General AI Challenge
Round One: Gradual Learning

Jan Feyereisl, Matej Nikl, Martin Poliak, Martin Stransky, Michal Vlasak
GoodAI, Prague, Czech Republic*

Abstract

The General AI Challenge is an initiative to encourage the wider artificial intelligence community to focus on important problems in building intelligent machines with more general scope than is currently possible. The challenge comprises of multiple rounds, with the first round focusing on *gradual learning*, i.e. the ability to re-use already learned knowledge for efficiently learning to solve subsequent problems. In this article, we will present details of the first round of the challenge, its inspiration and aims. We also outline a more formal description of the challenge and present a preliminary analysis of its curriculum, based on ideas from computational mechanics. We believe, that such formalism will allow for a more principled approach towards investigating tasks in the challenge, building new curricula and for potentially improving consequent challenge rounds.

1 Introduction

Existing artificial intelligence (AI) algorithms available today constitute the so-called narrow AI landscape, meaning that they have been designed, trained, and optimized by human engineers to solve a single, specific task or a very narrow collection of closely related problems. Although such algorithms sometimes outperform humans in their established skill-set, they are not able to extend their capabilities to new domains. This limits their re-usability, potentially increases the amount of data required to train them, and leaves them lacking generality and extensibility to higher order reasoning.

In contrast, algorithms capable of overcoming these limitations could eventually converge towards a repertoire of functionalities akin to a human-level skill-set. Such algorithms might be able to learn to come up with creative solutions for a wide range of multi-domain tasks. Systems employing aforementioned algorithms, oftentimes termed under the umbrella of general AI systems, are viewed by many as the ultimate leverage in solving many of humanity's direst problems [Bostrom, 2014; Domingos, ; Stone *et al.*, 2016].

*Correspondence: {jan.feyereisl, matej.nikl, martin.poliak, martin.stransky, martin.vlasak}@goodai.com

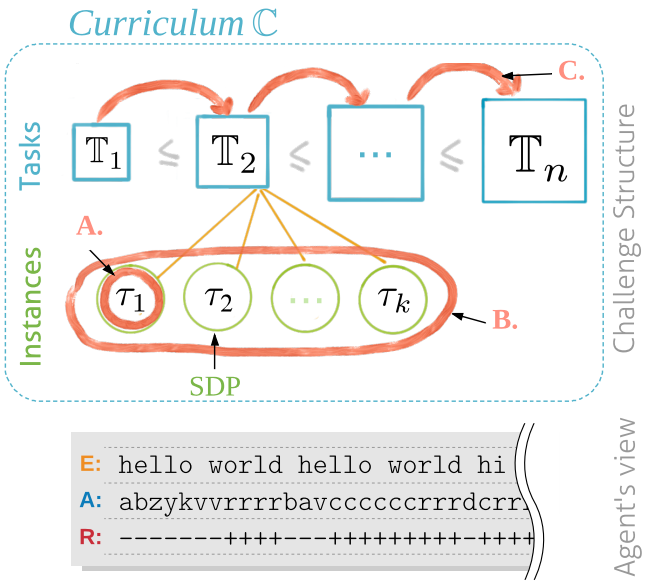


Figure 1: Overview of the first round of the General AI Challenge. The structure of the challenge is hierarchical and multi-objective, and requires learning at multiple levels to solve a curriculum $\mathbb{C}$. From fast learning (A.) at individual instance level of each task, where a 'good enough' policy π_i for a stochastic decision process (SDP) must be found, through shared sub-goal discovery (B.) across instances $\tau_1, \ldots, \tau_k$ leading to one-shot or few-shot learning, all the way to lower level knowledge and policy transfer (C.) across different tasks $\mathbb{T}_i$ for speeding up convergence towards solutions on yet unseen tasks. What agent sees is shown at the bottom, i.e. continual streams of data with no instance or task separation information, where **E** denotes the environment stream, **A** the agent's response and **R** the reward.

Gradual learning is an ability of learning systems to learn in a gradual manner. This is likely a necessary pre-condition for acquiring such a broad set of skills, striving to solve a wide range of disparate problems [Rosa *et al.*, 2016; Kumaran *et al.*, 2016; Tommasino *et al.*, 2016; Rusu *et al.*, 2016; Balcan *et al.*, 2015; Pentina and Lampert, 2015; Ruvolo and Eaton, 2013; Hamker, 2001]. Unfortunately, most existing algorithms struggle to deal with many tasks with different distributions at the same time and tasks that drift in distribution from one another [Ditzler *et al.*, 2015] and frequently exhibit catastrophic forgetting, once applied to new problems, especially under realistic computational constraints [Rusu *et al.*, 2016; Fernando *et al.*, 2017; Hamker, 2001; Kirkpatrick *et al.*, 2016; French, 1999; Gershman *et al.*, 2015]. Gradual learning is therefore a difficult and unsolved problem that requires a focused investigation by the community. The General AI Challenge provides a platform for such directed focus.

2 Round One: Gradual Learning

The aim of the first round of the challenge is to build an agent that exhibits **gradual learning** [Rosa *et al.*, 2016] as evidenced by solving a curriculum of increasingly complex and disparate sets of tasks. Despite the similarity in name to gradual learning in [Kumaran *et al.*, 2016] and other related concepts such as cumulative [Tommasino *et al.*, 2016], incremental [Rusu *et al.*, 2016], life-long [Balcan *et al.*, 2015; Pentina and Lampert, 2015; Ruvolo and Eaton, 2013; Hamker, 2001] or continual learning [Kirkpatrick *et al.*, 2016], in our setting, the concept encompasses a broader class of requirements that an agent needs to satisfy. These include:

1. *Exploiting previously learned knowledge*
2. *Fast adaptation to unseen problems*
3. *Avoiding catastrophic forgetting*

The above requirements are expressed in two primary objectives of the first round, each with its own set of evaluation criteria:

Objective 1 *(Quantitative)* Agent capable of passing an evaluation curriculum in the *shortest number of simulation steps*. Passing requires the ability to exploit previously learned knowledge and avoid catastrophic forgetting, all within a predefined time limit.

Alternatively, a description of a conceptual method for achieving gradual learning is also sought:

Objective 2 *(Qualitative)* An idea, concept, or design that shows the best promise for scalable gradual learning. A working AI agent is desirable but not necessary.

The fundamental focus of the challenge is on gradual learning, i.e. on the efficient re-use of already gained knowledge. Competitors are required to develop solutions with the following properties:

Property 1 (Gradual Learning) *An ability to re-use previously gained knowledge for the acquisition of subsequent knowledge to more efficiently solve hitherto new and yet unseen problems, while avoiding catastrophic forgetting.*

It is important to note here not only the re-use of existing knowledge, but also the focus on *efficiency* of solving new and unseen problems. Unlike compositionality or other concepts similar to gradual learning, our definition transcends the boundaries of meta-learning [Duan *et al.*, 2017b; Wang *et al.*, 2016; Duan *et al.*, 2017a; Santoro *et al.*, 2016; Chen *et al.*, 2016b; 2016a]. We seek agents that are able to learn to *quickly adapt* to new and unseen tasks, while exploiting the gradual structure of the underlying problems they are solving. One can think of the above as that the agents need to possess the ability to search for more efficient strategies to solve new problems.

Naturally, when solving new problems, agents must be able to retain the ability to solve old tasks. In many systems, the lack of such ability causes catastrophic forgetting [French, 1999], which must be avoided:

Property 2 (Avoiding Catastrophic Forgetting) *Avoiding the loss of information, relevant for one task, due to the incorporation of knowledge necessary for a new task.*

In summary, the aim is not optimizing for agent's performance of existing skills, i.e. how good an agent is at delivering solutions to problems it has already encountered. Instead, we desire optimizing for agent's performance on solving new and unseen problems, i.e. maximizing the speed of convergence to 'acceptable solutions' on new problems, while exploiting existing knowledge and ensuring its survival during the acquisition of new information.

3 Background

To fully appreciate our setting, below we provide background on why graduality can be beneficial and how it can be encouraged. This is followed by a more formal description of the challenge requirements, environment and evaluation procedures.

3.1 Benefits of Graduality

Given a complex task that needs to be solved, frequently a good strategy for finding a solution is to break the problem down into smaller problems which are easier to deal with. The same applies to learning [Bengio *et al.*, 2009; Salakhutdinov *et al.*, 2013]. It can be much faster to learn things gradually than to try to learn a complex skill from scratch [Alexander and Brown, 2015; Zaremba and Sutskever, 2015; Gulcehre *et al.*, 2016; Oquab *et al.*, 2014]. One example of this is the hierarchical decomposition of a task into subtasks and the gradual learning of skills necessary for solving each of them [Krueger and Dayan, 2009; Vezhnevets *et al.*, 2017; Lee *et al.*, 2017; Andreas *et al.*, 2017], progressing from the bottom of the hierarchy to the top [Mhaskar *et al.*, 2016; Mhaskar and Poggio, 2016; Poggio *et al.*, 2015; Polya, 2004]. A prime example in the natural world is the gradual acquisition of motor skills by infants during their first years of life [Adolph and Franchak, 2017].

3.2 Guidance through Curricula

Exploiting graduality during learning is clearly beneficial. Building systems that learn in a gradual manner should then

be encouraged and its benefits and limitations explored further. To enable such type of learning, one can control and guide the learning process. *Guided learning*, also called curriculum learning, [Gülçehre and Bengio, 2016; Bengio *et al.*, 2009; Vapnik, 2015] provides control by means of presenting the learner with parts of the problem in the order and extent that is likely to be most beneficial at that point in time. One method of providing such order is in the form of a learning curriculum [Bengio *et al.*, 2009], akin to a curriculum used in schools. In this scenario, easier topics are taught before more complex ones, in order to exploit the gradual nature of taught knowledge.

Gradual and guided learning has a number of other benefits over other types of learning [Gulcehre *et al.*, 2016; Gülçehre and Bengio, 2016; Bengio *et al.*, 2009; Pan and Yang, 2010]. For example, optimizing a model that has few parameters and gradually building up to a model with many parameters could be more efficient than starting with a model that has many parameters from the beginning [Stanley and Miikkulainen, 2002]. In this case, a smaller number of new parameters is learned at each step [Chen *et al.*, 2015; Kirkpatrick *et al.*, 2016; Andreas *et al.*, 2016b; Rusu *et al.*, 2016]. This might also result in reducing the necessity for exploration. Furthermore, apriori knowledge of the system's architecture might not be necessary [Zhou *et al.*, 2012; Rusu *et al.*, 2016; Ganegedara *et al.*, 2016], and architecture size can be dynamically derived during training to correspond to the complexity of given problems [Fahlman and Lebiere, 1990]. Last but not least, reuse of already learned skills is feasible and encouraged [Andreas *et al.*, 2016b; 2016a]. Once a skill is acquired, it is no longer relevant how long the skill took to discover. The cost of using an existing skill is notably smaller than searching for a skill from scratch.

4 Learning to Gradually Learn

Having described the objectives of the challenge and established the benefits of gradual learning through a curriculum, we will now focus on formal definitions and descriptions of the requirements of the first round of the challenge.

4.1 Instances, Tasks & Curricula

The gradual learning ability of an agent is evaluated by subjecting an agent to a sequence of n tasks $\mathbb{T}_1, \mathbb{T}_2, \ldots, \mathbb{T}_n$ of increasing complexity. Such sequence is called a curriculum:

Definition 1 (Curriculum) *A curriculum* $\mathbb{C} = (\mathbb{T}_1, \ldots, \mathbb{T}_n)$ *is an n-tuple of tasks of increasing complexity. Tasks are endowed with an arbitrary measure of complexity.*

In section 8.3, we present one possible way of measuring task complexity in a principled manner and to ensure proper ordering of curricula. It can be performed with the help of a measure from the field of computational mechanics, namely statistical complexity C_μ [Crutchfield, 1994].

Each task is observed by an agent through task instances $\tau \sim \mathbb{T}$. The publicly available curriculum provided as part of the challenge [Poliak *et al.*, 2017] can be seen as a curriculum of distributions over stochastic decision problems (SDPs). In particular, a variant of partially observable Markov decision processes (POMDPs), or more generally partially observable stochastic games (POSGs).

Definition 2 (Task and Instance) *A task* $\mathbb{T}$ *is a distribution over a set of Partially Observable Markov Decision Processes. An instance of a task* $\tau \sim \mathbb{T}$ *is a sample from said distribution.*

In other words, a single instance τ of a task $\mathbb{T}$ is a POMDP. A POMDP is an 8-tuple $(\mathcal{S}, \mathcal{A}, \Omega, \mathcal{P}, \mathcal{O}, r, \gamma, T)$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ a set of actions, Ω a set of observations, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ a transition probability distribution, $\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \Omega \to [0, 1]$ is an observation probability distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a reward function, γ a discount factor and T a horizon. In the standard POMDP formulation, the goal of an agent is to maximize its future discounted reward $\mathbb{E}_\sigma \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$, where the expectation is over the sequence of agent's belief state/action pairs $\sigma = ((b_0, a_0), \ldots, (b_T, a_T))$, where $b_0(s) = \mathbb{P}(\mathcal{S} = s)$ is an initial belief state, $a_t \sim \pi_\theta(a_t|b_t)$, $b_{t+1} = \mathbb{P}(b_{t+1}|b_t, a_t, o_t)$ and π_θ denotes a policy parameterized by θ.

Unlike in the standard setting however, the goal in the challenge round is different and distributed across multiple levels of hierarchy. We are not interested in maximizing the agent's future discounted reward, but rather a more complex set of objectives at different scales. For example at the instance level, it is sufficient to find an acceptable solution to each instance τ_i, as described in Algorithm 1. Figure 1 shows the three levels of hierarchy present in the challenge:

1. **Fast learning** (A): *Policy Search* − At the individual instance level, the agent is required to find a solution (e.g. a policy π) to a single POMDP/task instance τ.
2. **Slow learning** (B): *Meta-Policy Discovery* − The agent needs to discover a meta-strategy (e.g. a meta-policy π^{meta}) that quickly converges to a solution across all instances τ_i.
3. **Fast Adaptation** (C): *Policy Transfer* − The agent is required to exploit existing acquired knowledge and policies in order to adapt to each new task in a curriculum, to solve it faster.

Whether it is possible to find a correspondence between the challenge objectives and the standard reward formulation remains to be seen and up to competitors to determine.

In addition to the above hierarchy, a number of objectives and constraints need to be satisfied by competing agents. The primary goal of an agent is the completion of the entire curricula $\mathbb{C}$ in as short a time as possible.

Definition 3 (Quantitative Objective) *Successful completion of an evaluation curriculum* $\mathbb{C}_e$ *in the shortest number of time-steps possible among all competitors and within 24 hours from the start of the evaluation process on predefined evaluation H/W. Given a set of competing agents A, the fastest agent is determined according to:*

$$\arg\min_{\alpha \in A} \varrho(\alpha, \mathbb{C}_e)$$

where $\varrho : A \times \mathbb{C} \to \mathbb{N}_+$ *corresponds to* `RunCurriculum()` *in Algorithm 2 which returns the number of simulation steps it took an agent* α *to successfully complete a curriculum* $\mathbb{C}$.

Satisfying the condition set forth in Definition 3, the agent also has to show that two additional conditions are met, namely gradual learning and avoiding catastrophic forgetting.

Definition 4 (Gradual Learning) *A manifestation of Property 1, exhibited through a reduced number of computational steps required when solving a task $\mathbb{T}$ ensuing the solving of previous tasks, i.e. $\varrho(\alpha_{(\mathbb{T}_1,\mathbb{T}_2)}, (\mathbb{T}_3)) < \varrho(\alpha, (\mathbb{T}_3))$, where $\alpha_{(\mathbb{T}_1,\mathbb{T}_2)}$ denotes agent α that has already learned to solve tasks $\mathbb{T}_1$ and $\mathbb{T}_2$.*

The above condition does not ensure that gradual learning truly occurs, nevertheless it provides sufficient evidence that an agent improves its performance on subsequent tasks, having already solved other tasks before.

Definition 5 (Avoiding Catastrophic Forgetting) *A manifestation of Property 2 through the maintenance of fast convergence to an acceptable solution for already solved tasks, i.e. $\varrho(\alpha_{(\mathbb{T}_1,\mathbb{T}_2,\mathbb{T}_3)}, (\mathbb{T}_2)) \leq c\varrho(\alpha_{(\mathbb{T}_1,\mathbb{T}_2)}, (\mathbb{T}_2))$ where c is some constant of the agent.*

This condition ensures that learning to solve a new task does not impair the ability to solve previously encountered tasks.

4.2 Validation Curricula

To test generalization of gradual learning, an agent trained on a single curriculum $\mathbb{C}$ cannot be effectively evaluated on tasks from the same curriculum. A different curriculum is necessary. Curricula appropriate for a gradually learning agent also form a distribution $\mathfrak{C}$ from which a training and an evaluation curriculum should be drawn. The mini and micro tasks used in the challenge (described in section 6) form together one sample from this distribution. Using a single sample curriculum $\mathbb{C} \sim \mathfrak{C}$ to estimate the distribution $\mathfrak{C}$ might be too difficult. It is possible that competitors might need to create their own curricula. Examples of a number of possible final tasks from alternative curricula were shown in [Rosa, 2017].

5 Evaluation

As mentioned previously, simply maximizing reward R is neither sufficient, nor desired. Immediately after the agent reaches an acceptable performance R^* on an instance, the environment presents it with the next sample τ. Upon the successful completion of a number of instances from the same task, determined according to Algorithm 1, the environment presents the agent with the next task in the curriculum. This can be seen in Algorithm 2.

The `EnvStep` and `AgentStep` functions in Algorithm 1 update the environment and the agent respectively, while exchanging reward, input and output. Together, they form the core of the environment-agent communication loop, depicted in Figure 2.

The helper functions `SoftLimit` and `HardLimit` are used to compute the respective time step limits for a given instance. As the naming suggests, there are two types of limits considered when running an instance: a soft limit, and a hard limit. Both limits are represented in terms of a number of environment steps. They are dynamically computed as they depend on the current instance – in fact, they can even evolve as the agent progresses through the instance.

Algorithm 1: Progression through one task instance τ

input : An agent α, a task instance τ_i, the minimum number of positive rewards R^*, uninterrupted by a negative one, required to solve current instance
output: The number of steps t taken to solve τ_i.

```
function Solve(α, τ_i, R*)
    t ← 0;                                  # Counter of elapsed steps
    R_+ ← 0;                                # Counter of positive rewards
    R, o_t, τ_i ← EnvStep(τ_i, ∅);          # Environment
    # Agent-environment loop until reward/time-limit reached
    repeat
        α, a_t ← AgentStep(α, R, o_t);      # Agent
        R, o_t, τ_i ← EnvStep(τ_i, a_t);    # Environment
        t ← t + 1;                          # Increment elapsed steps
        # If correct, increment reward counter, else re-set it
        if R = 1 then
            R_+ ← R_+ + 1;                  # Increment reward counter
        else if R = −1 then
            R_+ ← 0;                        # Reset positive reward counter
        end
    until R_+ = R* or t = HardLimit(τ_i);
    return t;                               # No. of steps taken to solve instance τ_i
end
```

The soft limit can be thought of as a *success* limit. The agent is required to solve the instance within this limit in order for this attempt to count as successful. If it fails to do so, the prospective solution will no longer be considered as successful. The instance, however, does not end yet. This allows the agent to continue exploring and gather some additional useful knowledge about the unsolved problem.

A forceful instance termination comes only when the *hard* limit is reached. This behavior is beneficial in situations where an agent gets stuck in a particular instance.

The reasoning behind such limits is as follows: the agent has no way of communicating its needs, for example the need to change the instance to a different one, or switching to the previous task. Therefore, those limits can be thought of as supplement to such cases.

6 Learning Environment

Participants are provided with an environment and a public curriculum of tasks of increasing complexity [Poliak *et al.*, 2017]. Both, the environment and the associated tasks are specifically constructed to minimize distraction from unrelated research problems and to primarily focus on gradual learning and associated obstacles. The environment and tasks are built on top of the CommAI-env [Mikolov *et al.*, 2015; Baroni *et al.*, 2017].

Participants are asked to develop and train their agents on the public curriculum and possibly any other additional knowledge or data sources that they have at their disposal. The public set of tasks provides a reference example of how the evaluation curriculum might be structured and the type of

Algorithm 2: Progression through a curriculum $\mathbb{C}$

input : An agent α, a curriculum $\mathbb{C}$ and the number of successfully solved consecutive instances N_s required

output: The total number of steps T the agent needed to solve curriculum $\mathbb{C}$

```
function RunCurriculum(α, ℂ, N_s)
    T ← 0;                                  # Counter of elapsed steps
    # Solve each task 𝕋 in a curriculum ℂ
    foreach 𝕋 in ℂ do
        N_j ← 0;                            # Counter of consecutive successes
        R_j ← ReqReward(𝕋);                 # Required min. reward
        # Successfully solve N_s instances of task 𝕋
        repeat
            τ_i ~ 𝕋;                        # Sample a new instance
            t_i ← Solve(α, τ_i, R_j);       # Solve instance
            T ← T + t_i;                    # Increment elapsed steps
            t_s ← SoftLimit(τ_i);           # Calc. soft limit
            # If solved sufficiently quickly
            if t_i ≤ t_s then
                N_j ← N_j + 1;              # increment success counter
            else
                N_j ← 0;                    # else reset counter
            end
        until N_j = N_s;
    end
    return T;                               # No. of steps taken to solve ℂ
end
```

tasks that will be required to be solved by an agent. Tasks come in two groups, as simple *micro-tasks* and as more advanced *mini-tasks*. It is important to note, that natural language processing is not necessary to solve the tasks.

6.1 Mini-tasks

Mini-tasks are based directly on the CommAI-mini task set [Baroni *et al.*, 2017], with focus on simple grammar processing and deciding language membership problems for strings. The mini-tasks are simple for an educated and biased human, but they are tremendously complex for an unbiased agent that receives only sparse reward. For this reason, we believe that simpler tasks are necessary. We refer to those as *micro-tasks*.

6.2 Micro-tasks

The purpose of micro-tasks is to allow the agent to acquire a sufficient repertoire of prior knowledge, necessary for discovering the solutions to the more complex mini-tasks in a reasonable time, under the assumption that the agent is capable of gradual learning.

Decomposing mini-tasks into simpler problems yields a number of skills that the agent should learn first. For each such skill, there is a separate micro-task. The micro-tasks build on top of each other in a gradual and compositional way, relying on the assumption that the agent can make use of them efficiently.

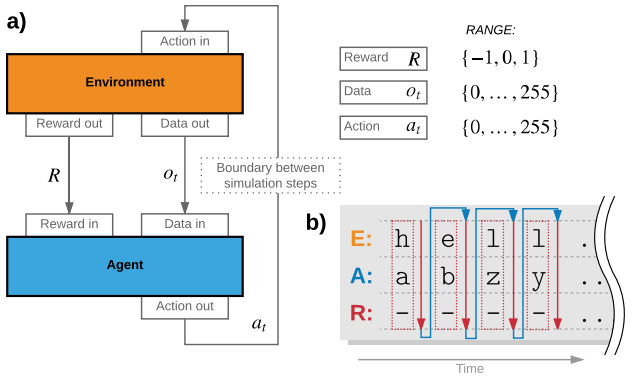


Figure 2: a) Interface between the agent and the environment. b) Depiction of environment output with overlaid simulation order.

One could argue that the micro-tasks (and even the mini-tasks) are too simple and can be solved by hard-coding the necessary knowledge into the agent. This is indeed true. However, such a solution is not desirable and goes against the spirit of the challenge - to learn new skills from data. The challenge will prevent any hard-coded solution by using hidden evaluation tasks which are different from the public curriculum. A hard-coded solution that does not exhibit gradual learning should fail on such evaluation tasks.

6.3 Environment-Agent Interface

The interface between the agent and the environment is depicted in Figure 2. The environment sends reward $\{-1, 0, 1\}$ and data (one byte) to the agent and receives the agent's action (one byte) in response. This happens in a continuous cycle. During a single simulation step, the environment processes the received action from the agent and sends reward with new data to the agent; the agent processes this input and sends an action back to the environment.

Specifically, the environment sends 8 bits of data to the agent at every simulation step and receives 8 bits of data back from the agent. This is different from the original CommAI-env [Mikolov *et al.*, 2015], which sends only a single bit instead of a full byte every time. Although this makes the interface slightly less flexible, the agent's communication should be more interpretable and it should reduce the complexity of the presented problems.

Note that the environment is not sending any special information about what task the agent is in at any point in time. The environment appears as a continuous stream of bytes (and rewards) to the agent.

7 Discussion

Looking back at Figure 1, it is apparent that there is a significant gap between the structure of the curriculum, the objectives to be learned within, and what kind of information the agent receives. Most notably, there is no explicit information about a successful completion of an instance or a task. Similarly, no indication of an instance or a task switch is provided. The lack of information and limited amount of feedback that

an agent has at its disposal makes this round challenging and different from most other learning problems.

Moreover, unlike standard POMDPs, for many of the tasks in the public curriculum, the reward is adaptive. This can be very challenging due to the drifting nature of the reward function. Another challenging aspect of the first round is the inability to request previous tasks from the environment. Hence, competing agents need to be sample efficient as well as 'epoch' efficient.

8 Task Complexity via Computational Mechanics

Computational mechanics [Crutchfield, 2011], a subfield of physics, is concerned with exploring the ways in which nature performs computations. How to identify structure in natural processes and how to measure it, as well as how information processing is embedded in dynamical behavior. The field offers a number of useful tools for thinking about and quantifying complex systems, some of which are relevant for building intelligent machines. Namely, we are interested in the concept of ϵ-machines [Crutchfield and Young, 1989] and the associated measure of statistical complexity C_μ.

In our scenario, these concepts and tools can be useful for exploring curricula of the challenge, from the point of view of the complexity of each individual task and the ability to begin investigating the possibility of automating the creation of more principled curricula for both training and evaluation.

8.1 ϵ-Machines as Minimal Optimal Models

The concept of ϵ-machines as a class of minimal optimal predictors was developed for quantifying structure in a stochastic process via the reconstruction of underlying causal states in a natural process [Crutchfield, 1994]. In their most common form, they can be thought of as a type of hidden Markov model (HMM), whose states have a unique, causal definition. ϵ-machines are, however, not limited to only a HMM representation and can in fact be used at various levels of representation hierarchy, depending on whether current representation hits the limits of the agent's computational resources [Crutchfield, 1994].

An ϵ-machine can be constructed with the help of an equivalence relation

$$\overleftarrow{y} \sim_\epsilon \overleftarrow{y}' \iff \mathbb{P}\left(\overrightarrow{Y}|\overleftarrow{Y} = \overleftarrow{y}\right) = \mathbb{P}\left(\overrightarrow{Y}|\overleftarrow{Y} = \overleftarrow{y}'\right) \quad (1)$$

where Y denotes a discrete random variable, $\overleftarrow{Y}$ and $\overrightarrow{Y}$ a block of past (e.g. $\overleftarrow{Y_t} = \ldots Y_{t-2}Y_{t-1}$) and future random variables, respectively, with $\overleftarrow{y}$ and $\overrightarrow{y}$ denoting a particular history or a future (a sequence of symbols from alphabet $\boldsymbol{\mathcal{Y}}$) of a generating process, respectively. The equivalence relation (1) states that any two histories $\overleftarrow{y}$ and $\overleftarrow{y}'$ are equivalent if the probability distribution over their futures is the same, i.e. $\mathbb{P}\left(\overrightarrow{Y}|\overleftarrow{Y} = \overleftarrow{y}\right) = \mathbb{P}\left(\overrightarrow{Y}|\overleftarrow{Y} = \overleftarrow{y}'\right)$. Given the above equivalence relation, one can then define a mapping from the space of histories $\overleftarrow{\boldsymbol{\mathcal{Y}}}$ to 'causal' states of the underlying process. Such mapping $\epsilon : \overleftarrow{\boldsymbol{\mathcal{Y}}} \to \boldsymbol{\mathcal{S}}$ is called an ϵ-map that takes a given past $\overleftarrow{y}$ to its corresponding causal state σ_i:

$$\epsilon\left(\overleftarrow{y}\right) = \sigma_i = \left\{\overleftarrow{y}' : \overleftarrow{y} \sim_\epsilon \overleftarrow{y}'\right\}$$

Intuitively, one can think of ϵ as a function that partitions the set of pasts $\overleftarrow{\boldsymbol{\mathcal{Y}}}$ according to the equivalence relation (1), into clusters that lead to the same distribution over futures.

We can then define the ϵ-machine as a tuple $(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{T}})$ where $\boldsymbol{\mathcal{Y}}$ denotes the process' alphabet, $\boldsymbol{\mathcal{S}}$ its causal state set, and $\boldsymbol{\mathcal{T}}$ a set of symbol-valued transition matrices:

$$\boldsymbol{\mathcal{T}} \equiv \left\{T^{(y)}\right\}_{y \in \boldsymbol{\mathcal{Y}}}$$

where $T^{(y)}$ comprises the following elements:

$$T^{(y)}_{ij} = \mathbb{P}(S_1 = \sigma_j, Y_0 = y | S_0 = \sigma_i),$$

where S_0 and S_1 denote two temporally consecutive random variables of a random variable chain defining the causal state process that underlies the stochastic process we are modeling. Each S_t takes on some value $s_t = \sigma_i \in \boldsymbol{\mathcal{S}}$. The set $\boldsymbol{\mathcal{T}}$ thus defines the dynamics over causal states of the underlying system. An ϵ-machine is then a unique, minimal-size, maximally predictive unifilar representation of the observed process [Crutchfield and Young, 1989].

Despite their desirable minimality and optimality properties, ϵ-machines are inherently able to model generative processes only. In order to exploit the useful properties that ϵ-machine's formulation offers for the analysis of our curriculum, we need to go beyond output processes and exploit a slightly more complex representation, that of input-output processes, namely ϵ-transducers.

8.2 ϵ-Transducer

In our scenario, we are interested in observing the structure of not only one process, but a coupling between two stochastic processes, which can also be viewed as the analysis of a communications channel between an agent and its environment. The concept of an equivalence relation can be extended, for this type of scenario, over joint (input-output) pasts:

$$\begin{aligned}\overleftarrow{(x,y)} \sim_\epsilon \overleftarrow{(x,y)}' \iff \\ \mathbb{P}\left(\overrightarrow{Y}|\overrightarrow{X}, \overleftarrow{(X,Y)} = \overleftarrow{(x,y)}\right) \\ = \mathbb{P}\left(\overrightarrow{Y}|\overrightarrow{X}, \overleftarrow{(X,Y)} = \overleftarrow{(x,y)}'\right)\end{aligned} \quad (2)$$

, defining a basis for a channel's unique, maximally predictive, minimal statistical complexity unifilar presentation, called ϵ-transducer. Similarly to ϵ-machines, a map from pasts to states can be created $\epsilon : \overleftarrow{(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}})} \to \boldsymbol{\mathcal{S}}$ that maps given joint input-output past $\overleftarrow{(x,y)}$ to its corresponding channel causal state

$$\epsilon\left(\overleftarrow{(x,y)}\right) = \sigma_i = \left\{\overleftarrow{(x,y)}' : \overleftarrow{(x,y)} \sim_\epsilon \overleftarrow{(x,y)}'\right\}.$$

The ϵ-transducer is then defined as the tuple $(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{T}})$, where, in contrast to ϵ-machines, $\boldsymbol{\mathcal{X}}$ additionally defines the

set of inputs and $\mathcal{T}$ the set of conditional transition probabilities:

$$\mathcal{T} \equiv \left\{T^{(y|x)}\right\}_{x\in\boldsymbol{\mathcal{X}},y\in\boldsymbol{\mathcal{Y}}}$$

where $T^{(y|x)}$ has elements:

$$T_{ij}^{(y|x)} = \mathbb{P}(S_1 = \sigma_j, Y_0 = y | S_0 = \sigma_i, X_0 = x)$$

The ability of ϵ-transducers to model input-output mappings, enables the incorporation of actions and makes modeling of tasks in a curriculum $\mathbb{C}$ feasible.

8.3 Statistical and Structural Complexity

Having defined both ϵ-machines and ϵ-transducers, we can now define a distribution $\tilde{\pi}$ over causal states for ϵ-machines as

$$\begin{aligned}\tilde{\pi}(i) &= \mathbb{P}(S_0 = \sigma_i) \\ &= \mathbb{P}\left(\epsilon\left(\overleftarrow{Y}\right) = \sigma_i\right)\end{aligned}$$

and an input-dependent state distribution $\tilde{\pi}_X$ for ϵ-transducers:

$$\begin{aligned}\tilde{\pi}_X(i) &= \mathbb{P}_X(S_0 = \sigma_i) \\ &= \mathbb{P}_X\left(\epsilon\left(\overleftarrow{(X,Y)}\right) = \sigma_i\right)\end{aligned}$$

In both cases, this defines the asymptotic probability of a process being in any one of its causal states. This information can then be used for quantifying the complexity of the underlying generating or input-output process, respectively. This measure, also called *statistical complexity* in the case of ϵ-machines, is defined as $C_\mu = \mathrm{H}[\tilde{\pi}]$, where μ points to the fact that it is a measure over an asymptotic distribution, and H denotes the Shannon entropy [Shannon and Weaver, 1948]. On the other hand, ϵ-transducer's input-dependent statistical complexity is defined as

$$C_X = \mathrm{H}[\tilde{\pi}_X].$$

The upper bound on ϵ-transducer complexity is then the channel complexity, calculated as the supremum of statistical complexity over input processes:

$$\overline{C_\mu} = \sup_X C_X$$

with topological state complexity being $C_0 = \log_2 |\boldsymbol{\mathcal{S}}|$ and due to the fact that uniform distributions maximize Shannon entropy, in general $\overline{C_\mu} \leq C_0$ [Barnett and Crutchfield, 2015].

Statistical complexity C_μ is a widely used measure of complexity in physics and complexity science. It has a number of benefits over other measures and intuitive interpretations. For example, it measures the amount of information a process needs to store in order to be able to reconstruct a given signal, or in the case of structural complexity, it defines the capacity of a communication channel. In the case of tasks in our challenge, it could provide a principled way of measuring an upper bound on task complexity that can be beneficial in the construction of structured curricula and subsequent automatic generation of tasks of increasing complexity.

9 Agent-agnostic Task and Curriculum Representation

Modeling tasks $\mathbb{T}$ of the challenge curriculum $\mathbb{C}$ with the help of ϵ-transducers allow for an agent-agnostic representation of the complexity of the problem each task tackles. It not only provides a way to measure complexity of each task, but also to compare and contrast entire task curricula and reason about the inherent compositionality and graduality of an entire curriculum (see Figure (3b)). Preliminary results for a number of tasks from the challenge curriculum can be seen in the appendix.

9.1 Task and Curriculum Complexity

Using statistical and structural complexity we are able to start reasoning about the correctness of the curriculum order. This allows for more well defined curricula and subsequent possibility of automating the curriculum generating process with guaranteed characteristics.

9.2 Task and Curriculum Graduality

The minimal representation of tasks also allows us to start thinking about comparison of how much structure is shared among tasks within a curriculum. This can be beneficial in designing and measuring the intrinsic graduality of a curriculum. As ϵ-machines and transducers are inherently graphical models, graph theoretic tools and algorithms, such as subgraph-matching, can be used to start comparing tasks and measure their level of shared structure that can be eventually exploited by an agent with gradual learning capabilities.

10 Conclusions

In this work, we have outlined the structure and goals of the first round of the General AI Challenge, focused on building agents that can learn gradually. We have argued that building such agents requires solving problems at three different levels of hierarchy. From fast learning, akin to a search for a policy at a level of an instance of a task, through slow learning that requires the discovery of a meta-policy that generalizes across instance, all the way to fast adaptation to new tasks, akin to policy transfer between disparate problems. Furthermore, we have proposed the use of a method from computational mechanics, namely ϵ-transducers for modeling of tasks in a curriculum that allow for quantifying task complexity and potentially graduality of entire curricula. Explicit calculation of the complexity of tasks and its subsequent use for building better curricula is left for future work.

Acknowledgements

We would like to thank the entire GoodAI team, in particular Olga Afanasjeva and Marek Rosa.

References

[Adolph and Franchak, 2017] Karen E Adolph and John M Franchak. The development of motor behavior. *Wiley Interdisciplinary Reviews: Cognitive Science*, 8(1-2):e1430—-n/a, 2017.

[Alexander and Brown, 2015] William H. Alexander and Joshua W. Brown. Hierarchical Error Representation: A Computational Model of Anterior Cingulate and Dorsolateral Prefrontal Cortex. *Neural computation*, 27(11):2354–2410, nov 2015.

[Andreas *et al.*, 2016a] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to Compose Neural Networks for Question Answering. 2016.

[Andreas *et al.*, 2016b] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural Module Networks. *Cvpr*, pages 39–48, 2016.

[Andreas *et al.*, 2017] Jacob Andreas, Dan Klein, and Sergey Levine. Modular Multitask Reinforcement Learning with Policy Sketches. In *ICLR*, 2017.

[Balcan *et al.*, 2015] Maria-Florina Balcan, Avrim Blum, Santosh Vempala, and Georgia Tech. Efficient Representations for Lifelong Learning and Autoencoding. 40:1–20, 2015.

[Barnett and Crutchfield, 2015] Nix Barnett and James P. Crutchfield. Computational Mechanics of Input-Output Processes: Structured Transformations and the epsTransducer. *Journal of Statistical Physics*, 161(2):404–451, 2015.

[Baroni *et al.*, 2017] Marco Baroni, Armand Joulin, Allan Jabri, Germàn Kruszewski, Angeliki Lazaridou, Klemen Simonic, and Tomas Mikolov. CommAI: Evaluating the first steps towards a useful general AI. *arXiv:1701.08954*, jan 2017.

[Bengio *et al.*, 2009] Yoshua Bengio, Jerome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, New York, New York, USA, 2009. ACM Press.

[Bostrom, 2014] Nick Bostrom. *Superintelligence : paths, dangers, strategies*. Oxford University Press, 2014.

[Chen *et al.*, 2015] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2Net: Accelerating Learning via Knowledge Transfer. *arXiv Preprint*, pages 1–10, 2015.

[Chen *et al.*, 2016a] Yutian Chen, Matthew W. Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matt Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. nov 2016.

[Chen *et al.*, 2016b] Yutian Chen, Matthew W. Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Timothy P. Lillicrap, and Nando de Freitas. Learning to Learn for Global Optimization of Black Box Functions. *NIPS*, nov 2016.

[Crutchfield and Young, 1989] James P. Crutchfield and Karl Young. Inferring statistical complexity. *Physical Review Letters*, 63(2):105–108, jul 1989.

[Crutchfield, 1994] James P. Crutchfield. The calculi of emergence: computation, dynamics and induction. *Physica D: Nonlinear Phenomena*, 75(1-3):11–54, aug 1994.

[Crutchfield, 2011] James P. Crutchfield. Between order and chaos. *Nat Phys*, 8(February):17–24, 2011.

[Ditzler *et al.*, 2015] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in Non-stationary Environments: A Survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, nov 2015.

[Domingos,] Pedro. Domingos. *The master algorithm : how the quest for the ultimate learning machine will remake our world*.

[Duan *et al.*, 2017a] Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-Shot Imitation Learning. 2017.

[Duan *et al.*, 2017b] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, Pieter Abbeel, and Computer Science. RL2: Fast Reinforcement Learning via Slow Reinforcement Learning. In *ICLR*, pages 1–14, 2017.

[Fahlman and Lebiere, 1990] Scott E Fahlman and Christian Lebiere. The {Cascade-Correlation} Learning Architecture. In D S Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 524–532. Morgan-Kaufmann, 1990.

[Fernando *et al.*, 2017] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. jan 2017.

[French, 1999] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[Ganegedara *et al.*, 2016] Thushan Ganegedara, Lionel Ott, and Fabio Ramos. Online Adaptation of Deep Architectures with Reinforcement Learning. 2016.

[Gershman *et al.*, 2015] Samuel J Gershman, Eric J Horvitz, and Joshua B Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278, 2015.

[Gülçehre and Bengio, 2016] Ça\uglar Çağlar Gülçehre and Yoshua Bengio. Knowledge Matters: Importance of Prior Information for Optimization. *J. Mach. Learn. Res.*, 17(8):1–32, 2016.

[Gulcehre *et al.*, 2016] Caglar Gulcehre, Marcin Moczulski, Francesco Visin, and Yoshua Bengio. Mollifying Networks. *Arixv*, pages 1–11, 2016.

[Hamker, 2001] Fred H Hamker. Life-long learning Cell Structures - Continuously learning without catastrophic interference. *Neural Networks*, 14(4-5):551–573, 2001.

[Kirkpatrick *et al.*, 2016] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. dec 2016.

[Krueger and Dayan, 2009] Kai A. Krueger and Peter Dayan. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394, mar 2009.

[Kumaran *et al.*, 2016] Dharshan Kumaran, Demis Hassabis, and James L. McClelland. What Learning Systems do Intelligent Agents Need? Complementary Learning Systems Theory Updated. *Trends in Cognitive Sciences*, 20(7):512–534, 2016.

[Lee *et al.*, 2017] Sungtae Lee, Sang-Woo Lee, Jinyoung Choi, Dong-Hyun Kwak, and Byoung-Tak Zhang. Micro-Objective Learning : Accelerating Deep Reinforcement Learning through the Discovery of Continuous Subgoals. mar 2017.

[Mhaskar and Poggio, 2016] Hrushikesh Mhaskar and Tomaso Poggio. Deep vs. shallow networks : An approximation theory perspective. (054):1–16, 2016.

[Mhaskar *et al.*, 2016] Hrushikesh Mhaskar, Qianli Liao, and Tomaso Poggio. Learning Functions: When Is Deep Better Than Shallow. *arXiv*, (45):1–12, 2016.

[Mikolov *et al.*, 2015] Tomas Mikolov, Armand Joulin, and Marco Baroni. A Roadmap towards Machine Intelligence. 2015.

[Oquab *et al.*, 2014] M Oquab, L Bottou, I Laptev, and J Sivic. Learning and transferring mid-level image representations using convolutional neural networks. *Proc. IEEE*, 2014.

[Pan and Yang, 2010] S J Pan and Q Yang. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, oct 2010.

[Pentina and Lampert, 2015] Anastasia Pentina and Christoph H Lampert. Lifelong Learning with Non-i . i . d . Tasks. In *NIPS*, pages 1–9, 2015.

[Poggio *et al.*, 2015] Tomaso Poggio, Fabio Anselmi, and Lorenzo Rosasco. I-theory on depth vs width: hierarchical function composition. (041), 2015.

[Poliak *et al.*, 2017] Martin Poliak, Martin Stransky, Michal Vlasak, Jan Sinkora, and Premek Paska. General AI Challenge - Round1. *https://github.com/general-ai-challenge*, 2017.

[Polya, 2004] G Polya. *How to solve it: a new aspect of mathematical method*. Princeton University Press, 2004.

[Rosa *et al.*, 2016] Marek Rosa, Jan Feyereisl, and The GoodAI Collective. A Framework for Searching for General Artificial Intelligence. *arXiv:1611.00685*, nov 2016.

[Rosa, 2017] Marek Rosa. General AI Challenge Specifications of the First (Warm-Up) Round: Gradual Learning - Learning Like a Human. *GoodAI*, 2017.

[Rusu *et al.*, 2016] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *arXiv*, 2016.

[Ruvolo and Eaton, 2013] Paul Ruvolo and Eric Eaton. ELLA: An efficient lifelong learning algorithm. *Proceedings of the 30th International Conference on Machine Learning*, 28(1):507–515, 2013.

[Salakhutdinov *et al.*, 2013] R Salakhutdinov, J B Tenenbaum, and A Torralba. Learning with {Hierarchical-Deep} Models. *IEEE Trans. Pattern Anal. Mach. Intell.*, (8):1958–1971, 2013.

[Santoro *et al.*, 2016] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-Learning with Memory-Augmented Neural Networks, 2016.

[Shannon and Weaver, 1948] C. E. Shannon and W. Weaver. A mathematical theory of communication. *Bell Syst. Tech. J*, 27:379–423, 1948.

[Stanley and Miikkulainen, 2002] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10(2):99–127, 2002.

[Stone *et al.*, 2016] Peter Stone, Rodney Brooks, Erik Brynjolfsson, Ryan Calo, Oren Etzioni, Greg Hager, Julia Hirschberg, Shivaram Kalyanakrishnan, Ece Kamar, Sarit Kraus, Kevin Leyton-Brown, David Parkes, William Press, AnnaLee Saxenian, Julie Shah, Milind Tambe, and Astro Teller. "Artificial Intelligence and Life in 2030." One Hundred Year Study on Artificial Intelligence: Report of the 2015-2016 Study Panel. Technical report, Stanford University, Stanford, CA, 2016.

[Tommasino *et al.*, 2016] Paolo Tommasino, Daniele Caligiore, Marco Mirolli, and Gianluca Baldassarre. A Reinforcement Learning Architecture that Transfers Knowledge between Skills when Solving Multiple Tasks. *IEEE Transactions on Cognitive and Developmental Systems*, X(X):1–1, 2016.

[Vapnik, 2015] Vladimir Vapnik. Learning Using Privileged Information : Similarity Control and Knowledge Transfer. *Journal of Machine Learning Research*, 16:2023–2049, 2015.

[Vezhnevets *et al.*, 2017] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal Networks for Hierarchical Reinforcement Learning. mar 2017.

[Wang *et al.*, 2016] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. nov 2016.

[Zaremba and Sutskever, 2015] Wojciech Zaremba and Ilya Sutskever. Reinforcement Learning Neural Turing Machines. *Arxiv*, pages 1–14, 2015.

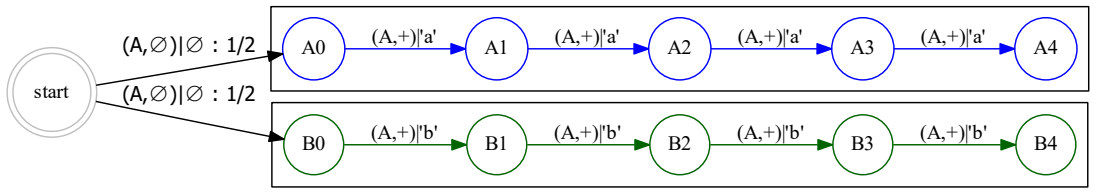
[Zhou *et al.*, 2012] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. Online Incremental Feature Learning with Denoising Autoencoders. *Journal of Machine Learning Research - Proceedings Track*, 22:1453–1461, 2012.

A Appendix

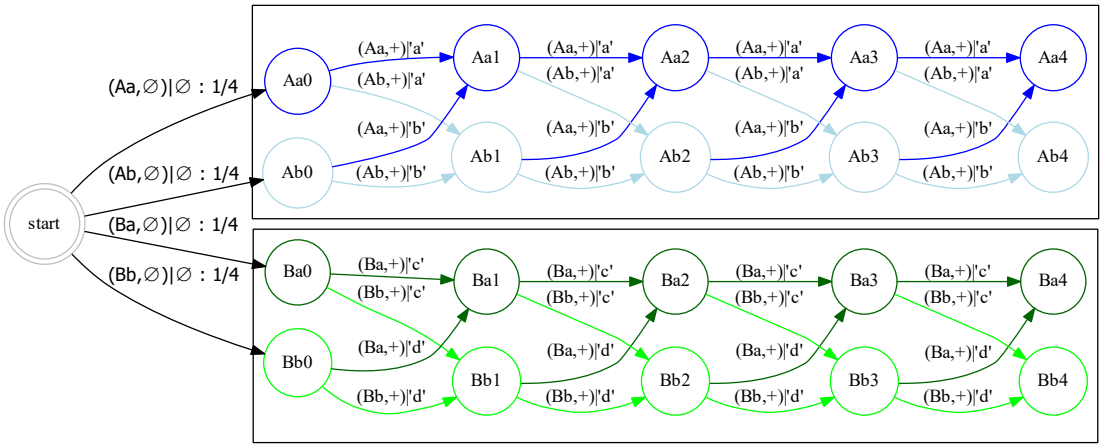
All figures in this section are simplified visualizations of ϵ-transducers, each for a particular task. These simplifications come in four flavors:

1. Only two instances, out of a numerous possible, are shown, as others would only be repetitions of an already presented pattern (black rectangles delineate instances).
2. For the sake of clarity not all transitions (arrows) are always visualized (e.g. error or instance switch transitions).
3. All transition probabilities from a state are uniformly distributed among visualized transitions. Only instance decision transition probabilities are shown for illustration.
4. In Figure 4, states are merged together, to avoid repetition when the environment is providing the current instance description. This way the visualization is still readable and focuses more on the important part – the part where the agent replies.

General notation for transitions (arrows) between causal states is as follows: $(o_{t+1}, r_t | a_t)$, where o_t is an observation/state that is the output of the environment, r_t the reward for the last performed action a_{t-1} by the agent. Note that the action a_t at current time-step determines the reward r_t at current time-step, but the observation/state for the *subsequent* step.



(a)



(b)

Figure 3: A simplified visualization of ϵ-transducers for two consecutive tasks A.2.1 (a) and A.2.2 (b). No error or instance switch transitions are shown. One can see the apparent possible sub-structures that the two tasks share, that could be exploited by a gradual learner. The *state* naming convention is as follows:

- First (capital) letter denotes an instance
- The number at the end of the state descriptor represents the number of consecutive positive rewards the agent has received
- The middle letter (only in the Subfigure b) defines from which group a letter was provided by the environment for the next step

For these visualizations, five consecutive positive rewards are required to complete an instance (fully shown in Figure 5).

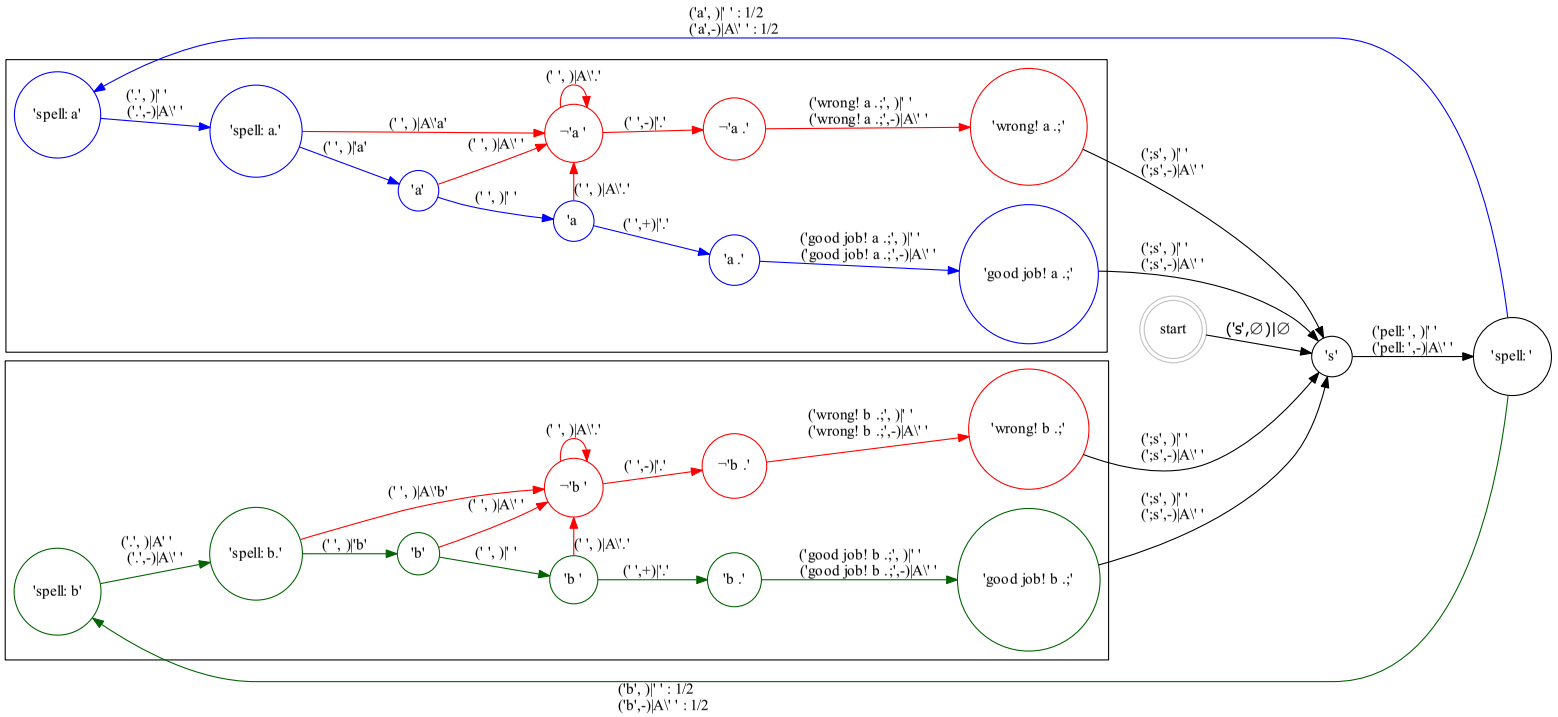


Figure 4: A simplified visualization of ϵ-transducer for task A.2.9.1 from the challenge curriculum.

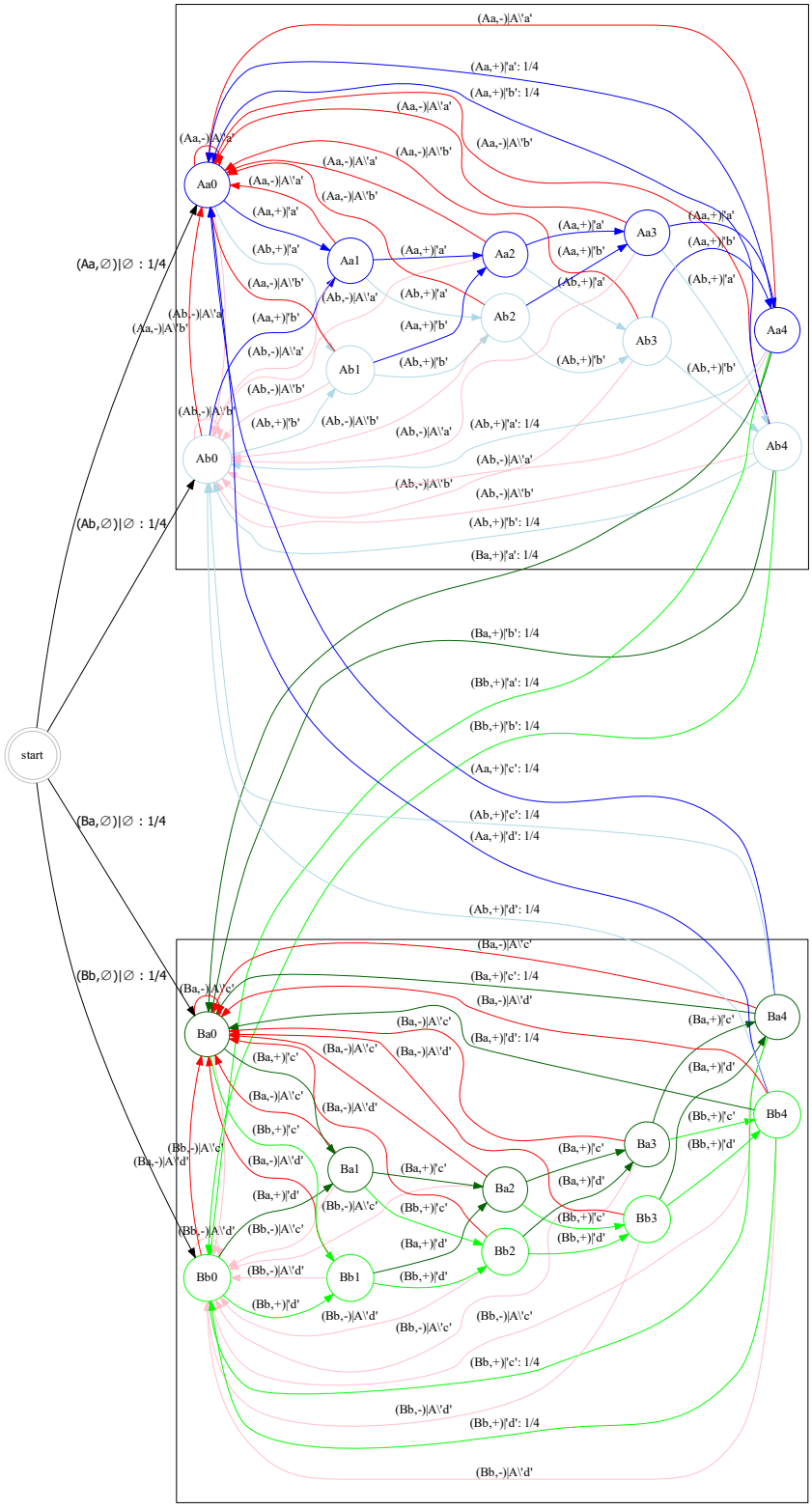


Figure 5: Full visualization of ϵ-transducer for task A.2.2 from the challenge curriculum. Same state naming convention applies as in Figure 3.