and an exception handling system. The class system allows specification of reusable programs (i.e., class methods) for initializing and modifying instances of complex data structures such as causal hypotheses, cause-effect knowledge, and observed actions. Instances of classes, called objects, are stored as collections of named pointers to other memories (i.e., class attributes). The underlying implementation of objects makes use of the existing mechanisms for variable binding in NeuroCERIL’s virtual machine; objects have corresponding lexical namespaces that store attributes as variable bindings, and can be retrieved via learned associations between the mem and env regions (see [44] for details on variable binding in NeuroLISP).

Exceptions are errors that occur during program evaluation, and are triggered by events such as attempted access to undefined variables, attributes, or class methods. The exception handling system provides a mechanism for specifying dynamic responses to exceptions. This obviates the need for excessive program expressions that perform checks on data before access; a program instead can specify what should be done if retrieval fails. For example, when evoking new hypotheses to explain an observed action, NeuroCERIL consults its causal knowledge-base to retrieve cause-effect recipes that are relevant to the observed action (see Sect. 3.2). Because evocation only matches an action to the first effect of a cause-effect recipe, the observed action may not evoke any new hypotheses even if it can be plausibly matched to the prediction of an existing hypothesis. In this case, the knowledge-base does not contain any entry for the observed action type, and retrieval will result in an exception that can be easily handled by skipping the evocation process (first loop of PROCESS_ACTION in Algorithm 1) and moving directly to prediction verification (second loop of PROCESS_ACTION in Algorithm 1). This situation is depicted in Fig. 3b, in which the observed action (Action : B₁) is matched to a hypothesis (Hyp : [X → A, B₁]), but does not evoke any new hypotheses from the knowledge base.

Exception handling is supported by the exception stack region (bottom right of Fig. 5), which maintains pointers to activity states in other regions that represent the state of the virtual machine. This region functions like the runtime and data stack regions (shared with NeuroLISP), which represent stack frames as distributed patterns of activity that have learned associations with activity patterns in other regions [44]. Responses to exceptions are specified in programs with “try” expressions that include a primary sub-expression to evaluate, and an additional sub-expression representing the error response (i.e., a “catch block”). When a “try” expression is evaluated, the virtual machine first stashes its state on the exception stack, which involves learning associations in the pathways exiting the exception stack region. Then, the virtual machine attempts to evaluate the primary sub-expression. If an exception occurs, the virtual machine retrieves its prior state from the exception stack, and evaluates the response sub-expression. Upon completion, the top of the exception state is popped, and evaluation of the program continues.

3.4 Experimental Evaluation

To evaluate NeuroCERIL, we performed empirical experiments using a battery of test demonstrations that was used to test CERIL. These tests include procedural maintenance tasks involving replacing, swapping, and discarding mock hard drives in a docking assembly, as well as toy block stacking tasks (see [8] for details). For example, the replace red with green tasks involve removing a disk drive from a slot with a red light, and replacing it with a disk drive from a slot with a green light. Prior to testing, NeuroCERIL was initialized during a training process that established in memory its virtual machine, causal inference algorithm, and knowledge-base of cause-effect schemas indicating how intentions can be implemented. Test demonstrations, which were not included during training, were then provided as inputs for a single trial of inference. Each task was provided as a sequence of LISP-like expressions encoding the observed motor actions (e.g., (grasp drive1 left-gripper)), a list of manipulable objects in the environment and their properties, and the changes to these objects that occur after each motor action. Using the hypothetico-deductive algorithm described in Sect. 3.2, NeuroCERIL identified a plausible hierarchical explanation for the input sequence in terms of top-level abstract intentions that may have caused the observed sequence, and output this explanation as a sequence of LISP-like expressions (e.g., (relocate drive1 slot1)). We compared NeuroCERIL’s output for each task with CERIL’s to confirm that it performs comparably, and carried out additional analysis on its memory usage and runtime to determine how well it scales with the length of demonstrations.

Runtime was measured as the number of timesteps in model simulations, and memory usage was evaluated by monitoring each simulation to count the number of associations that were learned during causal inference. Specifically, we monitored learning of attractor states and transitions in the underlying neural networks (stored in the recurrent connectivity of the mem region in Fig. 5), as well as associations between namespaces and memory states that represent variable bindings for both local variables and object attributes.
Table 1  NeuroCERIL performance on battery of robotic imitation learning tasks

| Demonstrated task               | Act | Interp | Timesteps | Attr | Transit | Bindings |
|--------------------------------|-----|--------|-----------|------|---------|----------|
| Remove red drive (1)           | 7   | 3      | 353,825   | 197  | 362     | 661      |
| Remove red drive (2)           | 10  | 4      | 490,085   | 250  | 468     | 917      |
| Replace red with spare (1)     | 14  | 6      | 653,758   | 341  | 642     | 1243     |
| Replace red with spare (2)     | 14  | 6      | 653,758   | 341  | 642     | 1243     |
| Replace red with green (1)     | 15  | 7      | 668,955   | 356  | 670     | 1276     |
| Replace red with green (2)     | 15  | 7      | 668,955   | 356  | 670     | 1276     |
| Swap red with green (1)        | 16  | 8      | 668,889   | 357  | 672     | 1278     |
| Swap red with green (2)        | 16  | 8      | 668,981   | 361  | 680     | 1285     |
| Toy blocks (IL)                | 24  | 8      | 1,224,377 | 591  | 1150    | 2326     |
| Toy blocks (AI)                | 30  | 10     | 1,524,253 | 735  | 1434    | 2905     |
| Toy blocks (UM)                | 39  | 13     | 1,975,927 | 945  | 1848    | 3763     |

(Stored in the connection from env to mem in Fig. 5). These associations represent the core data structures used during causal inference, such as observed actions, hypotheses, and inferred causes. We report the associations formed specifically during the inference process, and exclude those that represent the causal inference programs and cause-effect knowledge that is shared across demonstrations.

We further examined NeuroCERIL’s memory access patterns to gain a better understanding of its memory usage. We hypothesized that the majority of memories constructed during inference would be highly transient memories that are only accessed across brief intervals of time, such as abandoned causal hypotheses. This would indicate that NeuroCERIL might benefit from a functionally distinct short-term memory system in which memories rapidly fade if they are not refreshed by retrieval, much like human working memory. To test this hypothesis, we recorded instances of memory construction and access during inference, excluding demonstration-independent memories such as program representations and cause-effect knowledge. For each recorded memory, we determined its “lifespan” as the interval between its initial learning and the final time it was retrieved during the simulation (i.e., a memory is “born” when it is first learned, and “dies” after its last retrieval during the simulation). We then calculated the number of “living” memories over the course of the inference process and compared it to the total number of memories constructed. This provides a metric for the proportion of memories that are being actively utilized for causal inference.

4 Results

Table 1 shows the results for the same benchmark battery of procedural maintenance task demonstrations that were used to verify CERIL’s functionality. For each task, we report the number of actions recorded in the demonstration (Act), the number of top-level intentions in NeuroCERIL’s causal interpretation (Interp), the number of timesteps of neural network simulation required for causal inference (Timesteps), and three measurements of learned associations that indicate model memory usage: the number of learned attractors (Attr) and attractor transitions (Transit) in the mem region, and the number of learned variable bindings (Bindings). NeuroCERIL produced causal interpretations (sequences of top-level intentions) equivalent to the minimum cardinality explanations identified by CERIL for each of the tests.

Figure 6 shows an example of the causal interpretation inferred for the replace red with spare (1) task, which involves replacing a broken disk cartridge (cart2) in a mock disk drive drawer with a fresh cartridge (cart5). Actions and causal intentions are represented by rectangles that indicate the type of action/intention along with its parameters. Each action/intention points to its start and end timepoints, represented by circles (left side), which delineate concrete observed actions (leftmost column of boxes). The top-level explanation, composed mostly of abstract intentions, is indicated by bold boxes. NeuroCERIL reconstructs this explanation by following the shortest path from the final \((t_f)\) to initial \((t_i)\) timepoints using parsimony pointers (bold arrows, shown only for relevant timepoints; see Sect. 3.2).

Runtime and memory usage results provide an empirical estimate of the complexity of NeuroCERIL’s hypothetico-deductive causal inference algorithm. Figure 7 shows runtime and memory usage\(^2\) relative to the length of input demonstrations (Act in Table 1). Each datapoint corresponds to an individual imitation learning test task (rows in Table 1), and the dashed lines show the results of linear regression computed for each metric. These results can be compared to Table 1 in [8],\(^3\) as well as the theoretical analysis of

\(^2\) The model was tested on a GPU accelerated desktop computer, which completed one million timesteps of model execution in \(~88\) min using \(~20.5\) GB of GPU memory.

\(^3\) We used slightly more complex versions of the IL and AI block stacking tasks that include more blocks and actions than those reported in [8].
Fig. 6 Causal interpretation produced by NeuroCERIL for a procedural task involving replacement of a broken disk cartridge (cart2) in a mock disk drive drawer with a fresh cartridge (cart5). NeuroCERIL infers a hierarchy of abstract intentions linking observed concrete actions (left column of rectangles) to a compact high-level explanation (bold rectangles and arrows)
NeuroCERIL’s memory usage and runtime scale linearly with the number of observed actions in the demonstrated procedural task. Unlike CERIL, NeuroCERIL’s memory usage and runtime scale linearly with the length of the demonstration (x-axis). This is due to its online processing of demonstrations and incremental updating of data structures in memory that implicitly represent possible explanations.

In further analysis of memory usage, we focus on learned memory attractors, as they are a bottleneck for neural attractor memory [41]. We present the results for the replace red with spare (1) demonstration (shown in Fig. 6), but note that the results for other demonstrations are comparable. Figure 8a shows the “lifespans” of memory attractors constructed during causal inference for this demonstration, computed as the interval between initial learning and final retrieval. The x-axis indexes timesteps in which a memory attractor is constructed or retrieved, and each horizontal line indicates the lifespan of one memory attractor, indexed along the y-axis. Shorter lines indicate that a memory attractor is only accessed over a brief interval, while longer lines indicate memories that are utilized over longer periods of time. Some memories remain alive through the majority of the inference process, such as representations of the environment and inferred causes that make up the final top-level cover, while others have relatively short lifespans, such as falsified causal hypotheses. We refer to the “living memories” at a given timestep as the set of memories that have been learned prior to that timestep, and that will be accessed at a later timestep (i.e., a memory “dies” after the final timestep in which it is accessed). Figure 8b shows the total number of memory attractors learned over the course of the infer-
ence process, along with the number of “living” memories at each timestep, which corresponds to the number of overlapping horizontal lines at each point along the x-axis in Fig. 8a. Although the total number of learned memories increases steadily over time, the majority of these memories have relatively short lifespans and are rapidly abandoned. As a result, the number of “living” memories remains fairly stable over time, and never exceeds 20% of the total learned memories.

5 Discussion

In this paper, we presented NeuroCERIL, a brain-inspired neurocognitive controller for social robots that learn procedural tasks from human-provided demonstrations (i.e., robotic imitation learning). NeuroCERIL infers the intentions underlying demonstrated behavior using a novel causal inference algorithm based on human-like hypothetico-deductive reasoning, which combines bottom-up abductive inference with top-down predictive verification. This approach allows NeuroCERIL to iteratively construct plausible interpretations of demonstrated behavior as it is observed, make verifiable predictions about subsequent behavior, and generate compact explanations in terms of abstract intentions. By abstracting away the low-level implementation details of a demonstration, NeuroCERIL’s explanations can be generalized for planning alternative implementations that depend on variations in the environment. We evaluated NeuroCERIL on a benchmark battery of procedural maintenance and toy block-stacking tasks recorded in a virtual environment, demonstrating that it works effectively in robotic imitation learning domains. Our empirical results complement prior work demonstrating that cause-effect reasoning is an effective and provably correct approach to imitation learning [8] and we show that comparable performance can be achieved using only neural computations. In addition, our empirical results show that the model scales well with the length of demonstrated action sequences, and that the majority of its memory usage during causal inference is dedicated to transient short-term memories, much like human working memory. This demonstrates that causal inference can be achieved more efficiently with the use of hypothetico-deductive reasoning, which avoids exhaustive enumeration of potential solutions by focusing computations on viable partial solutions.

NeuroCERIL is distinguished from prior approaches to robotic imitation learning by its use of neural computations to understand demonstrated behavior in terms of causal relations that are directly related to high-level planning and cognitive-motor control. This not only affords generalization during imitation, but also facilitates an understanding of roles and perspectives that is critical to human-robot collaboration [7]. For example, NeuroCERIL’s inference algorithm may be generalized for understanding of human behavior during collaborative tasks, in which a robot must reason about behavioral plans that include components carried out by multiple agents. Such a system may be able to collaborate with humans more effectively by adapting its own intentions to those inferred about human participants (e.g., adjusting plans to avoid interfering with collaborators or help them achieve their goals). This is made more feasible by NeuroCERIL’s general data structures for representing hierarchical behavioral plans and algorithms for both deductive and abductive reasoning. In addition, NeuroCERIL maintains a model of the external environment in memory and tracks changes that are induced by demonstrated motor activity. NeuroCERIL’s understanding of demonstrations therefore provides an awareness of the physical consequences of behavior that is critical for safe and effective deployment of robots in sensitive environments.

Causal reasoning and compositionality are widely considered to be critical components of human cognition that are challenging for contemporary neural models to learn [54–57]. NeuroCERIL performs causal reasoning with compositional models in working memory that represent the external environment and encode high-level behavioral plans, and is therefore a significant step toward developing neural networks with human-like reasoning capabilities. In addition, we have previously proposed that neural models of working memory control, particularly in humanoid robots, provide a promising avenue to understanding conscious cognitive processing and its underlying basis in neural computations [35, 58]. NeuroCERIL is therefore also relevant to investigations of consciousness in machines and biological agents because it implements human-like cognitive algorithms in a brain-inspired neural architecture.

NeuroCERIL has several important limitations that suggest directions for future research. In this paper, we have focused on the causal inference component of imitation learning (left side of Fig. 1), and have not addressed the perceptual processing of demonstrated behavior or the generation of motor plans to implement learned skills during imitation. Prior work has demonstrated that neural networks can effectively segment actions from raw videos of demonstrated behavior [59–61]. In our own prior work, we have shown that programmable neural networks can implement basic hierarchical planning, and can perform adaptable motor control in simulated robots [41, 45]. It is therefore feasible to integrate NeuroCERIL with low-level neural models of perception and motor control to create a complete neurocognitive imitation learning system that performs the full range of action recognition, causal inference, and motor planning.

Our hypothetico-deductive causal reasoning algorithm relies on a knowledge-base of causal relations that describe how abstract intentions can be decomposed into sequences of increasingly concrete behavior. The algorithm’s capacity
for generalization depends on the richness of this knowledge-base, as it specifies the intentions that can be recognized and their possible implementations. Thus, it cannot be expected to identify unknown intentions, or to identify known intentions from unknown implementations. In the present work, we relied on human-authored domain knowledge that should be acquired through experience in future work. We note that imitation learning is a tractable approach to incrementally developing such domain knowledge, as each demonstration indicates a new causal relation that can be stored in the knowledge-base for subsequent learning. Thus, given only predefined knowledge of elementary motor behaviors (e.g., grasping objects and toggling switches), our causal inference algorithm can be straightforwardly modified to learn causal relations that compose these behaviors into increasingly complex skills such as relocating objects, arranging them in particular ways, and carrying out conditional procedures that depend on environmental conditions. This incremental learning procedure resembles human learning, in which simple skills must be learned as prerequisites to more complex skills (e.g., one must learn to boil water before learning to prepare coffee or tea). By learning different implementations of the same skill, an imitator can improve their ability to generalize higher-level skills to novel environments. Importantly, this has a combinatorial effect: complex skills with multiple sub-components can be implemented in many different ways if each sub-component affords even a small number of different implementations. Future work should address inference of the logical conditions that determine which causal relation should be used during planning to implement each intention (i.e., under what circumstances a particular implementation is appropriate).

Our causal inference algorithm also relies on constraints in demonstrated behavior. In particular, implementations of abstract intentions must be performed in a fixed order as specified in the causal knowledge-base, and cannot be broken up by unrelated actions. In reality, procedural tasks might involve interleaved action sequences performed with both hands, and may include steps that can be performed in arbitrary arrangements. Thus, future work might involve modifying our causal inference algorithm to support these variations. This might also permit generalization to additional cognitive domains in which hypothetico-deductive reasoning is relevant, such as visual scene understanding and linguistic processing.

Finally, NeuroCERIL uses a unified memory system that does not include functionally distinct short-term and long-term memory. This means that long-term memories such as programs and causal knowledge may be gradually degraded as new short-term memories are constructed during program evaluation. Our empirical results show that the majority of memories constructed during causal inference are only accessed during a narrow window of time, and are therefore highly transient short-term memories. This suggests that NeuroCERIL would benefit from a functional separation of short-term and long-term memory to protect the latter from interference.

Acknowledgements This work was supported by ONR award N00014-19-1-2044.

Data Availability The datasets generated during and/or analysed during the current study are available in the NeuroCERIL repository (https://github.com/vicariousgreg/neuroceril), which includes an implementation of the model as well as the encodings of behavioral demonstrations and model outputs generated during testing.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Ethical approval Our submitted work is original and has not been published or submitted elsewhere for review. There are no human/animal subjects or biological data involved in this work, and no confidential information of any kind.

References

1. Jones SS (2009) The development of imitation in infancy. Philos Trans R Soc B Biol Sci 364(1528):2325–2335
2. Melzoff AN, Kuhl PK, Movellan J, Sejnowski TJ (2009) Foundations for a new science of learning. Science 325(5938):284–288
3. Ravichandran H, Polydoros AS, Chernova S, Billard A (2020) Recent advances in robot learning from demonstration. Ann Rev Control Robot Autonom Syst 3:297–330
4. Hussein A, Gaber MM, Elyan E, Jayne C (2017) Imitation learning: a survey of learning methods. ACM Comput Surv (CSUR) 50(2):1–35
5. Billard A, Calinon S, Dillmann R, Schaal S (2008) Survey: robot programming by demonstration. Springer, Technical report
6. Schaal S (1999) Is imitation learning the route to humanoid robots? Trends Cogn Sci 3(6):233–242
7. Trafton JG, Cassimatis NL, Bugajska MD, Brock DP, Mintz FE, Schultz AC (2005) Enabling effective human–robot interaction using perspective-taking in robots. IEEE Trans Syst Man Cybern Part A Syst Hum 35(4):460–470
8. Katz G, Huang D-W, Hauge T, Gentili R, Reggia J (2017) A novel parsimonious cause-effect reasoning algorithm for robot imitation and plan recognition. IEEE Trans Cognit Dev Syst 10(2):177–193
9. Bandura A (2017) Psychological modeling: conflicting theories. Transaction Publishers, New Jersey
10. Melzoff AN (1995) Understanding the intentions of others: re-enactment of intended acts by 18-month-old children. Dev Psychol 31(5):838
11. Baldwin DA, Baird JA (2001) Discerning intentions in dynamic human action. Trends Cogn Sci 5(4):171–178
12. Tomasello M, Kruger AC, Ratner HH (1993) Cultural learning. Behav Brain Sci 16(3):495–511
13. Oztop E, Kawato M, Arbib MA (2013) Mirror neurons: functions, mechanisms and models. Neurosci Lett 540:43–55
14. Jackson PL, Melzoff AN, Decety J (2006) Neural circuits involved in imitation and perspective-taking. Neuroimage 31(1):429–439
15. Fogassi L, Ferrari PF, Gesierich B, Rozzi S, Chersi F, Rizzolatti G (2005) Parietal lobe: from action organization to intention understanding. Science 308(5722):662–667
16. Köster M, Langeloh M, Kliesch C, Kannagiesser P, Hoebl S (2020) Motor cortex activity during action observation predicts subsequent action imitation in human infants. Neuroimage 218:116958
17. Argall BD, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Auton Syst 57(5):469–483
18. Lee J (2017) A survey of robot learning from demonstrations for human–robot collaboration. arXiv:1710.08789
19. Barros JIO, dos Santos VMF, da Silva FMTP (2015) Bimanual haptics for humanoid robot teleoperation using ros and v-rep. In: 2015 IEEE international conference on autonomous robot systems and competitions. IEEE, pp 174–179
20. Fitzgerald T, Goel AK, Thomaz AL (2014) Representing skill demonstrations for adaptation and transfer. In: 2014 AAAI fall symposium series
21. Wu Y, Su Y, Demiris Y (2014) A morphable template framework for robot learning by demonstration: integrating one-shot and incremental learning approaches. Robot Auton Syst 62(10):1517–1530
22. Abbeel P, Coates A, Ng AY (2010) Autonomous helicopter aerobatics through apprenticeship learning. Int J Robot Res 29(13):1608–1639
23. Argall B, Browning B, Veloso M (2011) Learning mobile robot motion control from demonstrated primitives and human feedback. Robot Res 70:417–432
24. Ho J, Ermon S (2016) Generative adversarial imitation learning. Adv Neural Inf Process Syst 29
25. Osa T, Pajarinen J, Neumann G, Bagnell JA, Abbeel P, Peters J (2018) An algorithmic perspective on imitation learning. Found Trends Robot 7(1–2):1–179
26. MacGlashan J, Littman ML (2015) Between imitation and intention learning. In: Twenty-fourth international joint conference on artificial intelligence
27. Sun S-H, Noh H, Somasundaram S, Lim J (2018) Neural program automation (ICRA). IEEE, pp 1118–1125
28. Xu D, Nair S, Zhu Y, Gao J, Garg A, Fei-Fei L, Savarese S (2018) Neural-guided deductive search for real-time program synthesis from examples. arXiv:1804.01186
29. Davis GP, Katz GE, Gentili RJ, Reggia JA (2021) Compositional memory in attractor neural networks with one-step learning. Neural Netw 138:78–97
30. Katz GE, Davis GP, Gentili RJ, Reggia JA (2019) A programmable neural virtual machine based on a fast store-erase learning rule. Neural Netw 119:10–30
31. Sylvester J, Reggia J (2016) Engineering neural systems for high-level problem solving. Neural Netw 79:37–52
32. Davis GP, Katz GE, Gentili RJ, Reggia JA (2022) NeuroLISP: high-level symbolic programming with attractor neural networks. Neural Netw 146:200–219
33. Katz GE, Akshay, Davis GP, Gentili RJ, Reggia JA (2021) Tunable neural encoding of a symbolic robotic manipulation algorithm. Front Neurorobot 167
34. Gentili RJ, Oh H, Huang D-W, Katz GE, Miller RH, Reggia JA (2015) A neural architecture for performing actual and mentally simulated movements during self-intended and observed bimanual arm reaching movements. Int J Soc Robot 7(3):371–392
35. Lawson AE (2000) How do humans acquire knowledge? and what does that imply about the nature of knowledge? Sci Educ 9(6):577–598
36. Sprenger J (2011) Hypothetico-deductive confirmation. Philos Compass 6(7):497–508
37. Marcus JA (2012) An integrated model of clinical reasoning: dual-process theory of cognition and metacognition. J Eval Clin Pract 18(5):954–961
38. Reggia JA, Peng Y (1987) Modeling diagnostic reasoning: a summary of parsimonious covering theory. Comput Methods Programs Biomed 25(2):125–134
39. Lawson AE (2000) The generality of hypothetico-deductive reasoning: making scientific thinking explicit. Am Biol Teach 62(7):482–495
40. Holland D-W, Katz G, Langsfeld J, Gentili R, Reggia J (2015) A virtual demonstrator environment for robot imitation learning. In: 2015 IEEE international conference on technologies for practical robot applications (TePRA). IEEE, pp 1–6
41. Erol K, Hendler JA, Nau DS (1994) UMCP: a sound and complete procedure for hierarchical task-network planning. Aips 94:249–254
42. Lake BM, Ullman TD, Tenenbaum JB, Gershman SJ (2017) Building machines that learn and think like people. Behav Brain Sci 40
43. McCallum A, Nivre J, Tsarfaty A, Zhang J, Zettlemoyer L (2014) Neural network models for automatic named entity recognition. In: Proceedings of the 2014 conference on empirical methods in natural language processing. Association for Computational Linguistics, Atlanta, GA, pp 308–318
44. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M (2016) Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484–489
45. Kalyan A, Mohta A, Polozov O, Batra D, Jain P, Gulwani S (2018) Rearranging the familiar: testing generalization in recurrent networks. In: Proceedings of the 2018 EMNLP workshop BlackboxNLP: analyzing and interpreting neural networks for NLP, pp 108–114
58. Reggia JA, Katz GE, Davis GP (2019) Modeling working memory to identify computational correlates of consciousness. Open Philos 2(1):252–269
59. Lea C, Flynn MD, Vidal R, Reiter A, Hager GD (2017) Temporal convolutional networks for action segmentation and detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 156–165
60. Farha YA, Gall J (2019) Ms-tcn: multi-stage temporal convolutional network for action segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3575–3584
61. Simonyan K, Zisserman A (2014) Two-stream convolutional networks for action recognition in videos. Adv Neural Inf Process Syst 27

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Gregory P. Davis recently earned his PhD in computer science from the University of Maryland and is currently a postdoctoral associate in neuroscience at Weill Cornell Medicine. His research bridges these fields using biologically-inspired network models to study the neural basis of cognitive functions such as structured working memory and logical reasoning.

Garrett E. Katz is an Assistant Professor of Electrical Engineering and Computer Science at Syracuse University. His current research covers neural computation, automated planning and reasoning, and their integration in autonomous systems, with a focus on humanoid robotics. This research includes work on one-shot learning, robotic imitation learning, and automated program synthesis. He has published over 25 refereed papers in these areas and his research has been funded by SRC Inc., ONR, and DARPA.

Rodolphe J. Gentili is currently an Associate Professor in the Department of Kinesiology and a faculty member in the Neuroscience and Cognitive Science Program as well as the Maryland Robotics Center at the University of Maryland-College Park (USA). His research examines the underlying cognitive-motor control and learning processes in humans by employing a combination of experimental cognitive-motor neuroscience, computational modeling and robotics approaches. He has published over 70 articles in refereed journals and conference proceedings in these areas and his work has been supported by multiple research grants.

James A. Reggia is a Professor Emeritus of Computer Science at the University of Maryland with a joint appointment in the Institute for Advanced Computer Studies. He has conducted extensive research in nature-inspired computing, including in the areas of neural computation, machine learning, AI, genetic programming, machine consciousness, and artificial life. He has also worked in other more traditional areas of AI, such as cause-effect reasoning (abduction) and robotic imitation learning, and in modeling brain functions that can be related to neurological and neuropsychological disorders (Alzheimer’s disease, aphasia, PTSD, etc.). He has authored/co-authored over 200 refereed journal and conference papers in these areas, and has also routinely taught undergraduate and graduate courses in these fields.