Debated Backpropagation

Isaac James, Christopher Stone, and Alice Toniolo

School of Computer Science, University of St Andrews

Abstract. Dialogue has long been used in human society to explain seemingly opaque concepts. In this paper we focus on how to better explain training models for neural networks, to entertain as well as inform. We present a multi-agent argumentation-based dialogue system to generate human understandable dialogue to explain backpropagation. The system incorporates a model of agent personality and introduces social elements between agents to produce characterful discussion. Natural language templates are used to render utterances in English.

Keywords: Backpropagation · Explanation · Argumentation · Dialogue · NLG.

1 Introduction

Explainability in machine learning (ML) is not guaranteed by using an effective model. In fact, often the highest performing models (e.g. Deep Learning) are the least explainable. The issue of explainability in ML is pervasive, and is seen to affect the methods used to train models as well as how they perform [10].

In this paper, we seek to improve the explainability of one such training algorithm for neural networks, backpropagation, using dialogue as a means of explanation. Dialogue and argumentation are commonly used in human society to inform, persuade and deliberate over the course of action. To the same effect, dialogue can be used in computational systems to resolve conflict between cooperating software agents [14,17].

We propose a multi-agent system incorporating the elements of a neural network and backpropagation algorithm. The system includes a logical model of dialogue for agents to explain their preferred steps in the algorithm and a Natural Language Generation method for rendering these dialogues in a human understandable format. The system can also be configured to support other dialogue topics and agent personalities using a well defined interface. We define our own set of agents to give identity and character to our dialogue participants. Moreover, we give these agents personality and organise them in a social structure, with the aim of generating convincingly human dialogue that entertains as well as informs. Our method presents the initial steps to a novel approach to explaining an ML algorithm in an entertaining way.

2 Related work

Many current ML techniques, including neural networks, focus on constructing models in their internal representations [10]. These systems are difficult to explain as the models are opaque. Nevertheless, explainability plays a major role in optimising and

producing scientific outcomes [15]. Roscher et al. [15] presents two general forms of explanation: scientific explanations, which include the data, model and output obtained; and algorithmic explanations, which aim to "reveal underlying causes to the decision of an ML method". Existing research often focuses on scientific explanations either by embedding the generation of the explanations in the training algorithms or by creating an additional layer over existing algorithms to explain how a decision is computed [1]. In this research we follow the latter approach, but apply it to explaining training rather than the decisions made by the trained model.

In developing explanations, Roscher et al. [15] argues that explainability cannot be achieved solely by an explanation generated via an algorithm. Among other approaches, argumentation-based reasoning has recently been adopted for generating explainable systems due to the capabilities of argumentation to provide justifications for a decision (see [4] for a survey). When explanations are structured as argumentation frameworks, dialogue can be built upon them to provide means to interactively explore or challenge the explanations (e.g. [13,17]). We are concerned with explaining the steps of a back-propagation algorithm and what actions it dictates on the basis of the current state of the system. Among the types of dialogue proposed by Walton and Krabbe [19], the closest to our objectives are inquiry and deliberation. We adopt Riley's et al. [14] framework for multi-agent dialogues over actions, in which inquiry dialogue over beliefs is combined with persuasion dialogue over actions. Our dialogue does not deliver explanations, but the steps of backpropagation are externalised in a dialogical form, similar to [13].

Two additional dimensions are important in our work to enhance user engagement and support understanding of how a training algorithm works. Engagement with ML explanations has been sought through different means, including gamification of ML tasks (e.g., [5]) and storytelling [8]. In creating our narratives, we consider dialogue based explanations as argumentative speeches using rhetoric elements [2] whose informative and persuasive nature appeals to users' emotions to create engagement. Research on agent personality supports this task (e.g., [3]). Naturally, human normative reasoning is strongly influenced by emotion, which in turn is influenced by one's personality. Agent personality models can be used to simulate emotions. There exist many personality models generally consisting of a set of dimensions [6,7].

Finally, we deliver the rendered debates using a systematic approach for producing natural language text from non-textual data by borrowing techniques from Natural Language Generation (NLG) [9]. Generating a natural language text usually requires choosing the contents of an utterance and preparing the plan of the entire text before the final realisation of sentences [9]. However, template-based systems are able to map non-linguistic input directly to the realised sentences, omitting the need for an intermediate plan [18]. A dialogue system usually has to formulate one or at most a sequence of a few sentences at a time [11]. Thus, template-based systems can be useful for generating dialogues where the contents of an utterance are already decided (for example, by a dialogue move) and a limited set of sentence templates will suffice. NLG has been demonstrated in different argumentation systems for explanations, e.g. [13], but with less focus on the narrative output as in our approach.

We observed that there exists a wealth of related work in the areas of machine learning explainability, argumentation for explainability and multi-agent dialogue systems.

However, current work at the intersection of these fields, in particular using multi-agent dialogue systems to explain an ML algorithm in an entertaining way, is limited. In this research, we describe an initial system to address this gap.

3 A model of agents to discuss backpropagation

Here we present our general approach to deliver a dialogue based explanation of the backpropagation algorithm in an entertaining and easy to access way.

3.1 Neural Networks

A neural network is composed of a number of **neurons** connected via directed links [16]. Each link has a numeric **weight** associated with it. A neuron i connected to neuron j has weight $w_{i,j}$. The output of a neuron i is its **activation** value. To obtain the activation of neuron j, the weighted sum of the inputs is first calculated. This linear component is then transformed by a nonlinear component [16], called the **activation function**, g, to obtain the activation value $a_j = g(\phi_j + b_j)$. Here, neuron j is associated with a **bias** value, b_j. The bias is added to the weighted sum of inputs given to the activation function, which serves to adjust the activation higher or lower independent of the activations of the previous layer. Our neural network uses a **sigmoid** activation function, which outputs in the range (0, 1).

The goal of training a neural network is to set the weights and biases in order to minimise the error at the output layer. This error is a measure of how close the output is to the desired output, which is calculated using a loss function, $L(\mathbf{w}, \mathbf{b})$, where $\mathbf{w}$ and $\mathbf{b}$ are matrix representations of the weights and biases. For a multi-layer feedforward network, training can be achieved using backpropagation. At each step, the gradient of the loss function is computed with respect to the weights and biases. The negation of this gradient (the direction of steepest descent) is then used to update the weights and biases. This process is repeated until training converges on a minimal possible loss.

In this paper, we train a neural network for a classification task. Output targets are one-hot encoded, resulting in a target of 1 or 0 for each individual output neuron. For the purpose of later promoting discussion around whether these targets are met, we introduce the concept of **tolerance**, which is the minimal difference required between an output neuron's activation and its target for the neuron to be deemed **on target**.

3.2 Backpropagation dialogue in agent society

We construct our system around the idea of an agent society. Let us regard the natural hierarchy of a neural network's elements: the network itself, its layers and their respective neurons. We map this structure onto the concept of a labouring agent society, with each element represented by a single agent (see Fig. 1). The neuron agents reside at the lowest level, working to change weights and biases, and are concerned mainly by the personal impact of these changes on themselves and their immediate neighbours. Above them, the layer and network agents concentrate on the bigger picture: training outcomes and errors, guiding the lower levels towards improving these results. The single network

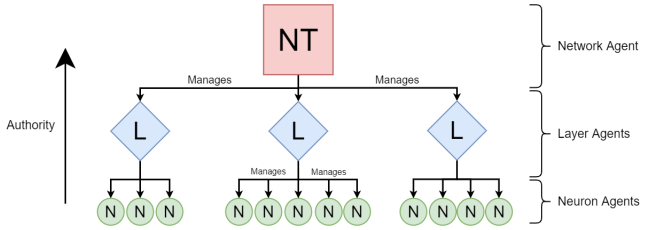

Fig. 1: Neural network agent society, showing structural and social hierarchy.

agent at the top is the most accountable, responsible for setting goals and communicating these to the layer agents. The layer agents form the 'middle-management' and must mediate the debate of the many neuron agents, whilst directing them to perform actions to meet the goals set by the network.

In backpropagation, the error in each output neuron is propagated back to connected hidden neurons according to the strength of the respective connections. The hidden neurons are therefore made responsible for some fraction of the output neuron's error. This notion of responsibility can be incorporated into an agent's personality. For example, their individual contribution to the overall training error might interact with their sense of moral or social responsibility to the other agents to influence their choices in a dialogue game. Agent roles and their social relationships are encoded within our application of the belief and personality framework of an agent.

With the aim to convey an explanation of the steps of the backpropagation algorithm, our dialogue can be structured as follows:

1. Begin with the network agent stating an overall goal (e.g., a desired classification).
2. Proceed with the network agent inquiring as to whether the goal has been met.
3. Participants then exchange relevant information to discover the answer to the inquiry.
4. If the goal has not been met, participants argue about what action to take until consensus is reached.

This dialogue incorporates two of the argumentation dialogue types proposed by Walton and Krabbe [19]: deliberation and inquiry. Through inquiry, agents may discover what is currently blocking the satisfaction of the goal, which can be used to inform a choice of action in the deliberation phase that follows.

3.3 Our agent framework

Our approach to represent the dialogue uses the framework from Riley et al [14] extended for our objectives. An agent x is equipped with the following components:

1. Epistemic knowledge: facts and defeasible rules to enable an agent's epistemic reasoning. Facts represent information regarding the current state of the backpropagation algorithm from which agents form b-arguments.
2. Normative knowledge: action rules formalised as an Action-Based Alternating Transition System (AATS) to enable an agent to decide what to do in order to reach the stated goal. From actions agents form a-arguments.

3. Inquiry dialogue protocol rules: enabling agents to collectively discover the status of the target goal using b-arguments; for example whether a neuron should activate.
4. Deliberation dialogue protocol rules: enabling agents to collectively decide what to do to reach the target goal (e.g. modify the activation threshold) using a-arguments.
5. An agent personality and role, modelled as a vector of binary personality dimensions, which determine what arguments take priority in the dialogue and how this is rendered in natural language.
6. A set of Natural language templates that are instantiated on the basis of the agent personality and role.

The following section will present how we instantiated this framework in our work.

4 A Multi-Agent System to Explain Backpropagation

We implemented our system in Python, on the basis of Nielsen's code [12]. The network uses a sigmoid activation function and uses mini-batch stochastic gradient descent.

4.1 Agents

Our agent model encapsulates an agent's epistemic knowledge, normative knowledge and personality dimensions. In our system we have five agent types: Network (NT), Layer (Li), Input Neuron (INi), Hidden Neuron (HNij) and Output Neuron (ONi). The symbols i, j indicate numerical indices that identify individual layer and neuron agents.

The network agent NT encapsulates the neural network, including the backpropagation algorithm. NT is also responsible for initialising and storing the other agent objects in the society, including the layers and neurons, as well as updating the values of their beliefs to reflect the current state of the network. For example, NT will update the weight, bias and activation beliefs of the neuron agents to ensure that they are consistent with the latest values. NT is also capable of updating its own beliefs between training iterations, such as the target classification of a training sample.

The layer agents Lis are responsible for deriving knowledge relevant to individual neuron agents from NT's beliefs, such as activation targets for individual output neurons using a target class known to the network. The neuron agents Nis are responsible for harbouring most of the normative knowledge in the society. Nis know about the weights and biases in the network and what actions can be performed to modify them.

4.2 Agent Personality

We model agent personalities as a vector of binary personality dimensions. These dimensions can be associated with certain modifications to an agent's knowledge framework, altering what information they may share in an inquiry or what actions they may propose during deliberation. Thus, an agent's personality is self-contained within the agent's knowledge framework, allowing the dialogue protocols to remain generic.

Personality dimensions may also be used to alter the natural language templates used for certain facts. This allows different personalities to be superimposed onto the

same sequence of moves via changes in natural language. An optional *'intent'* with an asserted fact or action argument, represents the intention of the agent behind the assertion and indicate the template variant to be used by the natural language generator.

4.3 Agent knowledge

Agent Configuration. An agent configuration file allows an agent to store the knowledge base in a concise data-interchange format, JSON. Configuration files make the system accessible to less-technical users by providing a way to edit an agent's knowledge base to craft different dialogues without the need to write any code. The agent configuration files include the required sections: facts, rules and AATS, below:

```
{ "facts": { "all": [...], "pa" : [...], "pb" : [...] },
  "rules": { "all": [...], "!pa": [...] },
  "AATS" : { "all": [...] }}
```

Knowledge definitions within these sections can be optionally associated with personality traits p_i, as shown. These form a key-value pair, with the personality trait string as the key and array of knowledge definitions as the value. A prefix `!` specifies that knowledge applies to all agents that *do not* display a particular personality trait.

Epistemic knowledge. The belief system represents an agent's epistemic knowledge and is formed by defeasible facts, literals denoted as α, and defeasible rules $\alpha_1 \wedge ... \wedge \alpha_n \rightarrow \alpha_0$ where $\alpha_1, ..., \alpha_n$ are the *premises* and α_0 is the *conclusion*.

Facts. An agent maintains facts α representing ground predicates listed in `facts` and including any predicates instantiated by the network agent. A fact α_1 has a structure $subject(X1, \ldots, Xn, Value, Intent)$ where x_i represents the agents which the fact relates to. These identifiers are important in our domain to be used in the fact's natural language template to refer to agents by name. In the configuration file we represent these facts as `$X1_..._$Xn_subject(Value)#Intent` strings with a different structure for ease of manipulation. The string `$self` represents the name of the agent the knowledge belongs to. Neuron specific references can also be used in a neuron configuration file, such as `$otherNeuronInLayer` and `$nextLayerNeuron`, which act as wildcards that can apply to multiple other neurons. The following fields are optional: `Value` specifies a numerical value for the fact and `Intent` associates an intention with a fact. `Intent` has no meaning in the context of the logical system and is only referred to during NLG to select the template used to render the fact in a sentence.

Facts instantiated by the network agent have named values specific to neuron agents:

- `X1_activation` & `X1_bias`: neuron $X1$'s last activation and bias values.
- `X1_X2_weight`: weight value of the link between neuron $X1$ and a neuron $X2$ in the proceeding layer.
- `X1_target`: value of 0 or 1, representing output neuron $X1$'s target activation on the last training sample.
- `X1_tolerance`: value in the range $[0, 0.5)$ representing how close neuron $X1$'s activation value must be to the target for it to be deemed on target.

An example of fact is `$self_$ON1_weight` indicating that the connection this weight is associated with is between the neuron agent (`$self`) and output neuron 1 (`$ON1`).

Rules. Rules are represented as $\alpha_1 \wedge ... \wedge \alpha_n \rightarrow \alpha_0$. In addition, special rules act as enablers for other rules, which is fundamental in generating a coherent dialogue. We define specific derivation enabling rules of the form $r_1 : \alpha_1 \wedge ... \wedge \alpha_n \rightarrow \delta$ such that $r_2 : \delta \wedge \alpha_1 \wedge ... \wedge \alpha_n \rightarrow \alpha$ but for simplicity we represent this with rule names $\alpha_1 \wedge ... \wedge \alpha_n \rightarrow r_2$. This facilitates more finely grained control over the order of rule assertions in the inquiry dialogue. To prevent issues with non-terminating defeasible derivations, the definition of a nested rule is constrained to ensure that no fact acting as a rule's premise can occur in its conclusion as part of another rule.

In the configuration file, `rules` $\alpha_1 \wedge ... \wedge \alpha_n \rightarrow \alpha_0$ are represented as strings of the form `premises -> conclusion`. In `premises` `&` and `!` are used in place of $\wedge$ and $\neg$. A clause α may consist of one or more numerical facts combined with Pythonic mathematical notation to construct a boolean sub-expression. For example, consider a rule string known to a neuron agent:

```
$self_targetIs1 & $self_activation < 0.5 -> $self_activationTooLow
```

This specifies that for the claim to be valid, the neuron agent's target activation must be 1 (`$self_targetIs1`) and its actual activation (`$self_activation`) must be a value less than 0.5. The claim is `$self_activationTooLow`, which is true if the neuron agent's activation fell below its target.

Normative knowledge. For deliberation dialogue, we need to bestow agents with normative knowledge about the effects of actions. Following [14], for an agent x an AATS is composed by a set of finite states Q_x, where $q_0^x \in Q^x$ is the initial state and $q_t^x \in Q^x$ is the goal. Actions in the set Ac^x can be applied if their preconditions are satisfied in a state q_i^x. A state $\tau^x(q_i, a_i)$ results by the performance of a_i from state q_i. We do not consider values in this work. The initial state of an agent's AATS can be derived from the defeasible facts agreed upon by all agents, which are discovered during the inquiry dialogue combining epistemic to practical reasoning. In our implementation of AATS, each transition is represented with `source` and `destination` states, and the `action` that links the two states. Actions are represented as strings, optionally composed with a separating underscore. This underscore indicates a special case of a fact modification action. The *modification* is represented by the substring preceding the first underscore (for example, `increase` or `decrease`) and the numerical fact to which the modification is applied is represented by the substring following the first underscore. For example, the transition below states that if the agent's activation is below target, the agent can increase its bias so that the activation is no longer too low.

```
{"source": ["$self_activationTooLow"],
"action": "increase_$self_bias",
"destination": ["!$self_activationTooLow"]}
```

4.4 Dialogue

In order for agents to participate in the dialogue, they must be able to construct arguments on the basis of their beliefs and AATS. Following [14], two types of arguments are formed by an agent given the epistemic and normative knowledge.
A defeasible derivation of a belief from facts and rules is represented with b-arguments, while an a-argument can be constructed when an agent wishes to argue for an action:

- A **belief argument** is denoted $B = \langle \Phi, \phi \rangle$, where ϕ is a defeasible belief and Φ is a minimally consistent set of beliefs from which ϕ can be derived.
- An **action argument** constructed by an agent x from its AATS is a 4-tuple $A = \langle q_x, a, q_y, p \rangle$, where $q_x = q_0^x$ is the *start state*, $a \in Ac^x$ is the proposed *action*, $\tau^x(q_x, a) = q_y$ is the *end state* and $p \in q_t^x$ is the *goal*.

Our dialogue is a combination of deliberation and inquiry. The deliberation protocol begins with an inquiry sub-dialogue whose topic is the set of facts that make up the deliberation goal. The aim of this inquiry is to collectively discover relevant information to the goal. After this, the agents move to deliberation where the goal is a target state for the physical system after the deliberated actions are performed.

In a dialogue, an agent x can make the moves: **open** $\langle x, open, dialogue(\theta, \gamma, \Lambda) \rangle$; **assert** $\langle x, assert, \Upsilon \rangle$; and **close** $\langle x, close, dialogue(\theta, \gamma, \Lambda) \rangle$. The type of dialogue protocol is described by $dialogue(\theta, \gamma, \Lambda)$: if $\theta = inquiry$, γ is a set of propositions whose truth values are to be found; if $\theta = deliberation$, γ is a state representing the goal. Λ is the set of participants, Υ is either a set of a-arguments or b-arguments depending on θ which has not yet been asserted. A **commitment store**, CS_x, stores an agent x's exchanged information. The **global commitment store**, CS_g, is the union of the commitment stores of all agents participating in the dialogue.

In the inquiry each agent asserts all its relevant beliefs that are not already present in CS_g. Relevant means that either this is a fact, a premise of a rule in the global commitment store, or a rule of which conclusion is a relevant defeasible fact. For the remainder of the dialogue, each agent checks whether any of its asserted defeasible rule are fully supported by the information in CS_g. For a supported rule, an agent will assert the rule conclusion as a b-argument if the conclusion is not already present in CS_g. If the agent cannot assert anything new then they will instead move to close the dialogue.

In the deliberation dialogue, CS_g is set to the sub-dialogue store after the inquiry is complete. If it is discovered during the inquiry that all the goal propositions are satisfied, then no action needs to be taken and the protocol can terminate early. Otherwise, the protocol proceeds with each agent asserting relevant a-arguments formed from their AATS. Each agent determines the start state $q_0^x \subset CS_g$ using information in CS_g, q_0^x is the largest possible subset of CS_g in Q^x. If any action transitions exist in its AATS from q_0^x to the end state, the goal, then the agent will assert an a-argument to propose the action. Algorithms 1-2 demonstrate these protocols.

In our implementation, CS_g acts as a blackboard in the system, broadcasting agent assertions. To determine the agent speaking order, we leverage a token-ring algorithm in which a token is exchanged among the participants of the dialogue in some fixed order, starting from the network agent `NT`. However, this choice has a cost in generating natural language. For example, unlike direct messages, broadcast messages have no specific recipient associated with them. Therefore, when generating natural language from a broadcast message, we may have to infer who the intended recipient is using information in the message if we wish to address a specific agent.

4.5 Natural Language Generation

We implement the rendering of dialogues into natural language using a template-based NLG system [18]. Templates, provided for each fact and action, are sentences with zero

Algorithm 1: INQUIRY

Input: A list $\Lambda = (x_1, x_2, ..., x_n)$ of agents
An initiator $x_I \in \Lambda$
A list $\gamma = (\alpha_1, \alpha_2, ..., \alpha_n)$ (the topic)
A global commitment store CS_g
Output: A list of moves
$moves \leftarrow \{\langle x_I, open, dialogue(inq, \gamma, \Lambda)\rangle\}$
$no\_participants \leftarrow |\Lambda|$
$closes \leftarrow 0, t \leftarrow 0$
while $closes < no\_participants$ **do**
  $x_i \leftarrow \Lambda[t]$
  $\Upsilon \leftarrow$ `RelevantBeliefs`$(x_i, \gamma, CS_g) \cup$
    `SupportedBArguments`(x_i, γ, CS_g)
  $\Upsilon \leftarrow \Upsilon - CS_g$
  if $|\Upsilon| > 0$ **then**
    $moves \leftarrow moves \cup \{\langle x_i, assert, \Upsilon\rangle\}$
    $CS_g \leftarrow CS_g \cup \Upsilon$
    $closes \leftarrow 0$
  else $closes \leftarrow closes + 1$
  $t \leftarrow (t + 1) \pmod{no\_participants}$
return $moves$

Algorithm 2: DELIBERATION

Input: A list $\Lambda = (x_1, x_2, ..., x_n)$ of agents
An initiator $x_I \in \Lambda$
A list $\gamma = (\alpha_1, \alpha_2, ..., \alpha_n)$ (the topic)
Output: A list of moves
$CS_g \leftarrow \emptyset$
$moves \leftarrow \{\langle x_I, open, dialogue(del, \gamma, \Lambda)\rangle\}$
$no\_participants \leftarrow |\Lambda|$
$closes \leftarrow 0, t \leftarrow 0$
$moves \leftarrow moves \cup$ `Inquiry`$(\Lambda, x_I, \gamma, CS_g)$
while $closes < no\_participants$ **do**
  $x_i \leftarrow \Lambda[t]$
  $\Upsilon \leftarrow$ `SupportedAArguments`(x_i, γ, CS_g)
  $\Upsilon \leftarrow \Upsilon - CS_g$
  if $|\Upsilon| > 0$ **then**
    $moves \leftarrow moves \cup \{\langle x_i, assert, \Upsilon\rangle\}$
    $CS_g \leftarrow CS_g \cup \Upsilon$
    $closes \leftarrow 0$
  else $closes \leftarrow closes + 1$
  $t \leftarrow (t + 1) \pmod{no\_participants}$
return $moves$

or more slots that can be replaced with interchangeable words or phrases to produce varied output. Additionally, identifiers in these slots are replaced by specific values, such as the numerical value of the associated fact or names from the agent list. Similar to the agent configuration files, these template strings can be defined in a separate JSON file, grouped via fact names. This file also includes the various strings that can be substituted in the place of the template slots. These non-terminal nodes are synonyms or variations of specific words that help producing a diverse, but equivalent, set of sentences.

In the inquiry dialogue, we generate sentences for assertions of defeasible facts, including those asserted as the claim of a b-argument, using templates for each fact. We do not immediately generate sentences for the assertion of defeasible rules, including those asserted as the support of a b-argument. If a rule is later used as a support, then sentences will be generated for the premise assertions, so we can avoid first generating a sentence for the rule to introduce these premises.

In the deliberation dialogue, we generate sentences for asserted a-arguments using templates for each action. Here, it is not necessary to generate sentences to describe either the AATS start state, which is a combination of facts already asserted in the dialogue, or the end state, which is the goal that was stated in the opening sentence.

For example, the natural language template for the weight fact `X1_X2_weight` and the action 'increase' are shown below. The fact includes substitutions for the names of the neuron agents that share the weight and the value of the weight.

```
weight: "the weight between X_1 and X_2 is VALUE"
increase: "I'll increase FACT"
```

5 An example application

In this section we present an application of our dialogue system. The deliberation goal is for a specific output neuron agent to meet its activation target on a given training

sample. The neural network is trained on the Iris flower dataset from the UCI Machine Learning Repository[1] using four input features to identify three classes.

Epistemic knowledge. In the inquiry portion of the dialogue, agents will discover whether the output neuron agent is on target before deliberating. This target, an activation of either 0 or 1, will first be passed down the social hierarchy.

Listing 1.1: Network agent belief base

```
1  ‘‘rules”: {
2  ‘‘all”: [‘‘targetClass1 −> targetIsSetosa”,
3       ‘‘targetClass2 −> targetClassVersicolor”,
4       ‘‘targetClass3 −> targetClassVirginica”]
5       }
```

Listing 1.2: Layer agent belief base

```
1  ‘‘rules”: {
2  ‘‘all”: [‘‘targetIsSetosa −> ON1_targetIs1”,
3       ‘‘targetIsSetosa −> ON2_targetIs0”,
4       ‘‘targetIsSetosa −> ON3_targetIs0”,
5       ‘‘targetIsVersicolor −> ON1_targetIs0”, ...]}
```

At the highest level, the network agent is given knowledge of the class labels (Listing 1.1). The layer agents are then given rules to translate the target class label into target activations for individual output neuron agents (Listing 1.2).

Listing 1.3 shows how the output neuron agent determines whether their activation is on target, using the rules in lines 5-10.

Listing 1.3: Output neuron agent belief base

```
1  ‘‘facts”: {
2    ‘‘!forgetful”: [‘‘$self_tolerance(0.1)”],
3    ‘‘forgetful”: [‘‘$self_isForgetful”, ‘‘$self_tolerance(0.4)”]},
4  ‘‘rules”: {
5    ‘‘all”: [‘‘$self_targetIs1 & $self_act > (1 − $self_tolerance) −> $self_onTarget”,
6      ‘‘$self_targetIs0 & $self_act < $self_tolerance −> $self_onTarget”,
7      ‘‘$self_targetIs1 & $self_act < (1 − $self_tolerance) −> $self_actTooLow”,
8      ‘‘$self_targetIs0 & $self_act > $self_tolerance −> $self_actTooHigh”,
9      ‘‘$self_actTooHigh −> !$self_onTarget”,
10     ‘‘$self_actTooLow −> !$self_onTarget”],
11   ‘‘!forgetful”: [‘‘$other_isForgetful −>
12       $other_targetIs1 & $other_act < (1 − $self_tolerance#correction) −> $other_actTooLow#correction”,
13     ‘‘$other_isForgetful −>
14       $other_targetIs1 & $other_act > $self_tolerance#correction −> $other_actTooLow#correction”]}
```

Forgetfulness demonstrates an example of personality dimension. Observe that the true value of the tolerance is known to all non-forgetful output neuron agents (line 2), whereas a forgetful agent will ‘misremember’ a more lenient version (line 3). In lines 11-14 we introduce the rules for non-forgetful agents which allow them to correct others that may have incorrectly determined themselves to be on target. Note that these rules are nested; the outer premise prevents an agent from asserting their belief of the tolerance unless another agent is forgetful. The *correction* intent is attached to the true value of the tolerance, which is later used to formulate the natural language as a correction.

Normative Knowledge. In the deliberation portion of the dialogue, agents suggest actions to help the output neuron agent meet its target. We let hidden neuron agents propose weight changes for their connections to help a neuron agent in the next layer meet their target. As an example, the increase weight action is shown in Listing 1.4. The source state for this transition is the next neuron’s activation being too low.

[1] https://archive.ics.uci.edu/ml/datasets/iris

Table 1: Example dialogues.

Dialogue A:	Net:	Time to classify another Iris flower.
	ON1 (S):	Reporting for duty! My activation is 0.56 and the classification tolerance is 0.10.
	Net:	Ok, the training data says this example is a Setosa.
	L2:	Output Neuron 1, your target is 1.
	ON1 (S):	My activation is too low. I've missed my target.
	HN1 (L2):	Don't worry, I'll increase the weight between myself and Output Neuron 1.
	ON1 (S):	No! I'll increase my bias, it'll work better anyway.
Dialogue B:	Net:	Let's get started! Time to classify another Iris flower.
	ON1 (F):	At your command! My activation is 0.25 and the classification tolerance is 0.50.
	Net:	Good, the training data says this example is a Virginica.
	L2:	Output Neuron 1, your target is 0.
	ON1 (F):	What a relief! My activation is on target.
	ON2:	Wait a second, that's not correct. The classification tolerance is actually 0.10, so your activation is too low.
	HN1 (L2):	Don't worry! I'll increase the weight between myself and Output Neuron 1.

Listing 1.4: Hidden neuron AATS

```
1 ‘‘all”: [{
2     ‘‘source”: [‘‘$nextLayerNeuron_activationTooLow”],
3     ‘‘action”: ‘‘increase_$self_$nextLayerNeuron_weight”
4     ‘‘destination”: [‘‘$nextLayerNeuron_onTarget”]}, ...
```

Listing 1.5: Output neuron AATS

```
1 ‘‘stubborn”: [{
2     ‘‘source”: [‘‘$self_activationTooLow”],
3     ‘‘action”: ‘‘increase_$self_bias#stubborn”,
4     ‘‘destination”: [‘‘$self_onTarget”]}, ...
```

We inject personality into output neuron (ON) agents through normative knowledge, letting stubborn agents insist on meeting missed targets without help by changing their bias. In Listing 1.5, the 'stubborn' key is later used to formulate the refusal of help.

Dialogue. We illustrate here two different dialogues that may be generated from the example application (see Table 1). In Dialogue A, ON1 has a stubborn personality. When it is discovered that they are not on target, a hidden neuron in the preceding layer suggests an action. ON1 refuses, insisting on its own course of action. In Dialogue B, ON1 has a forgetful personality. When they incorrectly find themselves to be on target, another ON corrects them. During deliberation, a hidden neuron then suggests an action.

6 Conclusions

A multi-agent dialogue system was demonstrated as a means of enriching backpropagation with dialogue. The generic dialogue framework ensures the ease of extending the system to support further examples. Our flexible system can produce entertaining and informative dialogue to help explain the mechanics of backpropagation.

Possible future work includes extending the dialogue topics to provide more variety in the system's output and improve explainability of related aspects of backpropagation, such as error propagation or overfitting. Extended topics could focus on using information aggregated over the course of many training iterations, preventing dialogue from becoming repetitive over the course of many epochs. Evaluation with users would help establish the advantages that this approach provides in understanding backpropagation and assess the usefulness of specific explanations as perceived by users. Depending on the familiarity of these users with ML concepts, such as neural networks, these studies could also help identify where dialogues may require additional preamble to explain

the technical concepts discussed by agents. Currently, the model of personality is limited; no partial dimensions or model of emotional state was implemented. The model of agent personality could be adapted to adhere to a more flexible model, such as the PEN [7] model, and used to extend the model of dialogue to support moral argumentation for agents to support their normative reasoning. Finally, the current dialogue system is not able to influence the training of the neural network. Actions proposed in deliberation could be used to dynamically modify the training to better explain the consequences of various modifications on performance.

References

1. Biran, O., Cotton, C.: Explanation and justification in machine learning: A survey. In: Proceedings of the IJCAI-17 Workshop on Explainable AI (XAI). vol. 8, pp. 8–13 (2017)
2. Blair, J.A.: Rhetoric, dialectic, and logic as related to argument. Philosophy & Rhetoric **45**(2), 148–164 (2012)
3. Castelfranchi, C., Rosis, F.D., Falcone, R., Pizzutilo, S.: Personality traits and social attitudes in multiagent cooperation. Applied Artificial Intelligence **12**(7-8), 649–675 (1998)
4. Čyras, K., Rago, A., Albini, E., Baroni, P., Toni, F.: Argumentative XAI: A survey. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence (2021)
5. Di Nunzio, G.M., Maistro, M., Zilio, D.: Gamification for machine learning: The classification game. In: Proceedings of the 3rd GamifIR'16 Workshop (2016)
6. Egges, A., Kshirsagar, S., Magnenat-Thalmann, N.: A model for personality and emotion simulation. In: Palade, V., Howlett, R.J., Jain, L. (eds.) Knowledge-Based Intelligent Information and Engineering Systems. pp. 453–461. Springer Berlin Heidelberg (2003)
7. Eysenck, H.J.: Biological dimensions of personality. (1990)
8. Faiella, F., Ricciardi, M.: Gamification and learning: a review of issues and research. Journal of e-Learning and Knowledge Society **11**(3) (2015)
9. Gatt, A., Krahmer, E.: Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. J. of Artificial Intelligence Research **61**, 65–170 (2018)
10. Gunning, D., Aha, D.: DARPA's explainable artificial intelligence (XAI) program. AI Magazine **40**(2), 44–58 (Jun 2019). https://doi.org/10.1609/aimag.v40i2.2850
11. Mykowiecka, A.: Natural-language generation—an overview. International Journal of Man-Machine Studies **34**(4), 497–511 (1991)
12. Nielsen, M.A.: Neural Networks and Deep Learning. Determination Press (2015)
13. Oren, N., van Deemter, K., Vasconcelos, W.W.: Argument-Based Plan Explanation, pp. 173–188. Springer International Publishing (2020)
14. Riley, L., Atkinson, K., Payne, T., Black, E.: An implemented dialogue system for inquiry and persuasion. In: Modgil, S., Oren, N., Toni, F. (eds.) Theorie and Applications of Formal Argumentation. pp. 67–84. Springer Berlin Heidelberg (2012)
15. Roscher, R., Bohn, B., Duarte, M.F., Garcke, J.: Explainable machine learning for scientific insights and discoveries. IEEE Access **8**, 42200–42216 (2020)
16. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. 3 edn. (2010)
17. Sklar, E.I., Azhar, M.Q.: Explanation through argumentation. In: Proceedings of the 6th International Conference on Human-Agent Interaction. p. 277–285 (2018)
18. van Deemter, K., Theune, M., Krahmer, E.: Real versus template-based natural language generation: A false opposition? Computational linguistics **31**(1), 15–24 (2005)
19. Walton, D., Krabbe, E.C.W.: Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning. State University of New York Press (1995)