Combining Diverse Forms of Human and Machine Intelligence

by

Andres Campero Nuñez

Submitted to the Department of Brain and Cognitive Sciences
in partial fulfillment of the requirements for the degree of

PhD in Artificial Intelligence and Collective Intelligence

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

© Massachusetts Institute of Technology 2022. All rights reserved.

Author .....
Department of Brain and Cognitive Sciences
February, 2022

Certified by .....
Thomas W. Malone
Professor
Thesis Supervisor

Certified by .....
Joshua B. Tenenbaum
Professor
Thesis Supervisor

Accepted by .....
Mark Harnett
Graduate Officer, Department of Brain and Cognitive Sciences

Combining Diverse Forms of Human and Machine Intelligence

by

Andres Campero Nuñez

Submitted to the Department of Brain and Cognitive Sciences
on February, 2022, in partial fulfillment of the
requirements for the degree of
PhD in Artificial Intelligence and Collective Intelligence

Abstract

Artificial Intelligence algorithms never operate in isolation but are always part of broader processes that often involve humans, other computer algorithms, incentive structures, and interfaces which modulate the interaction between them. This thesis takes this perspective and considers these broader processes by studying specific combinations of three forms of intelligence: symbolic artificial intelligence, neural artificial intelligence, and human intelligence. First, diverse forms of Neuro-Symbolic AI through three pipelines consisting respectively of neural perception with symbolic reasoning, symbolic inputs with neural reasoning, and a dual-integration that learns representations which are simultaneously symbolic and neural (Chapter 2); second, the AI research community as a Human-Symbolic combination through the presentation of a taxonomy of AI models, tasks and datasets (Chapter 3); and third, a specific form of Human-AI intelligence observing that a human in combination with GPT3 can perform an HTML code generation task better than either humans or computers alone (Chapter 4).

Thesis Supervisor: Thomas W. Malone
Title: Professor

Thesis Supervisor: Joshua B. Tenenbaum
Title: Professor

Acknowledgments

I am very thankful and owe gratitude to many.

Thomas Malone and Josh Tenenbaum have constantly given me a vote of confidence. They have been understanding, patient, willing to take risks with me, and overall, always supportive in my off-the-beaten-path adventures.

Josh during the first part of my PhD. Such and influential and foundational figure. You helped me see the important questions. You inspired me through admiration.

Tom during the second. I feel very fortunate to have met you. It has been an incredible formative and fun opportunity to work with you.

I believe that in the coming century the greatest advances in human knowledge and the most radical transformations of life and society will come from a deeper understanding of the mind and of intelligence as computation. You both helped me grasp this and put it in perspective. Josh always bringing me back to Cognitive Science; Tom giving insight about the importance of Collective Intelligence.

I would like to thank Armando Solar-Lezama, who has been supportive in multiple dimensions, directly and also through his influence on Josh and his group. A great inspiring example as a Mexican.

I would like to thank Rebecca Saxe, always warm, supportive, intelligent, and most importantly, always kind.

Thanks to the all the Brain and Cognitive Sciences administrative team, specially Julianne Ormerod, Sierra Vallin, Meredith Canode, and Kris Brewer.

Thank you to all the BCS and MIT Faculty for this formative period, to James DiCarlo, Tomaso Poggio, Laura Schulz, Andrei Barbu, Susan Carey, David Kaiser, Michale Fee, Mehrdad Jazayeri, Nancy Kanwisher, Roger Levy, Josh McDermott, Pawan Sinha, Edward Gibson, Pattie Maes, Abdullah Almaatouq, and many others.

A special thanks goes to Jose Sotelo and Moran Cerf. This would have not been possible if it not were for you, in a very literal sense.

To all the researchers that directly contributed to the work in this thesis. Andrew Francl, Tim Klinger, Aldo Pareja, Haoran Wen, Raza Abbas, Jaeyoon Song, San-

tiago Renteria, Edward Grefenstette, Tim Rocktaschel, Hienrich Kuttler, Roberta Raileanu, Thomas Malone and Josh Tenenbaum.

Thanks to the Center for Brains, Minds, and Machines, to the Toyota Research Institute, to the MIT Center for Collective Intelligence and the MIT Quest for Intelligence, to London Facebook AI Research, and to the MIT-IBM Watson AI Lab. They supported many of the projects in this thesis.

To the members of the impressive Cocosci community at MIT with whom I had recurrent interactions: Luke Hewitt, Max Kleiman-Weiner, Pedro Tsividis, Kevin Ellis, Amir Arsalan Soltani, Max Nye, Max Siegel, Tobias Gerstenberg, Tom Silver, Kelsey Allen, Cathy Wong, Vikash Mansinghka, Bernhard Egger, Josh Rule, Jiajun Wu, Ilker Yldirim, Tomer Ullman, Chris Baker, Timothy O'Donnell, Tejas D. Kulkarni, Alex Lew, Feras Saad, Tan Zhi-Xuan, Junyi Chu, MH Tessler, Jon Gauthier, Tyler Brooke, Joao Loula, Matthias Hoffer, Felix Sosa, Joey Velez, and many others.

Thanks to the people who enriched my experiences at UCL and FAIR in London: Sebastian Riedel, Edward Grefenstette, Tim Rocktaschel, Hienrich Kuttler, Roberta Raileanu, Nantas Nardelli, Matko Bosnjak, Pasquale Minervini, Pontus Stenetorp.

A very special thanks goes to the Stateoftheart.ai team: Hugo Ochoa, Eduardo Espinosa, Jorge Delgadillo, Antonio Teran, Luis Lara, Liuba Orlova, Cuco Resendiz, Lynne Bairstrow, Luisfe Nuñez, Elena Elorza, Jesse Parent, Gussi Espinoza, as well as Alfredo Hong, Diego Acevedo, Joseph Puerner, and Ehecatl Antonio Del Rio.

To the ever growing RIIAA organizing team including Jennifer Enciso, Benjamin Sanchez, Rudy Corona, Enrique Garduño, David Alvarez, Alejandro Noriega, Liuba Orlova, Luz Angelica, Dhyan Adler, Andres Guevara, Mario Rosas, Victor Jimenez, and so many others. To Ehecatl Antonio Del Rio for being a life companion.

To all the people that have influenced my intellectual thoughts during my PhD: Andrei Barbu, Candace Ross, Martin Arjovsky, Pablo Sprechmann, Felix Hill, Meire Fortunato, Georges Belanger, Edgar Dueñez, David-Lopez Paz, Alex Kell, Yoshua Bengio, Alan Aspuru, Alejandro Noriega, Bjarke Felbo, Florian Hillen, Aldo Pacchiano, Judith Amores, Benjamin Sanchez, David Alvarez, Michael Chang, Natasha Jacques, Michael Janner, Ross Morphy, Andreas Haupt, Samuel Barnett, Ronen

Zilberman, Alexander Rush, Jorge Nocedal, Jesse Parent, Michiel Bakker, Morgan Frank, Edmond Awad, Iyad Rahwan, Gemma Roig, Georgios Evangelopoulos, Daniel Yamins, Daniel Zysman, the participants of the Learning Workshop organized by Martin Arjovsky and Cinjon Resnick, Jeremy Nixon, Lorenzo Aldeco, David Theurel, Daniel Low, Ehud Kalai, Martin Schrimpf, and many others.

Thanks to all my friends! The most fun has always been with friends.

To my friends at Northwestern: Gabriel Ziegler, Sergio Armella, Haritz Garro, Rene Leal, Laia Navarro, Alex Theisen, Ryan lee, Matthew Leisten, Britanny lewis, Victoria Marone, Carlos Cordova, Laura Garcia, Walther Maradiegue, Monika Podgorski, Roman Acosta, Jose Carreño, Benjamin Grant, Nil Karacaoglu, Ricardo Dahis, Kartikey Sharma. To Michiel de Jong, such an important friend for counsel, bouncing of ideas, and help to understand the AI field and life more generally.

To my BCS cohort and MIT friends: Heather Kosakowski, Alexi Choueiri, Matthias Hoffer, Eghbal Hosseini, Andrew Francl, Andrew Bahle, Elli Pollock, Mahdi Ramadan, Sabrina Osmany, Jenelle Feather, Daniel Estandian, Charles Choueiri, Sanjay Guruprasad, Lauren Fratamico, Brando Miranda, Frederico Azevedo.

To Isabel Goldaracena, que me apoyo tanto y me dio tanto cariño. To Casa Broadway for some extremely fun and inspirational years, what a ride. To Casa Prospect, you guys were my home. "To Genesis!". To Fuerza Mexico. To all the Mexicans in Boston: Daniel Young and Fiona Aguilar, Aaron Sonabend and Vicky Levy, Juan Pablo Rodriguez, Cheko Cantu, Claudia Varela, Jorge Buendia, Bernardo Garcia Bulle, Julie Ricard, Alvaro Farias, Leila and Karim Pirbay, Ilaria Ricchi, Bruno, Luis Torres, Pablo Ducru, and so many others.

To all the friends that have supported me during the later period including Jose Pablo Zarco, Mariana Acevedo, Jose Manuel Cuevas, Mercedes Saldaña, Diana Medina, Daniel Herrera, Montserrat Avila, Dhyan Adler, and Alfredo Carrillo[2].

Thanks to the example of my mother Rosa Maria Nuñez, to the support of my father Alberto Campero, to the love of my (big!) family - all my cousins, aunts, uncles, and grandparents - and to my Nakama and brother Javier Campero.

For much to come, Gracias!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the recent progress of Artificial Intelligence (AI), there is increasing interest in creating superintelligent computer algorithms in a wide range of tasks. Despite these successes, an often overlooked fact is that these models never operate in isolation. An alternative perspective recognizes that in practice software is always used as part of broader processes that often involve humans, other computer algorithms, incentive structures, coordination mechanisms, tools, and other interfaces which modulate the interaction between them. In this thesis we take this perspective and study these broader processes that combine diverse forms of human and machine intelligence.

The thesis considers different combinations of three forms of intelligence: symbolic artificial intelligence, neural artificial intelligence, and human intelligence. Each of its three chapters studies particular instantiations of a combination of two of those forms: three pipelines of Neuro-Symbolic AI first, a Human-Symbolic organization of the research community in the form of a taxonomy second, and a proposed form of Human-AI (neural) Superintelligence third.

While there is increasingly broad convergence among scientists and engineers that the mind can in principle be replicated and understood in algorithmic terms, the specific nature of intelligence is controversial. Two broad perspectives exist with an ongoing debate that has parallels in both Cognitive Sciences and the Artificial Intelligence. In Machine Learning and AI, the symbolic form of intelligence emphasizes the role of algorithms that operate over symbols which work as interpretable labels

that can represent more complex operations. For example, in most programming languages, a 'function' has a name or symbol which can be 'called' to execute several lines of code. The second form of intelligence, which we term neural, takes loose inspiration from the biological brain and is based on artificial neural networks which can learn to solve problems based on adjusting multiple, usually non-interpretable 'weights'. This second form became what is now the field of Deep Learning. Similarly, the study of human concepts by cognitive scientists have historically also been divided into schools of thought which emphasize either the conceptual more symbolic role, where the meaning of a concept comes from its relationships to other concepts; or the more statistical role which posits that the meaning of a concept doesn't come from its definition, but from its statistical relationship with typical examples experienced in the world [23, 65].

The human mind has elements of both perspectives. These two forms of intelligence are strikingly complementary in their strengths and weaknesses. Symbolic approaches are compositional, highlight the richness of conceptual role, and require little data to train [45]. On the other hand, neural approaches require less explicit structure, and have been very successful at performing different tasks across many domains [59, 77, 123, 16]. The field of Neuro-Symbolic AI has been growing richer in the last years [33, 42]. **Chapter 2** presents three sections which study different specific 'pipelines' or ways of combining the two perspectives in the Neuro-Symbolic AI context: section 2.1 does neural perception first and symbolic reasoning next. Section 2.2 attempts a tight dual-integration, learning representations which are simultaneously symbolic and neural. Section 2.3 deals with symbolic inputs but realizes neural reasoning.

Chapter 3 considers the whole system of Artificial Intelligence algorithms and human researchers as a particularly powerful collectively intelligent Supermind [70]. Inspired on [71], the chapter develops a comprehensive taxonomy of Artificial Intelligence models, tasks, and datasets. The taxonomy constitutes a classical form of a symbolic organization of information and the AI research community itself is a human-machine combination of two forms of intelligence.

Chapter 4 studies and defines a specific form of Human-AI Superintelligence: the ability of the human-computer combination to perform tasks many times better than either humans or computers alone. We explore the task of code generation for the context of website replication in HTML, measuring performance in terms of speed subject to constraints on quality. We explore the combination of GPT-3 [16], a state-of-the-art neural language model in combination with the a human (for two populations of coders and non-coders) utilizing an interface for dragging and editing code. We show this combination of neural machine intelligence and human intelligence improves performance and, in some cases, is able to obtain Superintelligence.

Chapter 2

Neuro-Symbolic AI

2.1 Hierarchical Bayes and Deep Learning

2.1.1 Introduction

Recent advances in neural networks and other machine learning methods have led to computer vision object-recognition systems that are beginning to approach human-level performance. Trained on thousands of object categories, with thousands of labeled examples for each, deep convolutional networks can tell if a new image contains a familiar category almost as well as human adults can in a brief glance. Yet, even young children have abilities to learn and generalize that go beyond what current machine vision systems can do. Here we focus on three such abilities:

(1) By age 3, children can learn new object categories from just a single example. Furthermore, children generalize in different ways as appropriate for different kinds of categories: labels for artifacts with functionally relevant shapes are preferentially generalized according to those shapes, while labels for non solid substances or arbitrarily shaped objects are more likely to be generalized according to material properties.

(2) Children can learn to learn appropriate inductive biases, such as the shape and material biases described above, from experience with just a few examples each of a small number of categories that exemplify these biases in a consistent way. The shape-bias training studies of Smith and colleagues are the best known examples

[112].

(3) Children can, in a completely unsupervised way, sort novel objects into categories and supercategories in a meaningful way, and then use these hierarchical category structures as strong constraints to learn and generalize names for objects from just one or a few examples.

Previous attempts to capture these abilities in computational models have had some success, but not with models that are "image-computable" on the same stimuli that people see. These earlier models have used either adult similarity judgments [125] or highly simplified, idealized feature representations [55] to build their category hierarchies. Here we show that a computational framework can come close to capturing abilities (1-3) by combining two powerful representation-learning techniques: deep learning for feature construction and Hierarchical Bayes for unsupervised taxonomy construction.

We build on work by [99] who build a Hierarchical Bayesian model that "learns to learn" by incorporating information from past experience into a prior when inferring statistical properties of a novel category. In particular, when presented with a few image examples of a new category, the model infers a supercategory and uses the higher-order knowledge abstracted from previous categories to identify the relevant features and allow generalization (Figure 2-1).

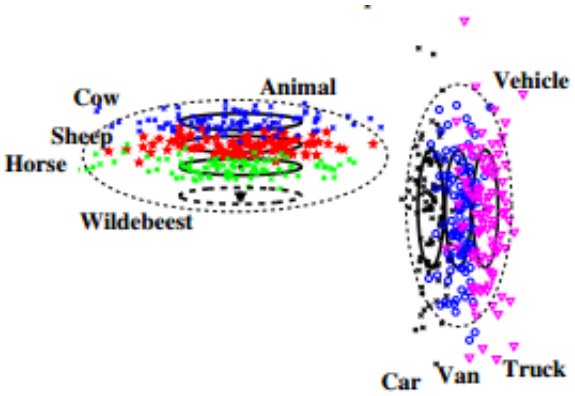


Figure 2-1: Learning a similarity metric for a new category. The goal is to identify the correct supercategory and estimate an appropriate similarity metric.

That work was extended by the same authors, who harnessed a two layer Deep-

Boltzmann Machine to generate low level feature representations of the images while learning a prior using a hierarchical Dirichlet process. [100]. Their experimental data showed that using this prior in combination with more powerful features gave them a distinct advantage over other methods of classification. This progression of work suggests that building a model that combines complex feature spaces with a hierarchical semantic structure may lead to further increases in performance.

Building on this line of work, we contribute a model that combines the two components: powerful image representations extracted from Deep Neural Networks (DNNs) and a Hierarchical semantic structure that works as a Bayesian prior. We show how the combination of these two components can "learn to learn" in ways that resemble some aspects of child cognition. Additionally, we explore how this model's performance is affected as we vary different aspects of the model architecture and the structure of the training data.

Other approaches to combine probabilistic graphical models and DNNs have recently been proposed that focus on building unsupervised clustering algorithms (Dilokthanakul et al., 2016; Johnson et al. 2016). Instead, the focus of our model is to capture certain aspects of human cognition. This leads to some notable differences. First, representations in our model are a fixed set of visual relevant features instead of being learned for the inference task at hand. In addition, our model's generative component is limited to a hierarchical structure that aims to recover the semantic relations between concepts in a useful and meaningful way while other models are fully generative but tend to have graphs with simpler semantic structures. We therefore propose a relatively simple model that is not intended for general unsupervised learning but that instead focuses on traits of human object and category learning.

More specifically, we test our model's capacity to capture the previously discussed human abilities (1-3) in an image recognition framework. First, we evaluate the ability of our model to learn novel categories from only one or a few examples. To address this we allow the model to construct a semantic structure from labeled examples in a data set and then judge the model's performance on a one-shot learning task. Second, we assess the models capability to construct inductive biases in low data environments.

We test this ability by repeating the first task but limiting the training data available to the model when it constructs the semantic tree. Finally, in a third task, we test the model's ability to learn a hierarchical semantic structure of novel objects in a completely unsupervised manner. Results suggest that this approach may be suitable for modeling certain aspects of cognition.

2.1.2 Model and Learning to Learn

Our model combines two Machine Learning approaches that have recently been successful at a range of differing tasks. On one hand, powerful deep networks construct feature spaces that enable rapid and accurate classification. On the other, Hierarchical Bayesian Models have proven successful in creating taxonomies of the different concepts learned from previous experience. These taxonomies can then be used as a prior to identify the relevant features for learning a new category from one or a few examples based on the distribution of other similar categories. We create various versions of our model to compare combinations of feature spaces extracted from different architectures with variants of the Hierarchical Bayesian component.

Learning begins by constructing a 2-level tree of categories and supercategories that best explains the training observations under a Bayesian framework. The model learns structure in the observations by first generating useful general features from a DNN and then developing hierarchical priors that allow previous similar experiences to bias the learning of new concepts and categories. The priors are constructed by inferring the means and variances that define the most relevant dimensions from the DNN feature representations for each category and supercategory (Figure 2-1).

Deep Network Features

We use features extracted from DNNs pretrained for object classification on ImageNet. We obtain a representation from each image by passing it through a network and extracting the response from the penultimate layer consisting of 4096 real-valued dimensions. In the regular deep network classification scheme, this response is then

passed through a linear weighting and a generalized logistic regression layer. This layer maps this representation onto probabilities for each class in the specific classification task for which the network was trained.

We compare the performance of the different versions of our model on features extracted from two different DNN architectures: Alexnet [59], which was the first implemented Deep Learning Model that significantly improved object classification on images; and VGG-16 [110], a more recent architecture with 16 layers that achieves above 90% top 5 classification performance on ImageNet.

Generative Semantic Organization

After obtaining a useful general image representation from the DNN, the Hierarchical Bayesian Model's parameters are inferred by approximating the posterior via Markov Chain Monte Carlo methods in the following way.

Consider a two-level hierarchy where N observed inputs are partitioned into C basic-level categories, these categories are in turn partitioned into K supercategories. In this hierarchy of observations, categories, and supercategories, the higher levels determine a prior over the distribution of the lower levels. In particular, the distribution over observations (feature vector representations of images in our case) of each of the different basic level categories are assumed to be multi-variate Gaussian with a category specific mean M_c and with precision terms τ_c^d that are assumed to be independent across the D dimensions of the feature space. These precision terms constitute a similarity metric by determining the relative importance of each of the features. In turn, we place a conjugate Normal-Gamma prior over $\{M_c, \tau_c\}$, this prior is determined by the supercategory specific level-2 parameters M_k, τ_k, α_k, where M_k and τ_k constitute the expected values of the lower level parameters and α_k controls the variability of τ_c around its mean. Finally, for the conjugate priors over the level-2 parameters, we respectively assume Normal, Exponential and Inverse-Gamma distributions that are further shaped by parameters α_0 and γ_0. The full generative model is given in Figure 2-2 [99].

Given a set of observations, the model iteratively performs Bayesian inference by

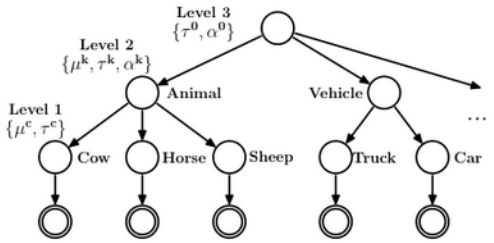


Figure 2-2: Hierarchical Model

alternating between sampling the parameters and inferring the category assignments. When learning the distributions at each step of the iteration, the supercategory membership is fixed and the parameters are sampled from posteriors that are analytically computed using the conjugate priors[1]. The supercategory membership for each category is learned in a similar way by fixing the currrent parameters and the rest of the hierarchical structure. Every category can be assigned to any of the existing supercategories or to a newly created one. The posterior probability of belonging to a supercategory is computed as a combination the likelihood that the parameters of the category come from the parameters of the supercategory and a Chinese Restaurant Process (CRP) prior [46]. This nonparametric prior is a distribution over a partition on integers in which the n^{th} number is assigned to set k with probability:

$$P(z_n = k|z_1, z_2..., z_{n-1}) = \begin{cases} \frac{n^k}{n-1+\gamma} & \text{if } n^k > 0 \\ \frac{\gamma}{n-1+\gamma} & \text{if k is new} \end{cases}$$

Where n^k is the number of previous integers assigned to set k and γ is a concentration parameter sampled from a $Gamma(1, 1)$ distribution.

In an unsupervised setting where the categories of the observations are also unknown, the model utilizes a similar strategy to assign observations to categories as is used when assigning categories to supercategories. The model iterates through

[1]For the case of α_k, the conditional posterior cannot be computed analytically and the parameter is sampled with the Metropolis-Hastings rule [129].

the observations and assigns each either to an existing or to a newly created category based on the prior and likelihood. By utilizing the CRP prior, the model can create an unbounded number of categories and supercategories. This entire process constitutes a Gibbs sampling procedure where both the tree structure and all of the parameters are simultaneously learned.

2.1.3 Tests and Results

We test the model in scenarios that attempt to capture aspects of human cognition related to learning from limited data. First we measure the model's ability to generalize previous knowledge to learn novel categories from only a few examples. Next, we assess the model on this task when the training data for all of the categories is also limited to only a few examples. Finally, we exploit the model's full hierarchy in a completely unsupervised setting by exploring how the model recovers the underlying semantic structure.

One-Shot Learning on MSR

In the first task, we test the model's ability to learn new categories form one or a few examples. First, we select a category that will be held-out for testing. Labeled observations for all other basic-level categories are provided for training. The model learns the semantic structure of the training set by clustering the basic categories into supercategories and inferring the relevant parameters at all levels of the Bayesian Hierarchy. The challenge is then to generalize the learned structure to the held-out category from only one or a few examples.

To do this, the model first infers the best supercategory from one or a few examples of the withheld category by marginalizing over the category level parameters. Next, the model uses the supercategory priors and training examples to estimate the category similarity metric and mean for each dimension in the feature space.

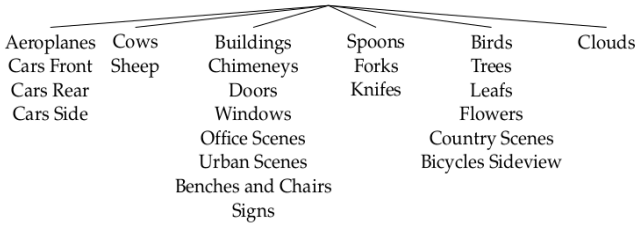


Figure 2-3: MSR semantic tree discovered by the Full Model

We evaluate different versions of our model on the MSR Cambridge dataset [58], which consists of 24 categories with varying numbers of images in each category. In total this dataset contains roughly 800 images. Figure 2-3 shows a typical partition over all the categories discovered by the full model. To quantify the models accuracy, a testset with unlabeled data from all categories is classified.

We repeatedly trained the model withholding one of the categories at a time and then inferred the withheld category parameters and supercategory membership using one or a few images. Next, we calculated the posterior probability for each testset image belonging to each category and variated a threshold to classify images as belonging to the heldout category or to any of the other categories. This created true and false positive rates for each point along our threshold which traced out a Receiver Operating Characteristic curve (ROC) for classifying objects from the withheld vs. all the other categories. The reported results are calculated by averaging the Area Under the ROC curve (AUROC) for the model trained with each of the 24 categories withheld (Table 2.1).

Table 2.1: Performance results using the area under the ROC curve (AUROC) on the MSR dataset in the one-shot learning task

	# Examples from Withheld Class							
	Alexnet				VGG			
	1ex	2ex	4ex	20ex	1ex	2ex	4ex	20ex
Oracle	.99	1	1	1				
HB-Full	.91	.96	.98	.99	.92	.97	.98	.99
One Supercategory	.87	.94	.97	.99	.88	.95	.98	.99
NearestN	.84	.86	.87	.90	.89	.90	.92	.95
T of T*	.76	.80	.84	.87				

Performance is compared for each combination of an Inference Model and a Network Architecture. HB-Full is the full version of the model described above. One Supercategory places all the categories in the same single supercategory. NearestN classifies new points with the label of the nearest neighbor of its feature vector in euclidean distance. Texture of Textures (T of T)* replaces our DNN features with the set of responses from a three layer convolutional neural network that uses precomputed weights that resemble Gabor filters[2]. Finally, the Oracle is the same than our full model, but uses the true empirical mean and variances from the whole population (including testset). Table 2.1 shows the results for the two different feature spaces used.

Limited training data regimes

In a second task, we test the capability of our model to extract inductive biases from experience with just a few examples. To evaluate this capability, our full model was limited to only 1, 4, 10 or 18 examples of each category used for training. The number of examples from the withheld category was varied separately. Table 2.2 shows the average AUROC for the same "one vs. all" metric used in the previous task[3]. For comparison, the full model performance from the previous table is included and labeled as "All examples"[4].

Table 2.2: Performance results using the area under the ROC curve (AUROC) on the MSR dataset with limited training data.

	# Examples from Withheld Class							
	Alexnet				VGG			
	1 ex	2 ex	4 ex	20 ex	1 ex	2 ex	4 ex	20 ex
# Training Examples								
1 ex	.87	.87	.88	.89	.90	.90	.90	.92
4 ex	.92	.96	.99	.99	.93	.97	.98	.99
10 ex	.92	.96	.99	.99	.92	.96	.98	.99
18 ex	.92	.95	.98	.99	.91	.96	.98	.99
All examples	.91	.96	.98	.99	.92	.97	.98	.99

We can see that the largest jump in performance happens when moving from 1

[2]Taken from [99]

[3]Averages across 10 random repetitions and all categories are reported.

[4]Each category contains a varying number of examples

to 4 training examples. This likely reflects the fact that a single example provides information about the mean of the category but not about the variance or similarity metric, which has to be inferred completely from the prior. However, 4 examples provide adequate information about the variance to allow the model to appropriately infer the parameters for new categories. As the number of training examples continues to increase, there are no further gains in performance. This is consistent with literature showing that children need at least two examples to learn inductive biases in certain contexts [112].

Unsupervised Learning on Gazoobian Objects

Humans and children can sort new objects into categories and supercategories in a semantically meaningful way. While our model is also able to of recover meaningful structure from labeled examples (Figure 2-3), real situations often demand learning where labels are completely absent. [103] explores this human capability with a dataset composed of 45 novel objects that were generated using a modeling software to simulate a specific taxonomic structure. The dataset consists of three supercategories supposed to be alien equivalents of plants, tools and snails from the planet "Gazoob". The objects in each supercategory are further organized into a structure that can be approximated by basic-level categories (gray box in Figure 2-4).

Our model has the ability to infer both categories and supercategories in an unsupervised manner from observations. [103] shows that a model based on agglomerative clustering that uses adult similarity judgments is able to recover the taxonomic tree (Figure 2-4). Here our model is tested with the harder task of recovering the taxonomic tree directly from the same images that people saw. The model accomplishes this task in a fully unsupervised manner using a single image of each object.

This "image-computable" model is able, although with some mistakes, to recover the three supercategories and most of the basic-level category structure (Figure 2-5). Other unsupervised clustering algorithms were also able to capture some of the semantic structure, but the hierarchy between categories and supercategories was not evident.

Figure 2-4: Ground Truth Tree of Gazoobian Objects as Generated from Human Similarity Judgments. Each of the three branches at the top of the tree denotes a supercategoy. The gray box in the lower left hand of the figure denotes a basic-level category.

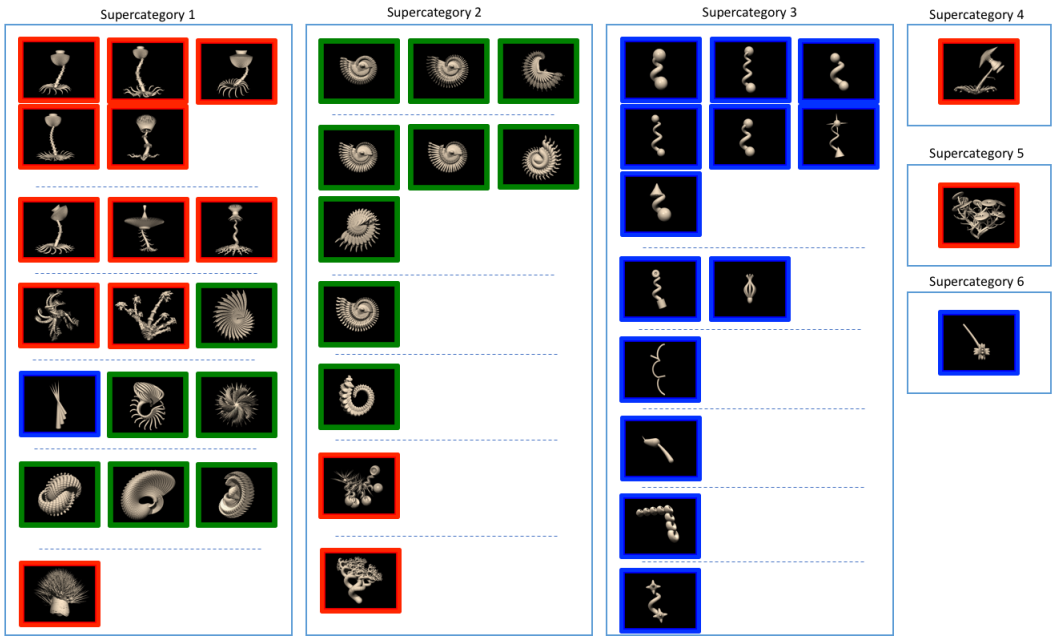


Figure 2-5: Model's Inferred Semantic Hierarchy of Gazoobian Objects. Outer boxes denote supercategories inferred by the model. Dashed lines separate model generated categories within each supercategory. Colored boxes around each object denote the ground truth supercategories as shown above.

2.1.4 Discussion

One can think of the task of concept learning as consisting of two elements. The first involves obtaining relevant features to represent the objects and categories commonly observed in the world. The second involves constructing a semantic hierarchical structure with links between categories that humans can use to navigate and perform tasks. While recent results demonstrate the capabilities of DNNs to classify categories provided a large number of training examples, they struggle to perform tasks that require understanding the semantic relationships between classes. The ability of Hierarchical Bayesian Models to build these semantic structures can further help with understanding and classifying new categories.

We demonstrate how these two approaches can complement one another by combining them in a computational model. We tested the model's abilities tasks designed to approximate human capabilities that are currently difficult for computer vision systems such as concept generalization, learning inductive biases, and constructing semantic structures. We show results for three tasks involving limited data availability. The model is able to learn relevant semantic structures from just a few examples of novel objects and effectively transfer appropriate similarity metrics from learned categories in the form of a prior. In all tasks, the computational framework comes close to capturing human abilities that other, more complex, machine vision systems struggle to reproduce.

2.2 Logical Induction and Distributed Representations

2.2.1 Introduction

Humans are continually acquiring, representing, and reasoning with new facts about the world. To make sense of the vast quantity of information with which we are presented, we must compress, structure and generalize from what we experience. This allows us to quickly understand new concepts and make useful predictions about them. For example, we might represent our knowledge of animals in a taxonomic hierarchy like that shown in Figure 2-6. Using such a hierarchy coupled with an inheritance rule that specifies that the attributes of higher nodes are shared by lower ones, we can achieve exponential compression over a representation which just lists the facts. Even more exciting, it allows us to infer a whole range of new facts about an individual simply by observing where it fits in the hierarchy. For example, observing that a Harpy Eagle is a type of Eagle allows us to immediately deduce that a Harpy Eagle can fly and breathe.[5] But how can such representations be learned from raw observations? This has been a key problem in semantic knowledge acquisition going back to at least to the 1960's in the work of [27], with symbolic, Bayesian, and neural approaches proposed [96, 49, 127]. In our view (and following [127]) there are three questions to be addressed in the development of a solution: (1) how can we induce logical rules from the observations? (2) how can we learn a small set of core facts (the taxonomy in the example) from which we can infer the observations (and more), and (3) how can this be done without explicit supervision on the structure of the rules?

In this paper we propose a model which can be used for both Inductive Logic Programming (ILP) and theory acquisition/compression. The network is neuro-symbolic in the sense that it represents predicates for both rules and facts using dense vectors which can be trained using gradient descent towards representations of known

[5]There are some kinds of reasoning which are not easy to do with a taxonomy (for example, handling the exception that penguins are birds but don't fly) but our proposal is not limited to taxonomic representations.

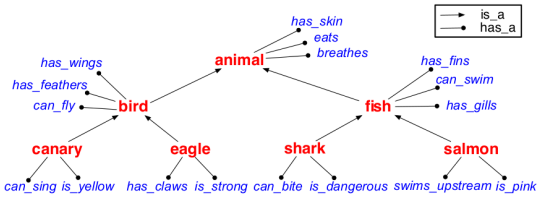


Figure 2-6: Animal Taxonomy. Constants are in red and blue, relations are indicated with lines and arrows.

predicates (including a fixed set of anonymous invented predicates). The network implements forward chaining and soft unification. [6]

For ILP problems, the network is given a set of "proto-rules" (rules with randomly initialized predicate parameters) and applies them using forward chaining to the background facts to produce consequent facts. After K steps of forward chaining (K is a hyper-parameter) the consequent facts are compared to the labeled target facts and the rule predicate parameters are trained towards representations of the predicates which yield all the true target facts and none of the false ones. Representations can be learned either for the known predicates used in the facts or for auxiliary invented predicates.

In theory acquisition/compression, the network is given a set of fact observations and asked to learn a logical theory – a set of core facts and a set of rules – which together entail the observations. The ability to learn facts is an aspect that has not been emphasized in many ILP approaches but is present in the Bayesian literature. For example, when observing that salmon can swim, have fins and have gills, the model can learn the core fact that salmon are fish even though that is not deducible directly. By encouraging sparsity in the set of core learned facts with a penalty term in the loss, the model can be trained to try to minimize the size of the theory it learns.

The remainder of the paper is structured as follows. In the next section we provide background material and related work. Then we present our model and training procedure. In Section 4 we present experiments which investigate different capabilities of the model and demonstrate the efficacy of our approach on a variety of ILP rule

[6]Soft unification relaxes the requirement that two predicate symbols must be identical for the rule to be applicable instead favoring a measure of the degree of similarity.

induction and domain theory learning datasets. We conclude with a discussion of limitations and future directions.

2.2.2 Background and Related Work

There is a rich literature on neuro-symbolic induction to which our approach is related on two main lines: inductive logic programming (ILP) and semantic cognition. ILP systems try to learn a set of logical rules which can be used to deduce facts of interest while semantic cognition is broadly concerned with how human beings acquire, represent, and integrate knowledge.

Inductive Logic Programming

In ILP, the goal is to learn (induce) logical rules which can be chained to successfully answer queries about a target relation, given positive and negative examples of that relation and some background facts. Logical rules are of the form:

$$h \leftarrow b_1, b_2, \ldots, b_k \tag{2.1}$$

where h is an atom called the *head* of the rule and $b_1, b_2, \ldots, b_k$ are atoms which constitute the *body*. An *atom* is a positive or negative literal. A literal is a predicate applied to terms which may, in our case, be either variables or constants. For example $\texttt{grandfather}(X, Y)$ is a (positive) atom whose predicate is $\texttt{grandfather}(\cdot, \cdot)$ and whose arguments (two in this case) are variables X and Y. When the arguments of the atom are all constants (e.g. `parent(Tom, Bill)` for constants `Tom` and `Bill`) we call it a *ground atom* which we also refer to as a *fact* when it is given as true or its truth value is inferred from rules. Intuitively, the head of the rule is true if each of the b_i in the body are. For example a rule might be:

$$\texttt{grandfather}(X, Y) \leftarrow \texttt{father}(X, Z), \texttt{parent}(Z, Y)$$

Which is read: *X is the grandfather of Y if X is the father of Z AND Z is the*

parent of Y. The head atom holds for any X and Y as long there is some individual Z for which `father`(X, Z) and `parent`(Z, Y) are both true facts.

Given background facts like `father(Bill, Mary)` and `parent(Mary, Liz)`, logical rules can be chained together to prove a goal fact like `grandfather(Bill, Liz)`. This is an example of a *forward chaining* deduction because it starts from a set of facts and unifies (matches) them with the body of a rule to derive the consequences. It is also possible to do *backward chaining* in which we start with a goal and work backwards by unifying it with the head of a rule, recursively trying to prove the body.

Symbolic ILP systems have a rich history dating back decades. A common approach to inducing rules is called *learning from entailment* [30], in which hypothesized rules are combined with background facts and trained to entail the positive and none of the negative concept examples. The FOIL algorithm [89] is an example of this approach. Our approach is also of this sort though we use neural networks to learn the rules. The classic ILP setting has been continually updated to handle richer knowledge. In [29] for example, they provide several formulations of probabilistic ILP. We do not consider probabilistic interpretations in our approach, though that is an interesting avenue for future research.

A related branch of research called Abductive Logic Programming attempts to learn consistent explanatory facts as well as rules [52]. For example, it might allow us to induce the fact that eagles are birds from the facts that eagles have wings and feathers together with the inheritance rule. Our approach for theory acquisition may be considered an example of this line of work.

Neuro-Symbolic Integration

Symbolic ILP systems do very well at generalizing from just a few examples. This is because they learn universal rules. They are, however, susceptible to noisy inputs and even a single bad fact can cause them to fail. On the other hand, neural systems generally are very robust to noisy input but are sample inefficient and prone to overfitting on small amounts of data. Neuro-symbolic systems aim for the best of both worlds. They can be made robust to noisy inputs while still retaining some of the

strong generalization properties typically associated with symbolic systems.

There is a long history of research in neural-symbolic systems from which we choose just a small set to present here. For a recent survey see [10].

In [106] they introduce Logic Tensor Networks (LTN) and Real Logic whose semantics grounds the terms, atoms and clauses of the language as continuous functions. They demonstrate that the logic can be implemented using neural networks for the groundings of the symbols and apply it to solving a database completion problem. A follow-up paper applies LTN's to semantic image segmentation [31]. Neither work considers the problems of rule induction or theory acquisition.

A recent work [72], starts from a probabilistic logic programming language (problog) and extends it to handle neural predicates which compute probabilities. Like Prolog and ProbLog, DeepProbLog is a backward chaining approach. It leverages the automatic differentiation system of ProbLog to incorporate neural predicates and trains with gradient descent. Like Logic Tensor Networks, ProbLog can train neural network implementations of relations. ProbLog does not do rule induction.

In Neural LP [126], the system can learn chaining-type rules. It uses a neural controller built on top of TensorLog [26] and is trained to learn rules to compute a ranked list of entities which satisfy a partially specified query. It differs from our approach in several respects. It requires a partially specified query. It represents predicates as TensorLog operators (matrices) whereas we represent them as parameter embeddings which can be associated with constants and a valuation to represent an atom. And it is not obvious how it could be applied to learn fact representations.

Sourek et al. [114] uses templates to create grounded networks that depend on the example. Tran and D'Avila Garcez [118] studies the incorporation and extraction of knowledge into deep networks. Kazemi and Poole [54] focuses on predicting the properties of objects.

Our approach is most directly related to two recent neural ILP approaches. In [36] inference is done through the forward chained application of a set of logical rules. During learning, a set of all the possible candidate rules is generated according to a provided template. Parameters are weights associated with pairs of candidate

rules. These weights are normalized to lie in $[0, 1]$ and interpreted as probabilities associated to the rule pairs as possible definitions of the concept. When there are a large number of rules, this method may suffer scalability issues. In addition it requires a representation of the truth values of all possible facts and non-facts. By contrast, [95] construct a function representing a backward-chained proof of the goal and require only a representation of the true facts. A more conceptual distinction arises in their parameterizations. In [36] the parameters are weights on rule pairs. In [95] they start with a set of parameterized rules which, as in our approach, acquire their meaning as the predicate embeddings of their head and body atoms are trained through unification with the predicates of the facts. It is not obvious how these approaches could be applied to theory acquisition requiring fact induction.

In our approach we follow [95] in parameterizing with embeddings but use forward rather than backward chaining so that we don't have to represent a proof tree explicitly. Unlike [36], we don't have to generate all the candidate rules. Instead, learning is at the level of individual atoms.

Semantic Cognition

Semantic cognition concerns the acquisition and integration of knowledge. Previous work has modeled semantic cognition as a kind of logical dimensionality reduction [127, 121], which uses probabilistic generative models that can simultaneously learn logical rules and a set of core relations that form a theory underlying the observed data. Like ILP approaches, these models can make deductive inferences through the application of logical rules. But unlike traditional ILP algorithms, these Bayesian models are also able to induce facts.

This ability to apply both inductive and deductive reasoning at the level of both facts and rules provides humans with a rich space of techniques with which to tame the complexity of everyday experience. These approaches illuminated a promising direction but were severely challenged by scalability issues. With the success of neural techniques we believe it is useful to revisit these ideas.

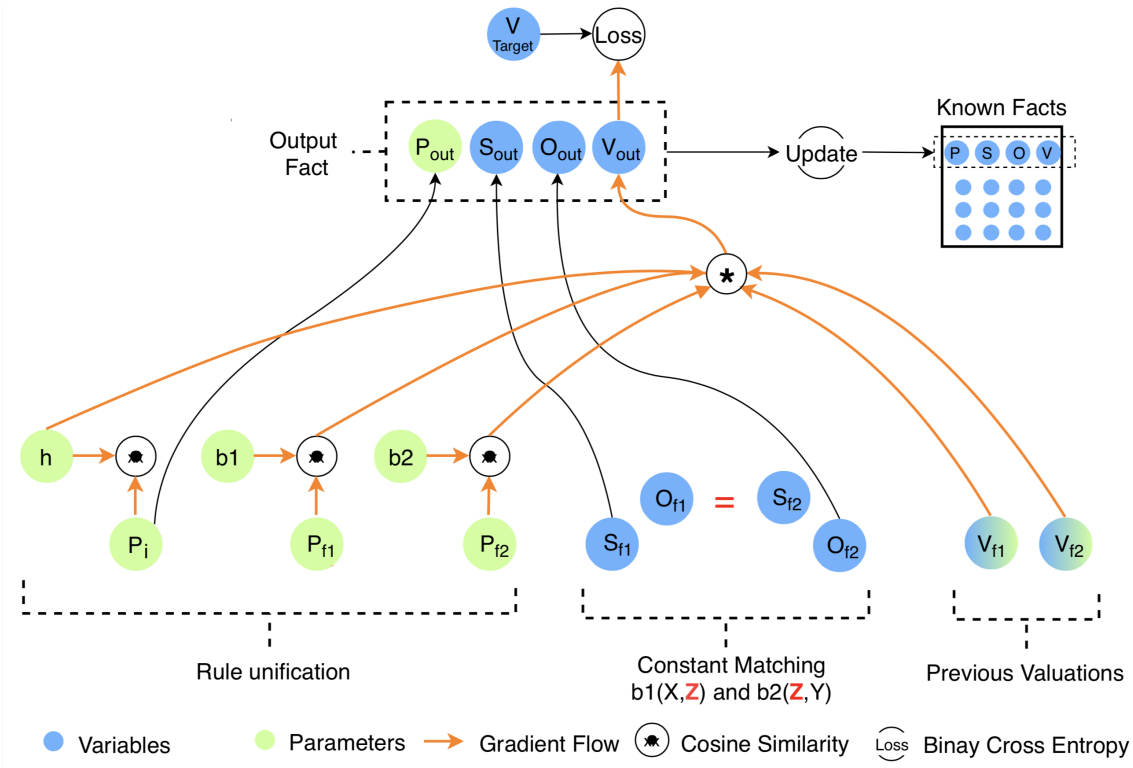


Figure 2-7: Overview of the model, a step of forward chaining. Parameters are represented in green and constitute the trainable embeddings, orange arrows indicate paths on which gradients flow (in the opposite direction).

2.2.3 The Model

As described above, our focus is on two tasks: rule induction in an ILP setting, and theory learning (learning both core facts and rules). In this section we describe a model which can be configured to perform either task and is trained using stochastic gradient descent. Figure 2-7 illustrates the architecture common to both tasks. A rule is shown on the bottom left with a head predicate h and two body predicates $b1$ and $b2$. The model represents the rule as a triple $((\theta_h, v_1, v_2), (\theta_{b1}, v_3, v_4), (\theta_{b2}, v_5, v_6))$. $\theta_{\mathbf{h}}$, $\theta_{\mathbf{b1}}$, and $\theta_{\mathbf{b2}}$ are parameterized embeddings in $\mathbf{R}^d$ corresponding to predicate symbols h, b_1, b_2; v_i denote variables X, Y, Z which are the subjects and objects that form the arguments of the atoms.

Facts are represented as quadruples (θ_p, s, o, v), where θ_p is a parameter vector shared by all the facts associated with predicate p; s is the subject constant; o is the object constant; and $v \in [0, 1]$ is a valuation representing the model's degree of belief

in the truth of the atom $p(s, o)$. Constants are internally represented as integers and may be mapped to symbols for interpretation. The set of all current known facts (either initially given or inferred) is shown in the figure at the upper right. We maintain a set of all predicates P and their associated embeddings which may include auxiliary predicates that can be used for predicate invention in the learning of rules and concepts. All predicates are randomly initialized.

To apply the rule to a pair of facts f_1, f_2, where $f_i = (\theta_{\mathbf{p_i}}, s_{f_i}, o_{f_i}, v_{f_i})$ we first check for constant matching between the structure of the rules and the constants of the facts. This is just checking that the variable arguments of the rule body can be assigned to the corresponding constants in the facts. If they cannot, then then network construction stops. For example, as in the figure, the rule $\texttt{grandfather}(X, Y) \leftarrow \texttt{father}(X, Z), \texttt{parent}(Z, Y)$ can be applied successfully to the pair of facts `father(Bill, Tom)`, `parent(Tom, Mary)`. However, if applied to `father(Bill, Tom)`, `parent(Anne, Mary)`, it would not match and would cause network construction to stop for this rule and fact pair. After a successful constant matching, a set of candidate output facts is generated, one for each predicate $p \in P$. The arguments of the fact are determined by the bindings of the rule body predicates to the input facts. The figure illustrates this with arrows flowing from the constants of the input facts to S_{out} and O_{out} forming a consequent fact at the top. The creation of separate facts for each predicate in P is required because of our ignorance of the correct predicate for the head of the rule.

The valuation for each candidate output fact is determined through soft unification by (differentiably) combining the values of the input facts with measures of the degree of similarity between each input fact predicate and the corresponding rule body embedding, as well as with the similarity between the rule head embedding and the candidate fact predicate in P. The values of the input facts are multiplied to implement a soft form of AND. [7] Specifically, we compute the value v_{out} of an

[7] There are many other choices which are more theoretically well-grounded, such as the t-norm, but we found that simple multiplication works the best in practice for this application. Note that because we restrict ourselves to at most two atoms in the body of a rule, there is no issue with underflow here.

output fact $(\theta_p, s_{out}, o_{out}, v_{out})$ resulting from the unification between a rule with head and body predicates $\theta_{\mathbf{h}}$, $\theta_{\mathbf{b_1}}, \theta_{\mathbf{b_2}}$ and facts f_1 and f_2 with values and corresponding predicates v_{f_1}, v_{f_2}, $\theta_{\mathbf{f_1}}$ and $\theta_{\mathbf{f_2}}$ as:

$$v_{out} = \cos(\theta_h, \theta_p) \cdot \cos(\theta_{\mathbf{b_1}}, \theta_{\mathbf{f_1}}) \cdot \cos(\theta_{\mathbf{b_2}}, \theta_{\mathbf{f_2}}) \cdot v_{f_1} \cdot v_{f_2} \tag{2.2}$$

If the predicate and arguments of a consequent fact matches one in the set of known facts, its value is updated with the max between its previous and newly inferred values (implementing an OR); if it is not, the new fact is appended to the set of known facts. In this way the valuation is dynamically extended at each step of inference.

We have described how the network is constructed to perform one step of inference with a single rule. To construct the entire network, we start with an initial set of facts and perform this inference procedure for each rule and for each pair of facts. If a pair of facts fails to unify with the rule then that branch of the construction terminates. The number of steps of inference K is a hyper-parameter as is the number of auxiliary predicates included in the predicate set P.

To train the network we use a loss function that depends on the task (we describe the setup for each task below). The loss gradients are back-propagated to update the predicate embeddings for the rules and for the facts (the predicates of the facts can also be fixed, i.e as one-hot vectors). The orange arrows of Figure 2-7 indicate the paths on which gradients flow (in the opposite direction). The rule and fact predicate embeddings are the parameters of the network, shown in green.

When a set of background facts is given, as in the case of the ILP tasks, we initialize the current valuation for the known facts to 1.0 and train the rules and predicates using a binary-cross-entropy loss to produce the correct values for the positive and negative target facts.

For theory learning, the aim is to learn a small theory which can recover the observations and generalize using the logical rules. Thus, we additionally learn a set of initial core facts that underlie the structure of the observations. In order to do that, unlike in the ILP setting, we additionally parameterize the valuations for all the

facts and initialize them to 0.5 reflecting our initial ignorance of their truth. We train them towards values which allow the model to faithfully recover the observations. The model may produce additional facts not in the observations but the loss penalizes only the implied facts whose predicates and arguments match those of an observed fact but whose values differ. A regularization term controlled by λ penalizes the squared sum of the initial core valuations, encouraging compression. Using the notation $i \sim f$ for facts f and i to indicate that their predicates and arguments match exactly, and $v(f)$ to denote the value of a fact f, the loss can be written:

$$\sum_{i \in I, f \in F, i \sim f} BCE(v(f), v(i)) + \lambda \sum_{i \in I} v(i) \tag{2.3}$$

where BCE is the standard Binary Cross Entropy.

2.2.4 Experiments

A wide range of previous work has focused on different aspects of logical induction. Here we test the capabilities of our algorithm in three different settings. First we test the capability of the algorithm to perform logical rule induction in the set of tasks covered by [36]. Second, we test our algorithm in a bigger dataset and compare with [95]. Finally, we test our model in the context of learning domain theories [127, 121]. The algorithm has to simultaneously learn the atoms of the logical rules, the representations of the facts of the predicates, and a set of core facts, to do that the algorithm learns to do deductive and inductive inferences of the facts.

Rule Learning ILP Tasks

We tested the model in the ILP problems from [36].[8] The task of an ILP problem is to learn a target relation given a set of background knowledge facts $\mathcal{B}$ and a set of positive $\mathcal{P}$ and negative $\mathcal{N}$ examples of a target relation.

As an example, consider the task of learning the predicate $\texttt{even}(X)$. The back-

[8] We only skip the Husband and Uncle tasks which require the datasets from [124].

ground knowledge is defined using the $\texttt{zero}(X)$ and $\texttt{succ}(X, Y)$ predicates.

$$\mathcal{B} = \{\texttt{zero}(0), \texttt{succ}(0, 1), \texttt{succ}(1, 2), ..., \texttt{succ}(9, 10)\}$$

The target positive and negative predicate extensions are:

$$\mathcal{P} = \{\texttt{target}(0), \texttt{target}(2), ..., \texttt{target}(10)\}$$

$$\mathcal{N} = \{\texttt{target}(1), \texttt{target}(9)\}$$

An example solution found by the algorithm is:

$$\texttt{target}(X) \leftarrow \texttt{zero}(X)$$

$$\texttt{target}(X) \leftarrow \texttt{target}(Y), \texttt{auxpred}(Y, X)$$

$$\texttt{auxpred}(X, Y) \leftarrow \texttt{succ}(X, Z), \texttt{succ}(Z, Y)$$

Where `auxpred` acquires the meaning of `succ2` which is true of X, Y whenever $X+2 = Y$.

Table 2.3 gives a performance comparison to [36]. Since universal logical rules are perfectly generalizable, and to facilitate comparison, we use the same evaluation metric: the percentage of runs with different random weight initializations that successfully learn rules to solve the task with less than $1e-4$ mean squared error. To avoid local minima, we explored adding a decaying normal noise to the embeddings. This had a small positive effect in some of the tasks, reported results include the effect of the noise. Details of the problems are given in Appendix A.

We see that our algorithm performs equally or better in most of the tasks. In contrast to theirs, search is not made at the level of rules but at the more compositional level of atoms. In fact, when the embeddings of the dictionary of predicates are fixed as one-hot vectors, our procedure is very similar to theirs: the parameters that form the predicates of the rules can be treated as weights that select the correct predicate. Thus, like in their model, training consists on selecting the weights that make the

Table 2.3: ILP percentage of successful runs. $|I|$ is the number of intentional predicates.

Task	$\|I\|$	Recursive	∂ILP	Ours
Predecessor	1	No	100	100
Even-Odd	2	Yes	100	100
Even-succ2	2	Yes	48.5	100
Less than	1	Yes	100	100
Fizz	3	Yes	10	10
Buzz	2	Yes	35	70
Member	1	Yes	100	100
Length	2	Yes	92.5	100
Son	2	No	100	100
Grandparent	2	No	96.5	100
Relatedeness	1	No	100	100
Father	1	No	100	100
Undirected Edge	1	No	100	100
Adjacent to Red	2	No	50.5	100
Two Children	2	No	95	0
Graph Colouring	2	Yes	94.5	0
Connectedness	1	Yes	100	100
Cyclic	2	Yes	100	100

embeddings look like the right one-hot vectors and the procedure becomes a symbol search, except at the level of the atoms instead of the rules. The more general case where the embeddings are trained and dense, opens the interesting direction to be explored of studying the learned vector embedding semantic space, as has been done for standard NLP tasks [76], which can potentially allow for similarity and analogical reasoning. In our table, we report the result with the trainable embeddings.

It is worth noting the failure of our model in the Graph Colouring and in the Two Children tasks. A quick exploration suggests that the global optima has a very sharp neighborhood while the local minima are attractors in most of the space. This is reminiscent of the Terpret problem [105, 43] and the local minima can only be avoided when the random initialization is very close to the correct rules.

Notice that since our approach is differentiable, it is not prone to some of the problems of symbolic ILP, and like in [36], it can handle ambiguity and noise.

Table 2.4: Performance on the COUNTRIES dataset

Task	Model AUC-PR			Rule examples and confidences
	NTP	NTP-λ	Ours	
S1	90.83 ± 15.4	100.00 ± 0.0	91.15 ± 15.4	0.85 loc(X,Y) $\leftarrow$ loc(X,Z), loc(Z,Y)
S2	87.40 ± 11.7	94.04 ± 0.4	86.87 ± 3.2	0.57 loc(X,Y) $\leftarrow$ neighbor(X,Z), loc(Z,Y)
S3	56.68 ± 17.6	77.26 ± 17.0	63.08 ± 28.2	0.59 loc(X,Y) $\leftarrow$ neighbor(X,Z), loc(Z,W), loc(W,X)

Countries We are not focused specifically on knowledge base completion but use the COUNTRIES dataset [14] to evaluate the scalability of our algorithm, comparing to other neural logical approaches. The dataset contains 272 constants, 2 predicates and 1158 true facts and is designed to explicitly test the logical rule induction and reasoning capabilities of link prediction models.

We compare on the 3 tasks described in [95], requiring reasoning steps of increasing length and difficulty (S1,S2,S3 in table 2.4). We report the Area Under the Precision-Recall-curve (AUC-PR) where results are comparable to the previous NTP approach. For completion, we also report $NTP-\lambda$ from the same paper which uses an additional neural link network as an auxiliary loss. Like them, we also show some example rules and a confidence score by taking the minimum similarity between the atoms of the rule and their decoded predicates.

To perform the forward chaining during training, at each epoch we randomly sample from a section of the knowledge graph both the targets and a set of facts to form the background knowledge. Like the related work, our model can also suffer from scalability issues, as in forward chaining the size of the facts grows exponentially with the number of steps. Restricting the number of considered facts through sampling was sufficient for the task at consideration but this could show problems when scaling to much bigger datasets, we discuss some future directions in the conclusion.

Table 2.5: Theory Learning Results. Succ is the percentage of successful initializations; Acc stands for the accuracy of the recovered facts; Const is the number of constants.

	Taxonomy			Family		
	# Preds	# Const	# Facts	# Preds	# Const	# Facts
Observed Data	4	36	145	6	10	30
Target Theory	4	36	40	4	10	28
	% Succ	% Acc	# Induced Facts	% Succ	% Acc	# Induced Facts
Algorithm	70	99	69	100	96	30.8

Learning Theories

We test the capability of our network to compress a set of observations in the form of a theory by learning a set of core facts in addition to the logical rules. We take the two domains considered by Katz et al.[127]: Taxonomy and Kinship.

Taxonomy A taxonomy is a set of observations structured into a tree where downstream nodes inherit the properties from the nodes above them in the tree. As shown in Figure 1, the tree constitutes a theory formed by two logical rules $\texttt{IS}(X, Y) \leftarrow \texttt{IS}(X, Z), \texttt{IS}(Z, Y)$; $\texttt{HAS}(X, Y) \leftarrow \texttt{IS}(X, Z), \texttt{HAS}(Z, Y)$ capturing inheritance and by a set of core facts represented with the edges. All observed facts can be recovered by iterative application of the rules (if salmon are fish, and fish have gills, then salmon have gills).

From a range of different possible taxonomies we report performance on the bigger original one from [96]. This data contains 145 facts composed of 4 predicates and 36 constants. The facts can be compressed into a tree structured theory as shown in the Appendix that contains only 40 core facts.

As shown in Table 2.5, the algorithm is able to learn the theory in 70% of the runs, achieving 99% accuracy and compressing close to the optimal level (average of 69 compared to the optimal of 40).

Kinship We also evaluated performance on the difficult kinship theory, which contains 10 constants and 6 observed predicates `mother`, `father`, `daughter`, `wife`, `husband` (see figure in the Appendix). In this case the compression of the theory consists of a set of 4 auxiliary core predicates with 28 facts. The algorithm has to learn the concepts `female`, `male`, `spouse`, `child` which acquire their meaning through their extensions and the 6 logical rules that generate the observations:

$$\texttt{mother}(X,Y) \leftarrow \texttt{female}(X), \texttt{child}(Y,X)$$

$$\texttt{father}(X,Y) \leftarrow \texttt{male}(X), \texttt{child}(Y,X)$$

$$\texttt{daughter}(X,Y) \leftarrow \texttt{female}(X), \texttt{child}(X,Y)$$

$$\texttt{son}(X,Y) \leftarrow \texttt{male}(X), \texttt{child}(X,Y)$$

$$\texttt{wife}(X,Y) \leftarrow \texttt{female}(X), \texttt{child}(X,Y)$$

$$\texttt{husband}(X,Y) \leftarrow \texttt{male}(X), \texttt{child}(X,Y)$$

Table 2.5 shows the statistics for the observed and target compressed data. The algorithm's performance is again quantified as the percentage of initializations where the rules are successfully learned, the accuracy of the recovered data and the number of learned core facts. The algorithm is able to perform this compression and learns a set of new predicates that conform the rules, recovering 96% of the data correctly (the algorithm sometimes deduces that some facts are true when they aren't because it can induce incorrect core facts).

2.2.5 Conclusions and Future Work

We have presented a forward chaining inference network which is parameterized in the embeddings that form its rules and facts. Learning in this model means simultaneously learning the right sub-symbolic representations, and the right resulting symbolic conceptual relations implied through the logical rules; together constituting a Dual-Factor definition of the concepts [23].

We articulated a set of desiderata for models which learn logical theories from observations which include: compression, rule induction, and the ability to learn without direct supervision. We showed how our inference procedure satisfies these three conditions in two settings. Our model was able to learn a significant compression for the taxonomy and kinship datasets proposed by [96] and [127]; learning interpretable representations not just for the parametrized rules but also for facts – a feature not emphasized in many traditional ILP solutions – using only the supplied observations as supervision. These are encouraging results for theory acquisition and point to the viability of this approach. As demonstrated on the ILP datasets from [14]; [28] and those from [95], the method also provides an interesting alternative in the ILP setting.

Limitations As in previous work, our model is provided with rules that conform to templates, ideally this should not be necessary. The network needs to consider the set of all possible facts when doing core fact induction (which is not necessary for the rule induction problems), this is not be scalable in practice. Forward chaining grows exponentially in the number of facts considered at each step, this can also present a problem when scaling to bigger datasets. From a cognitive science perspective, the model is still more limited than its Bayesian symbolic counterparts, specifically, while those models provide graded measures of confidence in their inferences, our neural logical reasoner does not currently provide meaningful consistent estimates of uncertainty.

Future Directions One straightforward attempt of learning the structural information of the rules provided by the templates (arity and variable order) would be to encode it by adding dimensions to the embeddings and have the algorithm interpret them by using independent unifications in the desired way. This would constitute a slightly more complicated learning task but would maintain the same structure and mechanism that could be trained through gradient descent. It would also be interesting to explore richer sampling procedures and the integration of forward with backward chaining, this could perhaps yield regimes more similar to those of humans

and could help scale to larger datasets. We would also like to investigate ways of providing better estimates of uncertainty – from a full neural probabilistic formulation, to a heuristic metric based on the number of initializations and on the unification scores.

2.3 AMIGo: Adversarially Motivated Intrinsic Goals

2.3.1 Introduction

The success of Deep Reinforcement Learning (RL) on a wide range of tasks, while impressive, has so far been mostly confined to scenarios with reasonably dense rewards [77, 123], or to those where a perfect model of the environment can be used for search, such as the game of Go and others [109, 32, 78]. Many real-world environments offer extremely sparse rewards, if any at all. In such environments, random exploration, which underpins many current RL approaches, is likely to not yield sufficient reward signal to train an agent, or be very sample inefficient as it requires the agent to stumble onto novel rewarding states by chance. In contrast, humans are capable of dealing with rewards that are sparse and lie far in the future. For example, to a child, the future adult life involving education, work, or marriage provides no useful reinforcement signal. Instead, children devote much of their time to play, generating objectives and posing challenges to themselves as a form of intrinsic motivation. Solving such self-proposed tasks encourages them to explore, experiment, and invent; sometimes, as in many games and fantasies, without any direct link to reality or to any source of extrinsic reward. This kind of intrinsic motivation might be a crucial feature to enable learning in real-world environments [104] .

To address this discrepancy between naïve deep RL exploration strategies and human capabilities, we present a novel meta-learning method wherein part of the agent learns to self-propose **A**dversarially **M**otivated **I**ntrinsic **Go**als (AMIGo). In AMIGo, the agent is decomposed into a goal-generating teacher and a goal-conditioned student policy. The teacher acts as a constructive adversary to the student: the teacher is incentivized to propose goals that are not too easy for the student to achieve, but not impossible either. This results in a natural curriculum of increasingly harder intrinsic goals that challenge the agent and encourage learning about the dynamics of a given environment.

AMIGo can be viewed as an *augmentation* of any agent trained with policy gradient-based methods. Under this view, the original policy network becomes the

student policy, which only requires its input-processing component to be adapted to accept an additional goal specification modality. The teacher policy can then be seen as a "bolt-on" to the original policy network, entailing that this method is—to the extent that the aforementioned goal-conditioning augmentation is possible—architecture-agnostic, and can be used on a variety of RL training model architectures and training settings.

As advocated in recent work [25, 133, 94, 61], we evaluate AMIGo for procedurally-generated environments instead of trying to learn to perform a specific task. Procedurally-generated environments are challenging since agents have to deal with a parameterized family of tasks, resulting in large observation spaces where memorizing trajectories is infeasible. Instead, agents have to learn policies that generalize across different environment layouts and transition dynamics [92, 67, 40, 130]. Concretely, we use MiniGrid [24], a suite of fast-to-run procedurally-generated environments with a symbolic/discrete (expressed in terms of objects like walls, doors, keys, chests and balls) observation space which isolates the problem of exploration from that of visual perception. MiniGrid is a widely recognized challenging benchmark for intrinsic motivation, which was used in many recent publications such as Goyal *et al.* 2019, Bougie *et al.* 2019, Raileanu and Rocktäschel 2020, Modhe *et al.* 2020 etc. We evaluate our method on six different tasks from the MiniGrid domain with varying degrees of difficulties, in which the agent needs to acquire a diverse range of skills in order to succeed. Furthermore, MiniGrid is complex and competitive baselines such as IMPALA (that achieve SOTA in other domains like Atari) fail. [91] found that MiniGrid presents a particular challenge for existing state-of-the-art intrinsic motivation approaches. Here, AMIGo sets a new state-of-the-art on some of the hardest MiniGrid environments, being the only method capable of successfully obtaining extrinsic reward on some of them.

In summary, we make the following contributions: (i) we propose Adversarially Motivated Intrinsic Goals—an approach for learning from a teacher that generates increasingly harder goals, (ii) we show, through 114 experiments on 6 challenging exploration tasks in procedurally generated environments, that agents trained

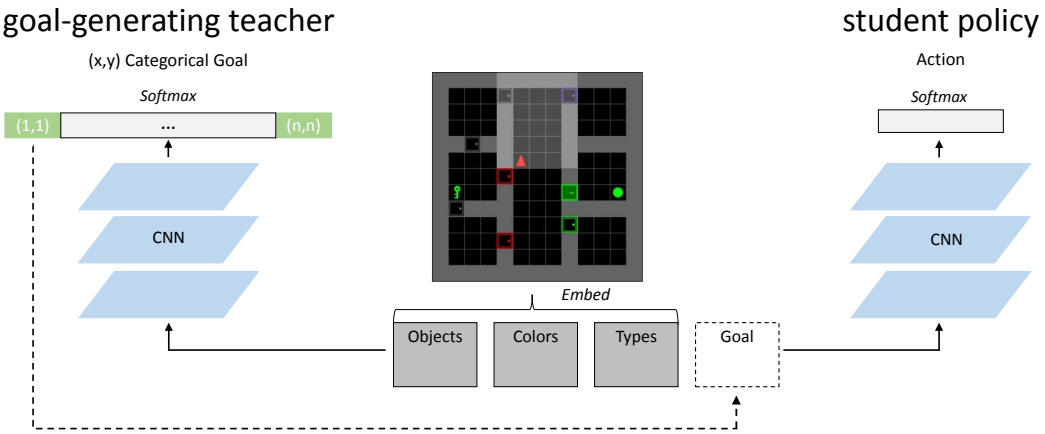


Figure 2-8: Training with AMIGo consists of combining two modules: a goal-generating teacher and a goal-conditioned student policy, whereby the teacher provides intrinsic goals to supplement the extrinsic goals from the environment. In our experimental set-up, the teacher is a dimensionality-preserving convolutional network which, at the beginning of an episode, outputs a location in absolute (x, y) coordinates. These are provided as a one-hot indicator in an extra channel of the student's convolutional neural network, which in turn outputs the agent's actions.

with AMIGo gradually learn to interact with the environment and solve tasks which are too difficult for state-of-the-art methods, and (iii) we perform an extensive qualitative analysis and ablation study.

2.3.2 Related Work

Our work has connections to many different research areas but due to space constraints, we will focus our discussion on the most closely related topics, namely intrinsic motivation and curriculum learning.

Intrinsic motivation [83, 84, 101, 4] methods have proven effective for solving various hard-exploration tasks [8, 85, 19]. One prominent formulation is the use of *novelty*, which in its simplest form can be estimated with state visitation counts [116] and has been extended to high-dimensional state spaces [8, 19, 82]. Other sophisticated versions of *curiosity* [101] guide the agent to learn about environment dynamics by encouraging it to take actions that reduce the agent's uncertainty [115, 19], have unpredictable consequences [85, 18], or a large impact on the environment [91]. Other forms of intrinsic motivation include *empowerment* [56] which encourages control of

the environment by the agent, and *goal diversity* [86] which encourages maximizing the entropy of the goal distribution. In [62], intrinsic goals are discovered from language supervision. The optimal rewards framework presents intrinsic motivation as a mechanism that goes beyond exploration, placing its origin in an evolutionary context [111], or framing it as a meta-optimization problem of selecting internal agent goals which optimize the designer's goals [113]. More recently [132] extend this framework to learn parametric additive intrinsic rewards. Our work differs from all of the above by formulating intrinsic motivation as a "constructively adversarial" teacher that proposes increasingly harder goals for the agent.

Curriculum learning [9] is another useful technique for tackling complex tasks but the curricula are typically handcrafted which can be time consuming. In our work, the curriculum is generated automatically in an unsupervised fashion. Another automatic curriculum approach learning was proposed by [102], where an agent constantly searches the space of problems for the next solvable one. However, this method is not scalable to more complex tasks. [39] generate a curriculum by increasing the distance of the starting-point to a goal. In contrast to AMIGo, this method assumes knowledge of the goal location and the ability to reset the agent in any state. A student can also self-propose a goal by hindsight experience replay (HER) [3], which has been demonstrated to be effective in alleviating the sparse reward problem. Recent extensions have improved goal selection by balancing the difficulty and diversity [37] of goals. In contrast to our work, in HER, there is no explicit incentive for the agent to explore beyond its current reach. Since HER is rewarded for all the states it visits, it is rewarded for easy-to-reach states, even late in the training process. Other related work has trained a teacher to generate a curriculum of environments that maximize the learning process of the student [87]. The question of the complementarity between these and our work is worth pursuing in the future. More similar to our work, [75] train a teacher to select tasks in which the student is improving the most or in which the student's performance is decreasing to avoid forgetting. Note that AMIGo uses a different objective for training the teacher, which encourages the agent to solve progressively harder tasks. Similarly, [90] train a goal-conditioned policy and a goal-

setter network in a non-adversarial way to propose feasible, valid and diverse goals. Their feasibility criteria is similar to ours, but requires training an additional discriminator to rank the difficulty of the goals, while our teacher is directly trained to generate goals with an appropriate level of difficulty.[9] Recent surveys of curriculum generation in the context of RL include [79] and [88].

Closer to our work, [117] use an adversarial framework but require two modules that independently act and learn in the environment, where one module is encouraged to propose challenges to the other. This setup can be costly and is restricted to only proposing goals which have already been reached by the policy. Moreover, it requires a resettable or reversible environment. In contrast, our method uses a single agent acting in the environment, and the teacher is less constrained in the space of goals it can propose.

Also similar to ours, [38] present GoalGAN, a generator that proposes goals with the appropriate level of difficulty as determined by a learned discriminator. While their work is similar in spirit with ours, there are several key differences. First, GoalGAN was created for and tested on locomotion tasks with continuous goals, whereas our method is designed for discrete action and goal spaces. While not impossible, adapting it to our setting is not trivial due to the GAN objective. Second, the authors do not condition the generator on the observation which is necessary in procedurally-generated environments that change with each episode. GoalGAN generates goals from a buffer, but previous goals can be unfeasible or nonsensical for the current episode. Hence, GoalGAN cannot be easily adapted to procedurally-generated environments.

A concurrent effort, [131] complements ours, but in the context of continuous control, by also generating a curriculum of goals which are neither too hard nor too easy using a measure of epistemic uncertainty based on an ensemble of value functions. This requires training multiple networks, which can become too computationally ex-

[9]Unfortunately, both the code for their method—which is far from simple—and for their experimental settings have not been made available by the authors. Therefore we not only cannot run a fair implementation of their approach against our setting for comparison, we cannot be guaranteed to successfully reimplement it ourselves as there is no way of reproducing their results in their setting without the code for the latter.

pensive for certain applications.

Finally, our approach is loosely inspired by generative adversarial networks(GANs) [44], where a generative model is trained to fool a discriminator which is trained to differentiate between the generated and the original examples. In contrast with GANs, AMIGO does not require a discriminator, and is "constructively adversarial", in that the goal-generating teacher is incentivized by its objective to propose goals which are challenging yet feasible for the student.

2.3.3 Adversarially Motivated Intrinsic Goals

AMIGO is composed of two subsystems: a goal-conditioned student policy which "controls" the agent's actions in the environment, and a goal-generating teacher (see Figure 2-8) which guides the student's training. The teacher proposes goals and is rewarded only when the student reaches the goal after a certain number of steps. The student receives reward for reaching the goal proposed by the teacher (discounted by the number of steps needed to reach the goal). The two components are trained adversarially in that the student maximizes reward by reaching goals as fast as possible, while the teacher maximizes reward by proposing goals which the student can reach, though not too quickly. In addition to this intrinsic reward, both modules are rewarded when the agent solves the full task.

Training the student

We consider the traditional RL framework of a Markov Decision Process with a state space S, a set of actions A and a transition function $p(s_{t+1}|s_t, a_t)$ which specifies the distribution over next states given a current state and action. At each time-step t, the agent in state $s_t \in S$ takes an action $a_t \in A$ by sampling from a goal-conditioned stochastic student policy $\pi(a_t|s_t, g; \theta_\pi)$ where g is a intrinsic goal provided by the teacher.

The teacher $G(s_0; \theta_g)$ is a separate policy, operating on a different "granularity" than the student: it takes as input an initial state and outputs as actions a goal g for

the student, which stays the same until a new goal is proposed. The teacher proposes a new goal every time an episode begins or whenever the student reaches the intrinsic goal. We assume that some goal verification function $v(s, g)$ can be specified as an indicator over whether a goal g is achieved in a state s. We use this to define the undiscounted intrinsic reward r_t^g as:

$$r_t^g = v(s_t, g) = \begin{cases} +1 & \text{if the state } s_t \text{ satisfies the goal } g \\ 0 & \text{otherwise} \end{cases}$$

At each time step t, the student receives a reward $r_t = r_t^g + r_t^e$, which is the sum of the intrinsic reward r_t^g provided by the teacher and the extrinsic reward r_t^e provided by the environment. The student, represented as a neural network with parameters θ_π, is trained to maximize the discounted expected reward $R_t = \mathbb{E}\big[\sum_{k=0}^{H} \gamma^k r_{t+k}\big]$ where $\gamma \in [0, 1)$ is the discount factor. We consider a finite time horizon H as provided by the environment.

Training the Teacher

The teacher $G(s_0; \theta_g)$, represented as a neural network with parameters θ_g, is trained to maximise its expected reward. The teacher's reward r^T is a function of the student's performance on the proposed goal and is computed every time this goal is reached (or at the end of an episode). As a result, the teacher operates at a different temporal frequency, and thus its rewards are not discounted according to the number of steps taken by the agent. To generate an automatic curriculum for the student, we positively reward the teacher if the student achieves the goal with suitable effort, but penalize it if the student either cannot achieve the goal, or can do so too easily. There are different options for measuring the performance of the student here, but for simplicity we will use the number of steps t^+ it takes the student to reach an intrinsic goal since the intrinsic goal was set (with $t^+ = 0$ if the student does not reach the goal before the episode ends). We define a threshold t^* such that the teacher is positively rewarded by r^T when the student takes more steps than the threshold to

reach the set goal, and negatively if it takes fewer steps or never reaches the goal before the episode ends. We thus define the teacher reward as follows, where α and β are hyperparameters (see Section 2.3.4 for implementation details) specifying the weight of positive and negative teacher reward:

$$r^T = \begin{cases} +\alpha & \text{if} \quad t^+ \geq t^* \\ -\beta & \text{if} \quad t^+ < t^* \end{cases}$$

One can try to calibrate a fixed target threshold t^* to force the teacher to propose increasingly more challenging goals as the student improves. Initial experiments with a fixed threshold indicated that the loss function was sufficient to induce harder goals and curriculum learning. However, this threshold is different across environments and has to be carefully fixed (depending on the size and complexity of the environment). A more adaptive—albeit heuristic—approach we adopt is to linearly increase the threshold t^* after a fixed number of times in which the student successfully reaches the intrinsic goals. Specifically, the threshold t^* is increased by 1 whenever the student successfully reaches an intrinsic goal in more than t^* steps for ten times in a row. This increase in the target threshold provides an additional metric to visualize the improvement of the student through the "difficulty" of its goals (see Figure 2-10).

Types of Goals

We can conceive of variants of AMIGo whereby goals are provided in the form of linguistic instructions, images, etc. To prove the concept, in this framework, a goal is formally defined as a change in the observation on a tile, as specified by an (x, y) coordinate. The agent must modify the tile before the end of an episode (e.g. by moving to it, or causing the object in it to move or change state). The verification function v is then trivially the indicator function of whether the cell state is different from its initial state at the beginning of the episode. Proposing (x, y) coordinates as goals can present a diverse set of ways for an agent to achieve the goal, as the coordinates can not only be affected by reaching them but also by modifying what

is on them. This includes picking up keys, opening doors, and dropping objects onto empty tiles. In some cases in our setting, moving over a square is not the simplest thing possible (e.g. when an obstacle can be removed or a door can be opened). Similarly, in other tasks and environments it could be easier to affect a cell by throwing something at it, rather than by reaching it. Likewise, simply navigating to a set of coordinates (say, the corner of a locked room) might require solving several non-trivial sub-problems (e.g. identifying the right key, going to it, then going to the door, unlocking it, and finally going to the target location). We give some examples of goals proposed by the teacher, alongside the progression in their difficulty as the student improves, in Figure 2-10.

Auxiliary Teacher Losses

To complement our main form of intrinsic reward, we explore a few other criteria, including goal diversity, extrinsic reward, environment change and novelty. We report, in our experiments of Section 2.3.4, the results for AMIGo using these auxiliary losses. We present, in Appendix B.4, an ablation study of the effect of these losses, alongside some alternatives to the reward structure for the teacher network.

Diverse Goals. One desirable property is goal diversity [86, 91]. In our implementation of AMIGo in the experiments of Section 2.3.4, we used entropy regularization to train the teacher and student, which encourages such diversity. This regularization, along with the scheduling of the threshold, helps the teacher avoid getting stuck in local minima. Additionally, we considered rewarding the teacher for proposing novel goals similar to count-based exploration methods [8, 82] with the difference that in our case the counts are for goals instead of states, based on the number of times the teacher presents a type of goal to the student. This did not improve performance and is not part of our model for the rest of the paper.

Episode Boundary Awareness. When playing in a procedurally-generated environment, humans will notice the factors of variation and exploit them. In episodic training, RL agents and algorithms are informed if a particular state was an episode end. To bias AMIGo towards learning the factors of variation in an environment,

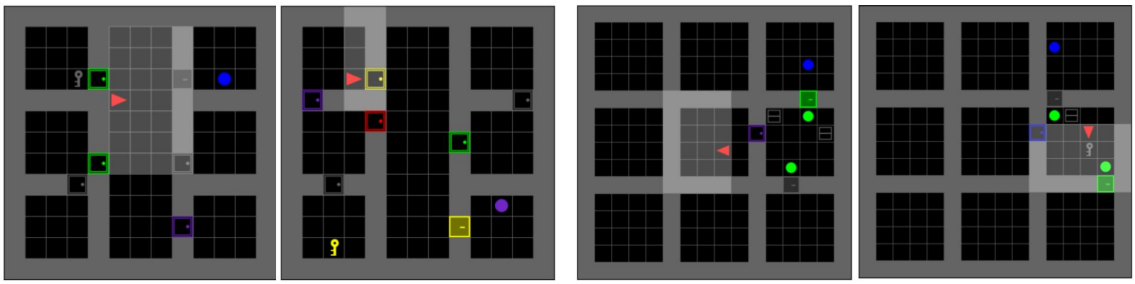
while not giving it any domain knowledge or any privileged information which other comparable intrinsic motivation systems and RL agents would not have access to, we positively reward the teacher if the content of the goal location it proposes changes at an episode boundary, regardless of whether this change was due to the agent. Thus, the teacher is rewarded for selecting goals where the object type changes if the episode changes (for example a door becomes a wall, or a key becomes an empty tile due to the new episode configuration). While this heuristic is quite general and could be effective for many tasks as it encourages agents to note environmental factors of variation, we note it might not be useful in all possible domains and as such is not an essential part of AMIGo. A comparison an extension of this loss other intrinsic motivation methods would interesting, but is not straightforward and is left for future research, we just note that this auxiliary loss on its own is not able to solve even the medium difficulty environments.

Extrinsic Goals. To help transition into the extrinsic task and avoid local minima, we reward both the teacher and the student with environment reward whenever the student reaches the extrinsic goal, even if this does not coincide with the intrinsic goal set by the teacher. This avoids the degenerate case where the student becomes good at satisfying the extrinsic goal, and the teacher is forced to encourage it "away" from it.

2.3.4 Experiments

We follow [91] and evaluate our models on several challenging procedurally-generated environments from MiniGrid [24]. This environment provides a good testbed for exploration in RL since the observations are symbolic rather than high-dimensional, which helps to disentangle the problem of exploration from that of visual understanding. We compare AMIGo with state-of-the-art methods that use various forms of exploration bonuses. We use TorchBeast [60], a PyTorch platform for RL research based on IMPALA [35] for fast, asynchronous parallel training. The code for these experiments is included in the supplementary materials, and has also been released under "https://anonymous" to facilitate reproduction of our method and its use in

other projects.



(a) Example episodes in **KCharder**. (b) Example episodes in **OMhard**.

Figure 2-9: Examples of MiniGrid environments. **KCharder** requires finding the key that can unlock a door which blocks the room where the goal is (the blue ball). **OMhard** requires a sequence of correct steps usually involving opening a door, opening a chest to find a key of the correct color, picking-up the key to open the door, and opening the door to reach the goal. The configuration and colors of the objects change from one episode to another. To our knowledge, AMIGo is the only algorithm that can solve these tasks. For other examples, see the MiniGrid repository.

Environments

We evaluate AMIGo on the following MiniGrid environments: KeyCorrS3R3 (**KCmedium**), ObstrMaze1Dl (**OMmedium**), ObstrMaze2Dlhb (**OMmedhard**), KeyCorrS4R3 (**KChard**), KeyCorrS5R3 (**KCharder**), and ObstrMaze1Q (**OMhard**). The agent receives a full observation of the MiniGrid environment. The layout of the environment changes at every episode as it is procedurally-generated. Examples of these tasks can be found in Figure 2-9. Each environment is a grid of size $N \times N$ (N being environment-specific) where each tile contains at most one of the following colored objects: wall, door, key, ball, chest. An object in each episode is selected as an extrinsic goal. If the agent reaches the extrinsic goal, or a maximum number of time-steps is reached, the environment is reset. The agent can take the following actions: turn left, turn right, move forward, pick up an object, drop an object, or toggle (open doors or interact with objects). Each tile is encoded using three integer values: the object, the color, and a type or flag indicating whether doors are open or closed. While policies could be learned from pixel observation alone, we will see below that the exploration problem is sufficiently complex with these semantic layers, owing to the procedurally generated

nature of the tasks. The observations are transformed before being fed to agents by embedding each tile of the observed frame into a single representation encoding the object type, color, and type/flag.

The extrinsic reward provided by each environment for reaching the extrinsic goal in t steps is $r_t^e = 1 - (.9 \cdot t)/t^{\max}$, where $t^{\max}$ is the maximum episode length (which is intrinsic to each environment and set by the MiniGrid designers), if the extrinsic goal is reached at t, and 0 otherwise. Episodes end when the goal is reached, and thus the scale of the positive reward encourages agents to reach the goal as quickly as possible.

AMIGo Implementation

The teacher is a dimensionality-preserving network of four convolutional layers interleaved with exponential linear units. Similarly, the student consists of four convolutional layers interleaved with exponential linear units followed by two linear layers with rectified linear units. Both the student and the teacher are trained using the TorchBeast [60] implementation of IMPALA [35], a distributed actor-critic algorithm. But while the teacher proposes goals only at the beginning of an episode or when the student reaches a goal, the student produces an action and gets a reward at every step. To replicate the structure of reward for reaching extrinsic goals, intrinsic reward for the student is discounted to $r_t^g = 1 - (.9 \cdot t)/t^{\max}$ when $v(s_t, g) = 1$, and 0 otherwise. The hyperparameters for the reward for the teacher r^T are grid searched, and optimal values are found at $\alpha = .7$ and $\beta = .3$ (see Appendix B.2 for full hyperparameter search details).

Baselines and Evaluation

We use **IMPALA** [35] without intrinsic motivation as a standard deep RL baseline. We then compare AMIGo to a series of methods that use intrinsic motivation to supplement extrinsic reward, as listed here. **Count** is Count-Based Exploration from [8], which computes state visitation counts and gives higher rewards to less visited states. **RND** is Random Network Distillation Exploration by [19] which uses a random net-

work to compute a prediction error used as a bonus to reward novel states; **ICM** is Intrinsic Curiosity Module from [85], which trains forward and inverse models to learn a latent representation used to compare the predicted and actual next states. The Euclidean distance between the representations of predicted and actual states (as measured in the latent space) is used as intrinsic reward. **RIDE**, from [91], defines the intrinsic reward as the (magnitude of the) change between two consecutive state representations.

We have noted from the literature that some of these baselines were designed for partially observable environments [91, 85] so they might benefit from observing an agent-centric partial view of the environment rather than a full absolute view [128]. Despite our environment being fully observable, for the strongest comparison with AMIGO we ran the baselines in each of the following four modes: full observation of the environment for both the intrinsic reward module and the policy network, full observation for the intrinsic reward and partial observation for the policy, partial view for the intrinsic reward and full view for the policy, and partial view for both. We use an LSTM for the student policy network when it is provided with partial observations and a feed-forward network when provided with full observations. In Section 2.3.4, we report the best result (across all four modes) for each baseline and environment pair, with a full breakdown of the results in Appendix B.1. This, alongside a comprehensive hyperparameter search, ensures that AMIGO is compared against the baselines trained under their individually best-performing training arrangement. We also compare AMIGO to the authors' implementation[10] of Asymmetric Self-Play (**ASP**) [117]. In their reversible mode two policies are trained adversarially: Alice starts from a start-point and tries to reach goals, while Bob is tasked to travel in reverse from the goal to the start-point.

We ran each experiment with five different seeds, and report in Section 2.3.4 the means and standard deviations. The full hyperparameter sweep for AMIGO and all baselines is reported in Appendix B.2, alongside best hyperparameters across experiments.

[10] `https://github.com/tesatory/hsp`

Results and Discussion

We summarize the main results of our experiments in Table 2.6. As discussed in Section 2.3.4, the reported result for each baseline and each environment is that of the best performing configuration for the policy and intrinsic motivation system for that environment, as reported in Tables B.1–B.4 of Appendix B.1. This aggregation of 114 experiments (not counting the number of times experiments were run for different seeds) ensures that each baseline is given the opportunity to perform in its best setting, in order to fairly benchmark the performance of AMIGo.

Table 2.6: Comparison of Mean Extrinsic Reward at the end of training (averaging over a batch of episodes as in IMPALA). Each entry shows the result of the best observation configuration, for each baseline, from Tables B.1–B.4 of Appendix B.1.

	Medium Difficulty Environments			**Hard Environments**		
Model	**KCmedium**	**OMmedium**	**OMmedhard**	**KChard**	**KCharder**	**OMhard**
AMIGo	**.93** ± .00	.92 ± .00	.83 ± .05	**.54** ± .45	**.44** ± .44	**.17** ± .34
IMPALA	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00
RND	.89 ± .00	**.94** ± .00	**.88** ± .03	.23 ± .40	.00 ± .00	.00 ± .00
RIDE	.90 ± .00	**.94** ± .00	.86 ± .06	.19 ± .37	.00 ± .00	.00 ± .00
Count	.90 ± .00	.04 ± .04	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00
ICM	.42 ± .21	.19 ± .19	.16 ± .32	.00 ± .00	.00 ± .00	.00 ± .00
ASP	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00

IMPALA and Asymmetric Self-Play are unable to pass any of these medium or hard environments. ICM and Count struggle on the "easier" medium environments, and fail to obtain any reward from the hard ones. Only RND and RIDE perform competitively on the medium environments, but struggle to obtain any reward on the harder environments.

Our results demonstrate that AMIGo establishes a new state of the art in harder exploration problems in MiniGrid. On environments with medium difficulty such as **KCmedium**, **OMmedium**, and **OMmedhard**, AMIGo performs comparably to other state-of-the-art intrinsic motivation methods. AMIGo is often able to successfully reach the extrinsic goal even on the hardest tasks. To showcase results and sample complexity, we illustrate and discuss how mean extrinsic reward changes during training in Appendix B.3. To analyze which components of the teacher loss were

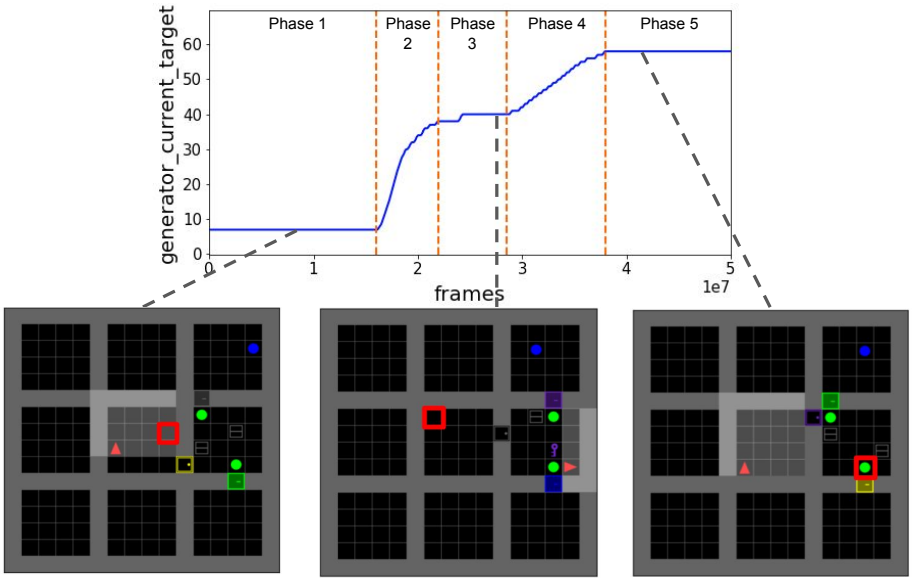


Figure 2-10: Examples of a curriculum of goals proposed for different episodes of a particular learning trajectory on **OMhard**. The red triangle is the agent, the red square is the goal proposed by the teacher, and the blue ball is the extrinsic goal. The top panel shows the threshold target difficulty, t^* of the goals proposed by the teacher. The teacher first proposes very easy nearby goals, then it learns to propose goals that involve traversing rooms and opening doors, while in the third phase the teacher proposes goals which involve removing obstacles and interacting with objects.

important, we present, in Appendix B.4, an ablation study over the components presented in Section 2.3.3. Qualitatively, the learning trajectories of AMIGo display interesting and partially adversarial dynamics. These often involve periods in which both modules cooperate as the student becomes able to reach the proposed goals, followed by others in which the student becomes too good, forcing a drop in the teacher reward, in turn forcing the teacher to increase the difficulty of the proposed goals and forcing the student to further explore. In Appendix B.5, we provide a more thorough qualitative analysis of AMIGo, wherein we describe the different phases of evolution in the difficulty of the intrinsic goals proposed by the teacher, as exemplified in Figure 2-10. Further goal examples are shown in Figure B-3 of Appendix B.6.

2.3.5 Conclusion

In this work, we propose AMIGo, a meta-learning framework for generating a natural curriculum of goals that help train an agent as a form of intrinsic reward, to supplement extrinsic reward (or replace it if it is not available). This is achieved by having a goal generator as a teacher that acts as a constructive adversary, and a policy that acts as a student conditioning on those goals to maximize an intrinsic reward. The teacher is rewarded to propose goals that are challenging but not impossible. We demonstrate that AMIGo surpasses state-of-the-art intrinsic motivation methods in challenging procedurally-generated tasks in a comprehensive comparison against multiple competitive baselines, in a series of 114 experiments across 6 tasks. Crucially, it is the only intrinsic motivation method which allows agents to obtain any reward on some of the harder tasks, where non-intrinsic RL also fails.

The key contribution of this paper is a model-agnostic framework for improving the sample complexity and efficacy of RL algorithms in solving the exploration problems they face. In our experiments, the choice of goal type imposed certain constraints on the nature of the observation, in that both the teacher and student need to fully observe the environment, due to the goals being provided as absolute coordinates. Technically, this method could also be applied to partially observed environments where part of the full observation is uncertain or occluded (e.g. "fog of war" in StarCraft), as the only requirement is that absolute coordinates can be provided and acted on. However, this is not a fundamental requirement, and in future work we would wish to investigate the cases where the teacher could provide more abstract goals, perhaps in the form of language instructions which could directly specify sequences of subgoals. Other extensions to this work worth investigating are its applicability to continuous control domains, visually rich domains, or more complex procedurally generated environments such as [25]. Until then, we are confident we have proved the concept in a meaningful way, which other researchers will already be able to easily adapt to their model and RL algorithm of choice, in their domain of choice.

Acknowledgements

This chapter was adapted from the following papers where the dissertation author was a primary author:

- Section 2.1: "Learning to Learn Visual Object Categories by Integrating Deep Learning with Hierarchical Bayes" [20] (CogSci 2017, by Andres Campero, Andrew Francl, and Joshua B. Tenenbaum)

- Section 2.2: "Logical Rule Induction and Theory Learning Using Neural Theorem Proving" [21] (2018, by Andres Campero, Aldo Pareja, Tim Klinger, Joshua B. Tenenbaum, and Sebastian Riedel)

- Section 2.3: "Learning with amigo: Adversarially motivated intrinsic goals" [22] (ICLR 2021, by Andres Campero, Roberta Raileanu, Heinrich Kuttler, Joshua B. Tenenbaum, Tim Rocktaschel and Edward Grefenstettete).

Chapter 3

AI Research as Collective Intelligence: A Taxonomy

3.1 Introduction

How is Artificial Intelligence research currently organized? How could it be organized better? How does it evolve and how can one track and analyze its progress? How does the evolution of models and tasks interact with the organization of the research community? Can we build frameworks that improve and enhance this organization?

These are the type of questions we attempt to tackle in this chapter. In order to do so we take stance at both the high-level and the implementation level [73] :

At the high level we advocate a Collective Intelligence perspective as a framework to think about the complex system of researchers, computer code, and all the incentive structures, coordination mechanisms, tools, and other interfaces which modulate the interaction between them.

At the implementation level, we build and present a taxonomy comprised of three main components. First a taxonomy of models classified into fields and sub-fields, as well as a chronology with the most important causal links describing the evolution of AI through successors and predecessors. Second, a taxonomy of tasks, sub-tasks and datasets. Third, as a dataset that could help provide a link between models and datasets, for each paper accepted at the conference NeurIPS 2020, we manually

classify its models, tasks, and datasets in a comprehensive manner. All The taxonomy is available a the Open Science Framework and is also available through an interactive viewer developped at Stateoftheart.ai, displayed here in Figure 3-1

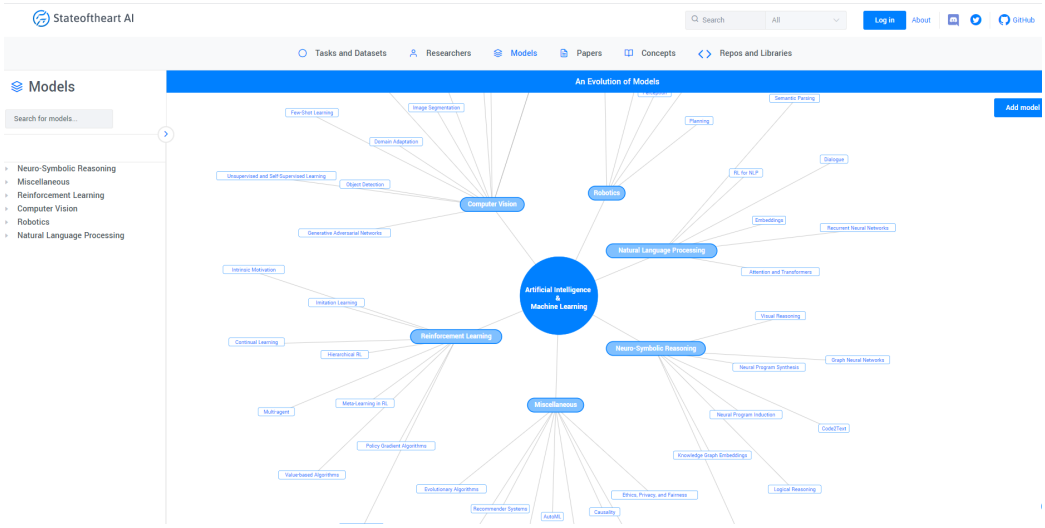


Figure 3-1: `Stateoftheart.ai` platform, an interactive viewer displaying the taxonomy of Models

Taxonomies are widely used across sciences and in daily life. In the context of AI Research they have been used to classify and organize models, tasks, applications and others (see Appendix C for a collection of taxonomies used in the literature). Furthermore, taxonomies have themselves been objects of analysis, learning and inference in diverse machine learning paradigms; including symbolic, Bayesian and neural approaches proposed [96, 49, 53]. Taxonomies have also been considered important cognitive elements for knowledge representation [103], theory learning [121] and semantic acquisition going back at least to [27]. Some of the work presented in this Thesis on Neuro-Symbolic reasoning, which combines deep learning with symbolic structures, has used taxonomies for the problem of concept learning [20, 21].

More specifically, there are many potential practical benefits of a principled general taxonomy for Artificial Intelligence and Machine Learning (i.e. for education, for popularization, for easing literature reviews, for facilitating navigation, etc). Nevertheless, here we focus on looking for a deeper way of classifying tasks and models from a research standpoint [97]. As argued in [71], a good taxonomy can help research

by pointing to new frontiers and signaling open questions, by suggesting next steps for research, by making analogies between different branches, by clarifying different alternatives facilitating component reuse, and others. Thus, we are inspired by the taxonomy of [71] which aims to "contribute to a more systematic theoretical and empirical foundation for understanding organizational process", in this case in the context of AI Research..

In recent years AI Research has been developing very rapidly. During the last 5 years in particular, in order to track different aspects of this accelerated process, multiple independent surveys, tutorials, websites, repositories are constantly being created and updated. Most research papers devote a section to do a review of the important related references. We aim to create a centralized public taxonomy to serve the role of a library of libraries as a living survey of the field. Next are some guiding considerations.

The MIT Process Handbook for Organizing Business Knowledge We are heavily inspired and guided by the Process Handbook [71] which developed an extensive, publicly available on-line knowledge base of business processes, as well as an interface to maintain, access, and navigate that knowledge. One of the key theoretical concepts involved in the handbook is the notion of *specialization* which breaks a process into different types (as opposed to decomposing it only into its parts). We adopt this as a guiding principle in the creation of the taxonomy of tasks. [1]

The Collective Intelligence Perspective We take the perspective of the AI research community as a collective intelligence system composed of human researchers, computer algorithms, and all the interfaces, incentive structures and coordination mechanisms that tie them together. This system can constitute a powerful "Supermind" [70], with the potential of achieving a new form of superintelligence.

[1] A second important element coming from coordination theory is the notion of *dependencies* between activities and the *coordination processes* that manage them. We attempted to take a first step to expose and classify the variables (dependencies) that allow the compatibility (coordination between models and tasks/datasets, while you can see some partial progress for the case of Computer Vision here `https://github.com/stateoftheartai/sotaai`, this endeavor was not continued.

Evolution of Knowledge and Science of Science Science can be described as a complex network involving scholars, projects, papers and ideas. In recent years, with the growth of digital data, the discipline that studies the evolution and progress of science has converged into a field called Science of Science [41]. Several research investigations have included the study and modelling of the spread of ideas [12], the emergence and development of scientific fields [11], the evolutionary dynamics of cultural change [57], and the topological transition of collaboration networks[13], among others. The work in [5] studies the evolution of the AI ecosystem from a venture and funding perspective. A related work to ours [6], generates visualizations of existing datasets of methods and Machine Learning sub-fields through node-link representations. This work is based on the data presented at `paperswithcode.com` which contains leaderboards of model's performance in various datasets and is a useful resource for the community to know the state-of-the-art across the field. Recent work also uses this resource to present a methodology to study certain dynamics of the AI research community based on co-authorships [74].

These approaches have permitted various research questions which range from specific, such as the work of MIT historian of science David Kaiser, who studies how US physicists grappled with specialization and the (re-)organization of their scientific journals in the face of runaway growth in numbers of new physicists and new articles per year [51]; to general, such as the work of Jurgen Renn [93] who takes a general theoretical perspective to study the co-evolution of epistemic communities and of systems of knowledge. Renn introduces the notion of an Epistemic Network, composed of social, semantic and artifact aspects.

The taxonomy, in particular the evolution of models could serve as an interesting database to ask some of these questions. For example, it can help make analogies by comparing the evolution of different subareas (such as the raise of unsupervised learning in both Natural Language Processing, and more recently, in Computer Vision; or such as interesting parallels between the Teacher-Student paradigms in the sub-fields of intrinsic motivation in RL and semi-supervised learning in Computer Vision).

In this chapter we present a new dataset consisting of taxonomy of models, tasks

and datasets. The taxonomy is really a meta-dataset as it compiles many existing sub-taxonomies across surveys, papers and other sources. The taxonomy intends to contribute to understand how research builds up, the lack thereof is a known problem for academia and for the progress of research more generally [80, 81] and can help understand the structure of the research communities. In the next section we discuss some desirable properties for a taxonomy and a few theoretical considerations. Section 3.3 presents the taxonomy. Section 3.4 discusses some current uses. Section 3.5 discusses some open future directions for improvement. The last section concludes.

3.2 Desirable Properties and Theoretical Considerations

Before introducing the taxonomy, we first describe a set of desirable properties and then some points of theoretical consideration.

3.2.1 Desirable Properties

Compatibility with the current practice of AI research We would like the taxonomy to reflect the current organization of the AI community and the research practice. This is similar to the "Intuitively appealing" desirable criteria from the Process Handbook [71]. To help with this we compiled a list of specific existing taxonomies found throughout the literature (see Appendix C).

Usefulness for human researchers The main target of the taxonomy is the research community, and it should be useful. Among others it could be helpful for doing literature reviews, to understand academic tendencies that can guide the choice of research topics, to do comparisons among the evolution of different areas, to find similar models, tasks and datasets, and to learn about fields outside of one's own field.

Support for automatic inference and reasoning through specialization and inheritance We would like to exploit some of the components of the taxonomy to eventually have the potential to afford for automatic or semi-automatic inference and reasoning. In particular, we would like to create a structure of specialization of tasks and datasets which would allow for inheritance, where specialized tasks automatically inherit properties of the parents. Ideally these properties would additionally guide the specification of the compatibility between models and tasks.

Comprehensiveness While not exhaustive, we would like the taxonomy to be broad and comprehensive of many of the different existing sub-fields that are generally identified with Artificial Intelligence and Machine Learning.

3.2.2 Theoretical Considerations and Limitations

Some points of consideration are in place.

Broader Cognitive Taxonomies It is important to note that a taxonomy of existing AI models and tasks covers only a subset of a broader taxonomy of Intelligence. For example, [70] recently proposed that the basic building blocks of intelligent behavior are *Decide* what actions to take, *Create* options for action, *Sense* the world, *Remember* the past and *Learn* to do all these things better. It would be good to frame and build a taxonomy that can afford a future expansion or an integration with such a broader taxonomy of intelligence.

Parts of Models and the Machine Learning pipeline Besides specialization and inheritance, a second important element of the Process Handbook is its organization of processes into parts and sub-parts [71]. As an example, identifying customers, manufacturing or obtaining products, delivering orders, and receiving payments are all parts of the process "Sell a product", not specializations. In our context, the analogous would be divide machine learning processes into its parts or components. For example a model can be partitioned according to its architecture, its learning

mechanism, its inference algorithm, its application domain, the type of representation it learns, etc... Similarly, the machine learning pipeline usually is composed of a few main parts such as: data preprocessing, inference, loss computation for learning. In this work we are not going very deeply into this categorization, but it could be interesting future work.

Unclear boundaries and pragmatic criteria. The distinction between different "areas" and "sub-areas" within AI is in many cases far from sharp. The factors that draw the boundaries between different research communities are often very diverse. To name a few, they can be based on the application domain, on the historical evolution of an architecture, on how the human mind is perceived, or even on sociological and cultural reasons. Thus, in developing the taxonomy, we drew upon informal knowledge based on how previous surveys, conferences and papers categorize their respective sub-fields, trying to reflect the organization of research itself. For example, the distinction between what constitutes a task and a dataset is itself blurry; even the distinction of Tasks and Models can be blurred (a model can be thought of as a particular specialization of a task, or a "way" of doing a task), and there are several fields that could be described both as tasks or as particular forms of doing algorithms. Moreover the boundaries of tasks and models are some times conceptual, some times empirical, and sometimes both. To be comprehensive we have to use highly pragmatic and subjective, although informed criteria.

3.3 The Taxonomy

3.3.1 Overview

The taxonomy is divided into two types of components: Tasks and Datasets on the one hand, and Models on the other. The taxonomy surveys all the field and was compiled manually based on surveys, conference tracks, tutorial and other sources. The taxonomy of tasks is constructed based on the principle of specialization with an indefinite number of levels that varies for each branch, the leafs of the taxonomy

are the Datasets which can be thought of as the ultimate specialization of a task. On the other hand, the space of existing Models is multi-dimensional and the conceptual partitions within each sub-field follow their own idiosyncrasies. While these taxonomies are intended to be comprehensive, we naturally we have to focus on the most important cannot avoid some degree of subjectivity. To complement this, we additionally present an exhaustive classification of all Models, Tasks and Datasets for every paper accepted to the Neurips2020 Conference. For a general summary of the taxonomy see Table 3.1.

Table 3.1: Summary Statistics of the Taxonomy

Models	**Tasks and Datasets**	**NeurIPS2020**
2545 Models	3545 Datasets	1898 Papers
6 Areas	6 Areas	6 Model Areas
55 Subareas	867 Task Nodes	49 Subareas
184 Conceptual Categorizations		2057 Datasets
5831 Parent-Child Model Relations		493 Papers with no Dataset
+ "Classical" area		+ "Classical" area

3.3.2 Models

The space of existing Models is very multi-dimensional and the conceptual partitions within each sub-field follow their own idiosyncrasies. For this reason it would be hard to conceptually organize the existing methods in a unique standardized way. We decided to use fixed hierarchy composed of areas, subareas and sub-sequential further conceptual categorizations. The dimensions that guide the conceptual categorizations in the research communities vary across areas. For some areas like Natural Language Processing (NLP) or Computer Vision (CV), the properties of the Architecture of the models is crucial: whether it is a Recurrent Neural Network, a Convolutional Neural Network, a Transformer, etc... In contrast, the subcommunities within Reinforcement Learning (RL) are organized more based on the scope and method: Intrinsic Motivation, Imitation Learning, Hierarchical Reinforcement Learning, etc... The model

Table 3.2: Model Taxonomy. Areas and Subareas

Neuro-Symbolic Reasoning	**Computer Vision**
Graph Neural Networks	Unsupervised and Self-Supervised Learning
Logical Reasoning	Autoencoders
Visual Reasoning	Generative Adversarial Networks
Neural Program Synthesis	Few-Shot Learning
Program Synthesis (Search)	Semi-Supervised Learning
Neural Program Induction	Convolutional Neural Network Architectures
Knowledge Graph Embeddings	3D
Code2Text	Video
	Object Detection
	Visual question Answering
	Image Segmentation
	Domain Adaptation

Natural Language Processing	**Reinforcement Learning**	**Robotics**
Attention and Transformers	Model-based RL	Systems
Embeddings	Policy Gradient Algorithms	Control
RL for NLP	Value-based Algorithms	Manipulation
Semantic Parsing	Multi-Agent	Navigation
Dialogue	Continual Learning	Perception
Recurrent Neural Networks	Meta-Learning in RL	Planning
	Imitation Learning	
	Intrinsic Motivation	
	Hierarchical RL	

Miscellaneous	**Classical**
ML for Science	Deep Learning
Causality	Machine Learning
AutoML	Statistics
Recommender Systems	Optimization
Ethics, Privacy, and Fairness	Computer Science
Computational Audition	Game Theory
Artificial Life	
Evolutionary Algorithms	
Climate Change	

organization in other areas like Robotics and Computer Vision are often divided depending on the Task for which they are used, for example for parsing Videos, or for Image Segmentation, or for Image Classification. See Table3.2 for a list of all areas and subareas.

Models Causal Evolution We also trace the historical evolution of some of the most important models. This is something that all research articles do, normally in a section called Literature Review. We are not aware of any other dataset like this one, which unlike a citation graph, was compiled manually by exposing the casual links that lead from one model to another (See figure 3-2 for an example).

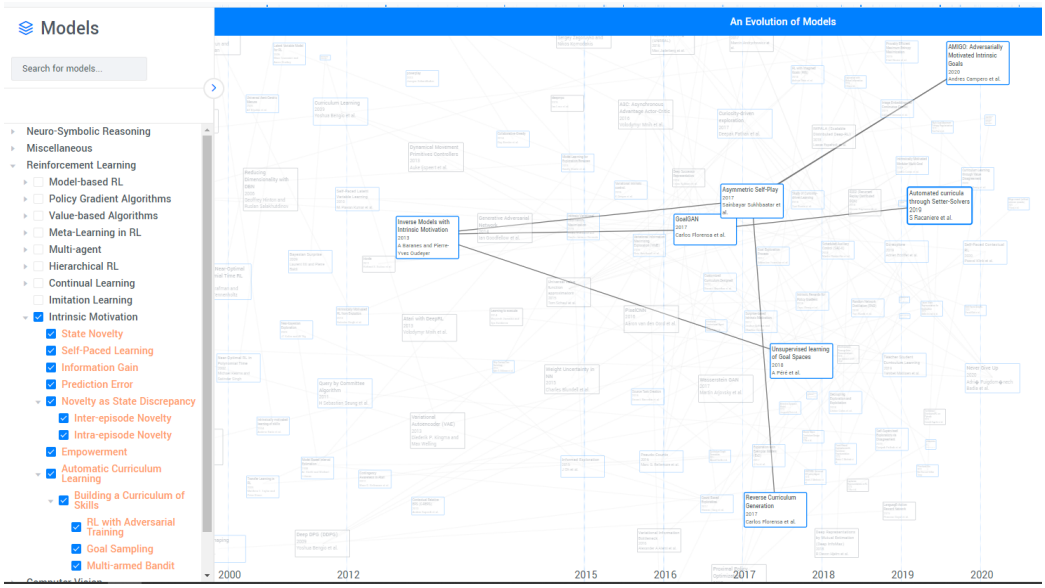


Figure 3-2: Evolution of models for Intrinsic Motivation within RL

3.3.3 Tasks and Datasets

The current practice of AI and Machine Learning heavily relies on the performance of different models on various datasets (although richer forms of meta-learning and continual learning in virtual worlds are slowly becoming more common). This allows for a structured organization of the space of tasks which affords specialization.

The taxonomy we built contains a nested structure of subtasks and subsubtasks with an indefinite number of levels, which varies from branch to branch. Datasets are

the ultimate specializations of tasks and constitute the leafs of the taxonomy. See figures 3-3 and 3-4 for example views of the general taxonomy and the Neuro-Symbolic field respectively.

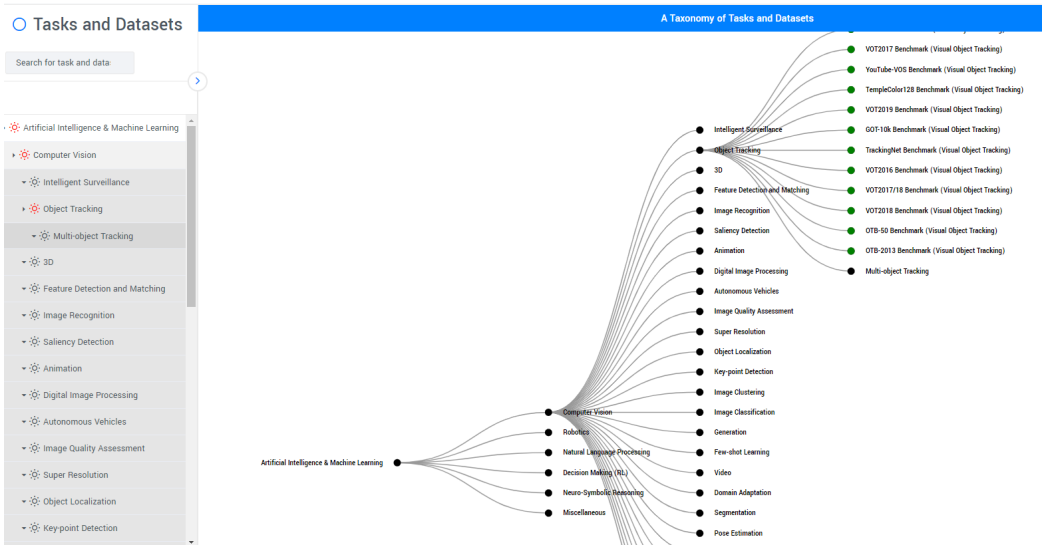


Figure 3-3: View of the taxonomy of Tasks, Subtasks and Datasets (in green). Notice that any given level of the taxonomy can contain both datasets and tasks depending on whether it has further subspecializations

The taxonomy is constructed to afford specialization, and therefore inheritance. This with the intention of enabling the construction of a framework that can eventually be used for semi-automatic inference and easy navigation by exposing the properties that distinguish the different nodes of the taxonomy such as the dimensionality and structure of the output, the size and number of nodes in the input, and even the structure of the data (an environment as a Markov Decision Process for RL, a dataset of images for Computer Vision, and a Graph for some more structured tasks). The taxonomy is built based on many survey papers, conference tracks, tutorials an others. Part of the taxonomy is a reorganization of some of the data contained in `paperswithcode.com`. See `Stateoftheart.ai` and Appendix C for the full taxonomy.

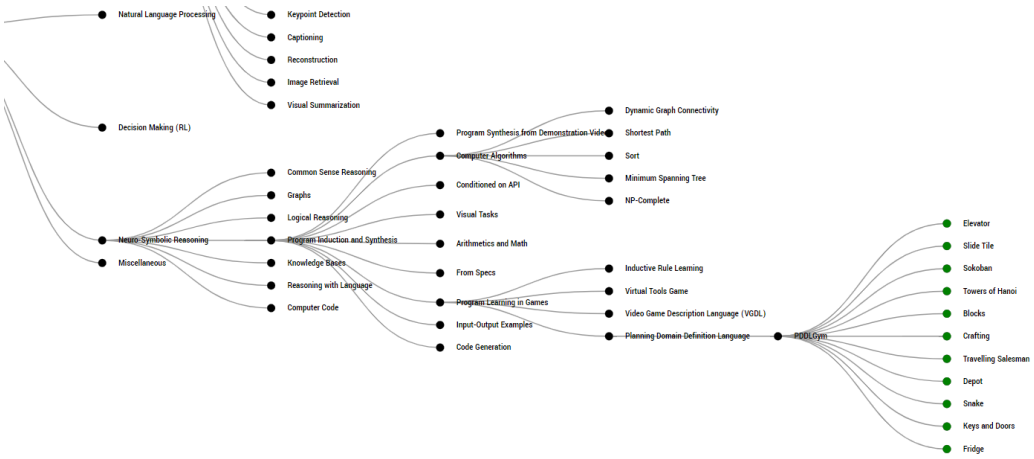


Figure 3-4: Arbitrary expansion of the Neuro-Symbolic Reasoning branch as an example.

3.3.4 NeurIPS 2020 Dataset

To complement with a well defined and exhaustive dataset, we manually analyzed every one of the 1898 papers accepted to the NeurIPS 2020 conference and classified them according to our taxonomy of models. We also classified every task and dataset measured in the paper, naturally some papers are of a more theoretical nature and do not contain any particular dataset or tasks. See Table 3.3 for summary statistics and Appendix C for links to the full taxonomy.

Table 3.3: NeurIPS 2020 Dataset. Summary counts are followed by Model and Dataset counts by area. Subareas of "Miscellaneous" and "Classical" are shown in italics.

Concept	**Total Count**	
Papers	1898	
Model SubAreas	49	
High-level Tasks	616	
Unique Datasets	2057	
Total Dataset Count	4504	
Papers with no Dataset	493	
Area	**Model/Paper Count**	**Dataset Count**
Neuro-Symbolic	105	401
Computer Vision	309	1887
Natural Language Processing	51	240
Reinforcement Learning/RL	227	429
Robotics	9	18
Total Miscellaneous	254	1036
ML for Science	-	*365*
Ethics, Privacy and Fairness	-	*97*
Time Series	-	*96*
Recommender Systems	-	*39*
Causality	-	*32*
Speech	-	*27*
AutoML	-	*16*
Music and Audio	-	*11*
Other Miscellaneous	-	*353*
Total Classical	943	-
Machine Learning	*479*	-
Deep Learning	*282*	-
Optimization	*86*	-
Statistics	*45*	-
Game Theory	*27*	-
Computer Science	*19*	-
Other	*5*	-

3.4 Current Potential uses of the Taxonomy

There are many potential practical benefits of a principled general taxonomy for AI and Machine Learning(for education and popularization, for easing literature reviews, for facilitating navigation, etc). Nevertheless, the focus of this taxonomy is to look for a deeper more systematic way of classifying tasks and models. It is designed to contribute to understand how research builds up, the lack of which is a known problem for the progress of research and general advancement of the field [80, 81].

The current taxonomy can be used to point to new frontiers by signaling open questions, suggesting next steps for research, making analogies between different branches, clarifying different alternatives and facilitating component reuse. It is already constantly being used to to navigate the state-of-the-art of AI [2]. It helps avoid the redundancy across the many existing sources while simultaneously providing unified standardized source that helps organize and understand the structure of the field.

While the contribution of this chapter is the taxonomy on itself and does not include a formal evaluation, to highlight how the taxonomy performs in terms of the desirable properties presented above we mention some of its ongoing current uses:

The Center for Science and Technology from Georgetown University has been a collaborator to continue developing some of the mappings of this taxonomy, interested in the evolution of science considering questions such as knowledge diffusion and concept propagation within the framework of **Science of Science**[41]. The Center for Humans and Machines from the Max-Planck Institute for Human Development asked for access to our API to study research questions related to **Epistemic Evolution** [93] which can include the study and modelling of the spread of ideas [12], the emergence and development of scientific fields [11], and the topological transition of collaboration networks[13], among others. Researchers from Harvard developing the Data Science Reference Framework[3] have requested access to the code of `Stateoftheart.ai` to use as a template and base to integrate into a broader **organizational computational tool** for Data Science.

[2]Currently 40 daily users navigate the taxonomy through the platform `Stateoftheart.ai`
[3]`https://www.youtube.com/watch?v=YDLzciKzJug`

3.5 Improvements and Open Directions

While the taxonomy is already finding some uses for the research community, it can be expanded in several directions.

Cross-Taxonomy Integration It would be enriching to integrate Models, Tasks, Datasets and even Concepts and Researchers into a single Knowledge Graph, exposing more complex connections between the different constituents.

Exploiting analogies Several analogies can be made by comparing the evolution of different subareas examples include the raise of unsupervised learning in both Natural Language Processing, and more recently, in Computer Vision, both using very similar loss functions; or - as we learned while constructing the taxonomy - the interesting parallels between the Teacher-Student paradigms in the sub-fields of intrinsic motivation in RL and semi-supervised learning in computer vision. It would be interesting to explore this systematically.

Richer Taxonomy of Tasks Instead of constraining the organization of tasks to a tree. It would be interesting to create a richer structure that allows navigating the taxonomy horizontally (i.e. Task A + "Language" $\rightarrow$ Task B)

Multiple views It would be beneficial to enable multiple alternative views which can emphasize different aspects and dimensions of the taxonomy depending on the use-case. For example, one such dimension could favor the conceptual vs the causal. Another would be the sparse-dense dimension, allowing to view either only the most important models, or showing all the specific models in the evolution of a sub-field.

Incorporating Machine Learning Various machine learning algorithms could potentially be helpful (e.g. graph embeddings, graph neural networks, topic models) to both exploit the taxonomy and to expand it.

Synthesizing Programs The taxonomy could ideally be used to help synthesize code, for example by navigating the space of tasks based on exposed attributes, which could in turn help select the compatible relevant models. In order to do this, the system should be able to both suggest alternative configurations and realize preliminary evaluations of these alternatives. From the community perspective, there could be analogies between specific program synthesizers and the collective system. For example, making an analogy with [33], there would be a library of models (library of primitives) which grows and develops as researchers contribute with code and nodes, this library would be used to generate programs by searching over the taxonomy, in the process improving itself. Inheritance could allow for program adjustments such as substituting a task by its specialization, and such as finding a model compatible with a task based on specific attributes.

3.6 Conclusion

This taxonomy is really just a starting point. While the current use of the taxonomy is by human researchers, it could be enhanced to be used by semi-automated algorithms and there are many dimensions and ways to continue expanding and enhancing its collective aspects. Ultimately the taxonomy is an organized Knowledge Base of AI research with an interface to interact with it. Could it help accelerate research? Its possible. For example, understanding from an evolutionary perspective how different methods and models have evolved could suggest new directions worth studying.

This work is at the intersection of two of the most powerful sources of superintelligence: Artificial Intelligence and Collective Intelligence. The Supermind perspective puts an emphasis on the role of groups [70] which in our case are composed by the inter-operation of both humans and machine nodes. As this collective system continues to advance, the taxonomy too could be expanded or connected into a broader cognitive taxonomy, one that goes beyond AI research, into a general taxonomy for the Design of Superminds.

Acknowledgements

The taxonomy presented was constructed in collaboration with Santiago Renteria.

The taxonomy is available at the Open Science Framework (Appendix C). It is also available through an interactive viewer developed at **Stateoftheart.ai**, built in collaboration with Hugo Ochoa, Eduardo Espinosa, Jorge Delgadillo, Antonio Teran, Luis Lara, Liuba Orlova and Cuco Resendiz; with logistic help from Lynne Bairstrow, Luisfe Nuñez, Elena Elorza, Jesse Parent and Gussi Espinoza. This platform was developed by Stateoftheart AI PBC, a coorporation for which the author of this Thesis is the founder and CEO.

Financial support for this work was provided in part by the Toyota Research Institute (Grant Nos. LP-C000765-SR and PO-000889).

Chapter 4

Human-AI Combination for Generating Software

4.1 Introduction

Starting in the 1950s, when Alan Turing proposed his famous "Turing test," he inspired generations of computer scientists with the vision of developing artificially intelligent computers that would someday equal human performance [48, 68]. Substantial progress has been made in realizing this vision, and there have been many studies comparing computer performance to that of humans [120, 109, 63]. But no computer today can equal human performance in all ways, and many experts believe that such an accomplishment is still far in the future [15, 63].

In the meantime, people and computers working together in various ways have become commonplace in many aspects of modern life, and there have been a number of studies evaluating the performance of such human-computer combinations [119, 122, 47]. But it is surprising, that there isn't yet a widely recognized scientific equivalent of the Turing test to systematically measure the synergistic improvements in how well people and computers together can perform tasks better than either could alone or better than some other relevant benchmark.

Here we propose such a test and demonstrate its use in two studies, both evaluating how people combined with a state-of-the-art AI program can perform a typical

software development task. We also discuss various potential benefits of this test, including stimulating progress with contests like the autonomous vehicle contests sponsored by DARPA [17, 7], and providing opportunities for collaboration between computer scientists and social scientists to develop and study not only new technologies themselves but also novel ways of combining humans and computers to use these technologies.

4.2 Approach

There are two simple ideas behind the test we propose:

1. Instead of viewing humans and computers as competitors in performing tasks, view them as collaborators.

2. Instead of viewing human performance as an upper bound, try to maximize the ratio between the performance of a human-computer system and a relevant benchmark such as humans only, computers only, or current practice.

To formalize these ideas, we let

$$X_i = \text{the performance of system i on a given task}$$

$$\rho = \frac{X_i}{X_j}$$

where the values of i and j represent different types of systems such as:

$$H = \textit{human}$$

$$C = \textit{computer}$$

$$HC = \textit{human-computer}$$

$$B = \textit{placeholder for any relevant baseline}$$

In general, we will be interested in maximizing

$$\rho = \frac{X_{HC}}{X_H}$$

With this formulation, when $\rho > 1$, there is some benefit from the human-computer combination relative to the baseline, and when $\rho >> 1$ there is substantial benefit.

We are particularly interested in maximizing a version of ρ that measures the synergy [64, 2] between humans and computers, which we define as

$$\hat{\rho} = \frac{X_{HC}}{max(X_H, X_C)}$$

In this case, when $\hat{\rho} > 1$, the combination of humans and computers performs better than either alone. In other words, when $\hat{\rho} > 1$, there is synergy between the humans and computers.

Of course, to systematically measure any of these kinds of performance, we need to specify the task(s) being performed (e.g., recognizing faces, developing software), the dimension(s) being measured (e.g., speed, cost, quality), and details about the types of systems performing the tasks (such as capabilities of the humans and computers and configurations of the human-computer systems). For example, in specifying dimensions of performance to be measured, there are often multiple dimensions that can be traded off against each other (e.g., speed can often be increased by reducing quality). In applying this test, therefore, it is often useful to specify multi-dimensional performance measures such as maximizing speed subject to the quality being above a specified level.

In the remainder of the paper, we describe how to use this test in two studies of a software development task performed by human-computer pairs. In both studies, the primary computer component is GPT-3 [16], a massive, state-of-the-art AI system that uses machine learning techniques to produce text that is sometimes strikingly human-like. GPT-3 can also produce other kinds of text, such as the code for computer software, and that is the focus of the studies described here.

4.3 Results

In both studies, the task is to develop software code in HyperText Markup Language (HTML) that will generate the web pages shown in Figure 4-1. The web pages include images, links to external sites, buttons, and text of different sizes and forms (see "Web pages" in Materials and Methods).

(a) Saving Planet Earth, with 27 items. (b) Conference Website, with 13 items

Figure 4-1: Webpages used for experiment

4.3.1 Study 1

In Study 1, the subjects are human coders with expertise in HTML coding (see "Subjects" in Materials and Methods). We consider two experimental conditions. In the control condition (human-only or "H"), the human generates all the code using a simple text editor with no AI support. (Of course, the text editor is a simple kind of computer support, but for simplicity, we consider this a "human-only" condition because the computer support here is trivial relative to that in the treatment condition.) In the treatment condition (human-computer or "HC"), the human directs the high-level organization of the website using natural language while leaving the detailed code generation to the GPT-3 AI algorithm (see "GPT-3" in Materials and Methods for details). The order and conditions in which a given subject sees the two tasks are randomized. We do not test a computer-only (or "C") condition because we assume that neither GPT-3 nor any other currently available computer system is capable of performing alone the basic task in this study: looking at a purely visual representation of a web page and generating the HTML code needed to produce that web page.

In other words, for Study 1, the conditions are:

H = human coders with conventional HTML text editor

C = computers (assumed impossible today, so not tested)

HC = human coders with GPT-3 interface

As a performance measure, we focus on the speed of performing each task, subject to a constraint on the quality of the solutions. (See "Web pages" in Materials and Methods for more details about how quality is defined and measured). More

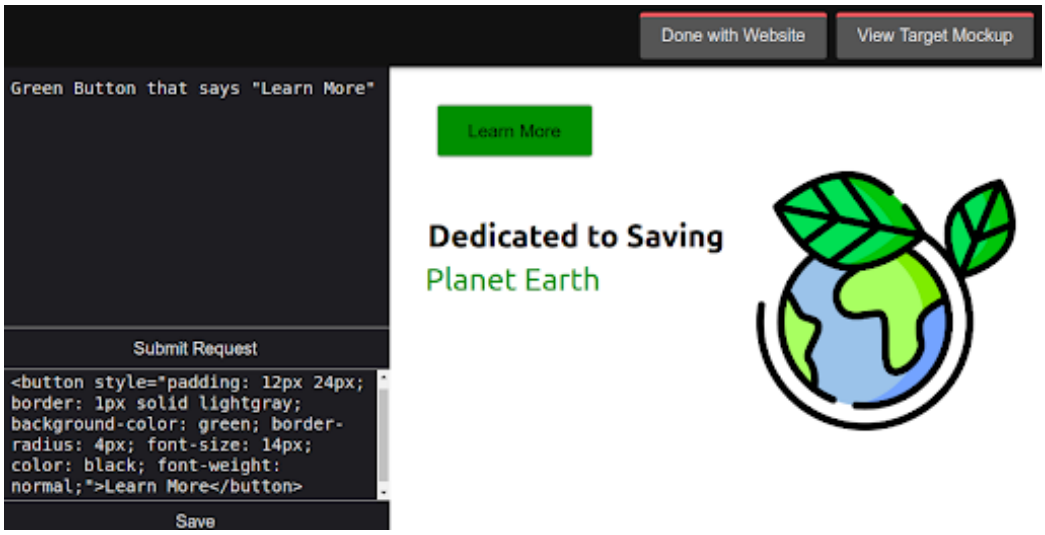


Figure 4-2: Display of the interface for the Human-Computer condition. The top left corner is the field for the subjects to input the natural language instructions. Below on the left are the obtained HTML outputs which the user can modify or delete. On the right is the visual display of the rendered HTML items which can be dragged by the user to different positions. The figure illustrates some of the kinds of elements GPT-3 is able to create in response to simple textual descriptions, such as tables, buttons, images, and other HTML tags.

specifically, we test the hypothesis that the combination of humans and computers (HC) can achieve synergy relative to both humans alone (H) and computers alone (C), subject to an acceptable quality constraint (>80% correct submissions). In other words, we hypothesize that

$$\hat{\rho} = \frac{X_{HC}}{max(X_H, X_C)} > 1$$

and

$$A_{HC},\ A_H > .80$$

where

X_i = speed of system i

A_i = accuracy of system i (submission proportion above quality threshold)

Since $X_C = 0$ by assumption, our hypothesis about $\hat{\rho}$ reduces to

$$\hat{\rho} = \frac{X_{HC}}{X_H} > 1$$

To see whether the hypothesis was confirmed, we consider two alternative ways of calculating $\hat{\rho}$: (a) the ratio/t-test method, and (b) the regression method.

Ratio / t-test method. The simplest way to calculate $\hat{\rho}$ is to compute the ratio of the averages for the HC and H conditions shown in Table 4.1(a):

$$\hat{\rho} = \frac{X_{HC}}{X_H} = \frac{0.032}{0.027} = 1.18$$

We can also use a t-test to compare the means of the two conditions. This shows that the HC condition is significantly faster than the H condition (t=2.407, p=0.017). And Table 4.1(a) shows that the accuracy conditions for both conditions are satisfied $(A_{HC}, A_H > .80)$. In other words, this simple test shows that synergy is present in these human-GPT-3 groups.

Regression method. By doing a more sophisticated regression analysis, it's possible to obtain error bars on the ratio itself and to control for various other factors such as task, task order, and subject. As shown in SI Appendix Section D.1, the ratio $\hat{\rho}$ can be estimated using the following generalized mixed-effects linear regression:

$$y_i = B_0 + B_1 C_i + B_2 O_i + B_3 T_i + v_i + \epsilon_i$$

In order to normalize the distribution, we define y_i as the logarithm of speed for the ith measurement. This also has the benefit of making the regression equation consistent with a multiplicative model of performance that enables us to interpret the exponential of each coefficient, e^{B_i}, as the multiplicative increase in speed due to the different independent variables.

For our purposes, the most important coefficient is $B_1 = log(\hat{\rho})$ the coefficient for the effect of the treatment condition (HC) relative to the control condition (H); B_0 is the fixed intercept; B_2 and B_3 are the fixed coefficients to control for the effect of the task order and of the task; C_i, O_i, and $T_i \in \{0, 1\}$ are indicator variables respectively indicating which of the conditions, which order, and which of the two tasks occurred for the i_{th} measurement; v_i is the random coefficient for the subject of the i_{th} measurement; and ϵ_i is a Gaussian error term.

Our pre-registered hypothesis (`https://aspredicted.org/see_one.php`) was that the ratio $(\hat{\rho})$ obtained from the regression would be greater than 1.0 with 95% con-

fidence. As Figure 4-3 shows, this hypothesis was confirmed. In other words, this regression version of the test also shows that the teams combining humans and GPT-3 obtained statistically significant synergy, improving by about a factor of 1.3 beyond the performance of humans alone. It is interesting to note that, in this case at least, controlling for the other factors such as task and order gives a higher ratio for $\hat{\rho}$ (see Appendix D.2 for full regression results).

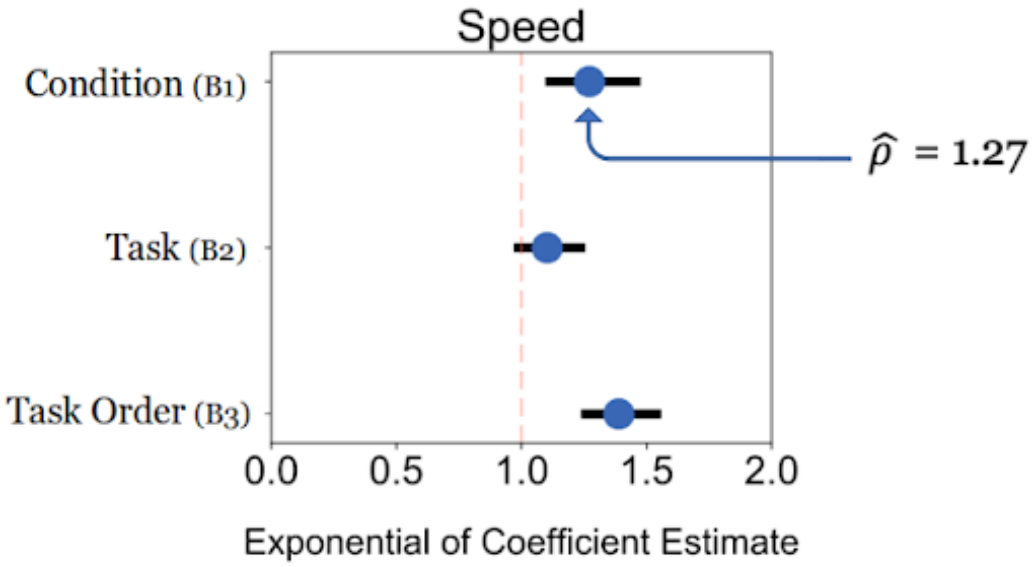


Figure 4-3: Study 1 Regression results

Table 4.1: Empirical Summary of Results. Speed and quality averages per condition (see Appendix D.4 for full distributions).

Condition	Speed (tasks/mins)	Accuracy (% of successful subjects)	# of Observations
(a) Study 1			
Human Only (H)	0.027	84%	99
Human - Computer (HC)	0.032	88%	98
(b) Study 2			
Human - Computer (HC')	0.030	95%	97

4.3.2 Study 2

In Study 2, the subjects are humans who do not know how to code in HTML ("non-coders," see "Subjects" in Materials and Methods), and we have only one experimental condition. In this human-computer (HC) condition, the non-coder subjects do the same two tasks using the same GPT-3 interface as the coders in Study 1 (see "GPT-3" in Materials and Methods). The order tasks are presented to subjects is randomized. As before, we assume that computers could not perform this task alone, and we also assume that the non-coders (who do not know HTML) could not do it themselves alone either.

In other words, for Study 2, the conditions are:

H' = human non-coders [assumed impossible, so not tested]

C' = computers (assumed impossible, so not tested)

HC' = human non-coders with GPT-3 interface

Our pre-registered hypothesis for this study was that the proportion of "successful" subjects would be greater than 50% with 95% confidence, where a subject is "successful" if they achieve at least 90% of the total possible points (`https://aspredicted.org/see_one.php`). Using the data in Table 4.1(b), a one-sample proportion test (as specified in our pre-registration) shows that this hypothesis is strongly confirmed (z = 8.87, p « .0001).

As before, we also test the hypothesis that $\hat{\rho} > 1$, and we observe that

$X_{HC'} = 0.30$ tasks per minute

$X_{H'} = X_{C'} = 0$ (by assumption)

Since the denominator of $\hat{\rho}$ is 0, $\hat{\rho}$ will, of course, always be undefined, regardless of the value of $X_{HC'}$. However, in this context, it is reasonable to interpret $\hat{\rho}$ as an arbitrarily large number, which we denote here as "∞". In other words, we say that

$$\hat{\rho} = \frac{X_{HC'}}{max(X_{H'}, X_{C'})} = \frac{.030}{max(0,0)} = \text{“}\infty\text{”}$$

In practical terms, this means that non-coder humans and the GPT-3 computer system have very strong synergy. They can do something together that neither could do at all alone, and this is a very desirable result from the perspective of the $\hat{\rho}$ test.

Comparing coders to non-coders

Since there could be many factors affecting the differences between the populations of coders and non-coders in our study, we should be cautious about making causal inferences from a comparison of the results between these two populations. It is, however, interesting to calculate a value of ρ by comparing the two here.

As before, we do this in two ways. First, using the ratio/t-test method, to estimate ρ , not $\hat{\rho}$, we obtain

$$\rho = \frac{X_{HC}}{X_{H'}} = 1.071$$

This suggests that the coders are slightly faster than the non-coders, but a t-test shows that the difference between the two is not significant (t=-.718, p = 0.474).

Second, using the regression method, we obtain a similar value for ρ of 1.015 (p =0.902) and the results are also not significant (that is, ρ is not significantly different from 1). In other words, it appears that not only does using GPT-3 allow the non-coders in our study to do a task they could not otherwise have done, it allows them to do the task as fast as experienced coders do. (see Appendix D.3 for full regression results)

Calculating ρ for cost

We have focused so far on using ρ to analyze differences in speed, but it is also possible to use it to analyze other performance dimensions. For example, one obvious question here involves cost: If non-coders are now able to do a task using GPT-3 that previously would have required coders, could it be more economical to use non-coders for this task?

To obtain a suggestive answer to this question, we estimated the costs, C_i, of performing the task in condition i by taking into account (a) the time each subject spent doing the task, (b) the cost of this time based on that subject's individual hourly rates as shown on the UpWork.com site at the time of the experiment, (c) the number of calls to GPT-3 each subject made, and (d) the average cost of GPT-3 calls at the time of the experiment (see "Estimating costs" in Materials and Methods). The

resulting cost estimates are shown in Table 4.2.

Table 4.2: Average costs for each condition. Hourly rate, time spent on eac task, number of calls to GPT-3 and total cost

Condition	Rate/h	Mins/task	GPT3 Calls	Total Cost
NonCoders-GPT3 ($C_{HC'}$)	\$11.40	41.91	28.5	\$10.57
Coders-GPT3 (C_{HC})	\$16.28	38	24	\$8.82
Coders Alone (C_H)	\$16.28	42.47	-	\$10.92

Using the ratio / t-test method to estimate ρ for the two conditions in Study 1, we obtain

$$\rho = \frac{C_{HC}}{C_H} = \frac{10.57}{10.92} = 0.97$$

The t-test in this case is not significant (t = -0.405, p = 0.686). The results of the regression method are similar: $\rho = 0.890$ (p = 0.109). So the use of GPT-3 does not appear to significantly reduce the total cost for coders developing HTML code in this study.

It is also interesting to calculate a different version of ρ for the comparison between coders with GPT-3 in Study 1 and non-coders with GPT-3 in Study 2. Using the ratio / t-test method , we obtain

$$\rho = \frac{C_{HC}}{C_{HC'}} = \frac{10.57}{8.82} = 1.20$$

and the t-test is not significant (t = 1.911, p = 0.057).

However, when we estimate this version of ρ using the regression method, we see that the cost for coders was significantly greater than for non-coders ($\rho = 1.40$, p = .010). In other words, the more sensitive regression test allows us to see that non-coders using GPT-3 can create the HTML code for websites at significantly less cost than experienced coders. We'll discuss some of the implications of this finding in the Discussion section (see Appendix D.7 for full regressions).

4.4 Discussion

In this work, we proposed and demonstrated an approach to systematically studying the performance of human-computer groups. The approach focuses on the ratio (ρ) of improvement that human-computer groups provide relative to other possibilities, similar to how previous studies have looked at large improvements in fields such as hardware and theoretical algorithms [107].

Substantive results

There were three primary substantive results of the empirical work we described. First, we found that the combination of humans coders with GPT-3 was about 1.3 times faster(a 30% improvement) relative to human coders alone. This means there was a statistically significant level of what we define as *synergy* in this combination. It is also interesting to observe that this level of speed improvement (30%) is consistent with self-reported improvements from using Copilot, another GPT-3-based software development tool (see for example [50]). We speculate that this improvement ratio may be characteristic of this class of software development tasks using today's state-of-the-art AI-based tools but that this ratio may improve in the future.

Second, we found that human *non*-coders using GPT-3 were able to do the same task as the coders even though we assume that neither the human non-coders nor today's AI algorithms could have done this task alone. In other words, this human-computer combination had extreme synergy ($\rho >> 1$) which we might call *superintelligence*.

Third, we found that, in the samples of coders and non-coders we studied, the non-coders using GPT-3 were able to create websites about as fast and at significantly less cost than the coders. This would presumably be very desirable for managers and owners of organizations and for the newly-empowered non-coders themselves. But it might be seen as a form of deskilling by the coders whose jobs could now be performed by people with less skill–and for lower compensation [cites for: Attewell, P. (1987). The deskilling controversy. Work and occupations, 14(3), 323-346. Downey,

M. (2021). Partial automation and the technology-enabled deskilling of routine jobs. Labour Economics, 69, 101973.].

Methodological comments

Methodologically, we believe that the test we have proposed provides a simple and broadly comparable measure of the performance of human-computer groups. Unlike the statistical techniques used in most previous empirical studies of human-computer groups, this metric provides a clear and intuitive indication of not only *whether* the human-computer groups were better but also of *how much* better.

More specifically, we described two methods for calculating the ratio ρ. The *ratio/t-test method* is the easiest to apply, and the ratio part of this test can be used even if only average values are available without any details the distribution of the data. The *regression method* is potentially more sensitive because it can control for other variables, such as tasks, subjects, and task order, and it also automatically provides significance tests and confidence intervals for the ρ ratio. But it is somewhat more complicated to perform, and it requires access to the raw data.

Broader implications

A complement to the Turing test. As noted in the Introduction, generations of computer scientists have been inspired by the Turing test to try to create computers that can equal human performance. And to a significant degree, the vision of computers that can replicate–and potentially replace–humans has dominated much public discussion and business decision-making in recent years.

But as many observers since at least the 1960s have pointed out, computers can also be used to *augment* human intelligence, not just to replace it (e.g., [66, 34, 70, 69]. And, perhaps, our focus on the Turing test has even prevented us from recognizing and developing these possibilities in ways that would have been better for business and society [108, 1].

We believe that the test proposed here, by focusing on and quantitatively measuring the *combined* performance of people and machines can help correct this imbalance

and lead to better economic and societal uses of computers.

Contests for the best human-computer performance. For example, one way this test could help stimulate progress is by having contests like the autonomous vehicle contests sponsored by DARPA that were so useful in the development of autonomous vehicles (e.g., [98]). But in this case, the contests would not be about seeing which teams could create computers to equal human performance; they would be about creating human-computer systems to get the highest possible values of ρ for various tasks.

The contest organizers might, for example, identify the task to be done and specify (a) the pool of human participants, and (b) the specific hardware and software platforms that contestants could use. The goal of the contestants would then be to design human-computer groups that would produce the highest values for ρ.

Among other things, such contests would provide opportunities for constructive collaborations between computer scientists and social scientists that could be of very substantial practical value. These multi-disciplinary teams would not only develop new algorithms and other technologies but also design and study new ways of configuring human-computer systems to perform various tasks more effectively.

In summary, we hope that the work described in this paper can help advance progress toward finding more powerful ways of combining people and computers to do more and more of the tasks in our economies and our societies.

4.5 Materials and Methods

4.5.1 Web Pages

To represent the space of possible web pages, we used two sample pages that varied in difficulty:

1. "Saving Planet Earth" (Fig. 4-1a) was designed by us and contains 27 components used in scoring the correctness of the code generated.

2. "Conference Website" (Fig. 4-1b) was adapted and simplified from a task given

towards the end of an HTML course called Learn to Code (`https://learn.shayhowe.com/practice/adding-media/index.html` and contains only 13 components.

In addition to the visual images shown in the figures, each image also included pop-up messages to specify the details of active elements such as buttons and links. Subjects could see these messages by rolling their mouse over the images.

Evaluating the correctness of code generated. The correctness of the code generated for each page was scored based on the total number of components in the page. A component was defined as one individual HTML element (e.g., a button or a link), and each component was worth a maximum of 4 points, one point each for:

1. *position* relative to the overall frame of the webpage (0.5 points if the position is slightly off).

2. *content*, text or graphical (0.5 points if mostly correct, e.g., if the correct text says "Over 100 locations" and the subject said "100 locations").

3. *color*, that is, text color, background color, button color, etc.

4. *functionality*, that is, what the component does (i.e., text just displays text, links should link to external pages).

Since each component was worth a maximum of 4 points, the image in Fig. 2a was worth 27x4 = 108 points, and the one in Fig. 2b was worth 13x4 = 52 points. We consider a submission "correct" if it receives at least 90% of the maximum number of points available for it.

All the code generated was evaluated by the same person for consistency. A second person rated 10% of the code submissions to measure inter-rater reliability. As expected, following the rubric, both raters had an almost perfect overlap never differing by more than one point out of the 52 and 108 total points per web page.

Quality Threshold. For accuracy, we calculate the proportion of submissions which are correct as defined above, and we expect at least 80% of submissions in a condition to be "correct."

Level of Effort. For all analyses, we excluded any submissions in which the subject didn't put effort, defined as spending less than 10 minutes and using less than 7 calls to GPT-3 for the "human-computer" ("HC") condition; and as spending less than 10 minutes and generating less than 15 lines of code for the "human-only" ("H") condition. These are reasonable numbers below which the task cannot be performed.

These criteria led to excluding 3 out of 200 submissions in Study 1 (2 in condition HC, 1 in condition C), and 3 out of 100 submissions in Study 2 (2 from the same person).

4.5.2 Subjects

Recruiting. Subjects were recruited from the online labor market UpWork.com. For the posting requesting "coders," 100 subjects were recruited, and for the one requesting "non-coders," 50 were recruited. See Appendix D.8 for full postings.

Screening. The status of subjects as coders or non-coders was confirmed by using text message conversations with the subjects and by reviewing their resumes and biographical information on UpWork. The software interface used in the experiment also asked the subjects to self-classify as a non-coder or coder. Subjects who did not self-classify for the group they were recruited for were not included in the study.

4.5.3 GPT-3

GPT-3 description. GPT-3 stands for "Generative Pre-trained Transformer 3" [16]. It is a massive AI system that uses autoregressive machine learning techniques to generate many kinds of text. With over 175 billion parameters, it was extensively trained on a vast corpus of text including all of Wikipedia, many books, and much more material from the Internet. It has outperformed existing state-of-the-art models in many benchmarks. Particularly impressive and relevant is its performance in the few-shot domain, where the model is conditioned to a specific task by providing only a few examples.

Use of GPT-3 in these studies. We developed special software for our subjects to

use as an interface to GPT-3. This software uses the OpenAI API to interface with GPT-3, and it provides two primary additional features:

1. *Parameters and prompts.* To perform specific tasks, GPT-3 needs some detailed parameters and (usually) some examples to indicate what kinds of text outputs the system should generate in response to various kinds of inputs. See SI Appendix D.5 for details of the parameters and examples we provided.

2. *User interface.* The user interface for GPT-3 that was used by subjects in both studies is shown in Figure 4-2. It works as follows: First, the human subject describes in English text a component needed in the web page (e.g., "Green button that says 'Learn More'" as shown in the upper left corner of Figure 4-2). Then the GPT-3 system automatically generates the HTML code needed to produce that component (e.g., the code shown in the lower left corner of Figure 4-2). And, simultaneously, the interface we created adds that component to the replica of the web page shown on the right side of the screen (e.g., the green button shown in the middle of Figure 4-2). If the human thinks the displayed component is correct, the human can then move that component to its proper location in the replica of the web page. If not, the human can (a) delete the component and try again to describe in English what the component should be, or (b) modify the HTML code directly (if the human knows enough to do so).

4.5.4 Estimating Costs

We compute the total cost per task per subject for the different conditions by computing the human cost which depends on the time spent in a task and the hourly rate per subject; and the GPT-3 cost based on the number of calls and the average cost per call. The Davinci engine is $.06 per 1000 tokens, each call uses an average of 66 tokens plus an additional 578 tokens for the prompting. The average cost per GPT-3 call is $.039.

4.5.5 Instructions and Incentives

Instructions. All subjects were provided with the following instructions:

"In this experiment, you are asked to solve two different problems. The main goal of each problem is to try and replicate a mockup website as accurately as possible. You can receive up to $30 for participating in this experiment.

- The base amount will be $10 for completing both problems.
- You will receive an additional bonus of up to $10 dollars per problem, conditional on getting it right and depending on how fast you do it. Before starting the actual problem you will have some time with a practice mockup to get familiar with the interface, so that you don't lose your time in the actual problem when it counts towards the amount of bonus you get (the faster you do it the bigger the bonus)".

Payment. Each subject received a base payment of $10 for participating in the experiment. They received no bonus for incorrect submissions. For correct submissions, their bonus was determined according to the folllowing rule:

- $10.00 if done more than 10 minutes faster than the average
- $7.50 if done between the average and 10 minutes faster than the average
- $5.00 if done in a time between the average and 10 minutes slower than the average.
- $2.50 if done more than 10 minutes slower than the average

Note that this detailed payment rule was not provided to the subjects. They received only the instructions specified above.

Training Session. Subjects were encouraged to practice using a simple example web page they could try to replicate with unlimited time. The performance on this training session was not evaluated. Subjects were additionally provided with videos of how to interact with the interface.

Acknowledgements

The unpublished work on this chapter was done in collaboration with Thomas Malone, Haoran Wen, Jaeyoon Song, Raza Abbas, and Abdullah Almaatouq as part of the research for the Center for Collective Intelligence at MIT. The interface was built by Haoran Wen.

Financial support for this work was provided in part by the Toyota Research Institute (Grant Nos. LP-C000765-SR and PO-000889).

Appendix A

ILP Task Descriptions

A.1 ILP Tasks

Description of tasks and the proto-rule templates used during training. More details are in [36]:

Predecessor In this task we aim to learn the predecessor relation, $predecessor(X, Y) \leftarrow succ(Y, X)$, from basic arithmetic facts $\{zero(0), succ(0, 1), suc(1, 2), ...\}$. We use the following template:

$$F(X, Y) \leftarrow F(Y, X) \qquad (9)$$

Even-Odd In this task we aim to learn the *even* predicate. Here the background knowledge is the same as in Predecessor (above). We must include an extra auxiliary predicate that learns to encode the relation *odd*. We use the following templates:

$$F(X) \leftarrow F(X) \qquad (1)$$

$$F(X) \leftarrow F(Z), F(Z, X) \qquad (2)$$

$$F(X) \leftarrow F(Z), F(Z, X) \qquad (2)$$

Even-succ2 Described in Experiments in Section 2.2.

$$F(X) \leftarrow F(X) \quad (1)$$
$$F(X) \leftarrow F(Z), F(Z, X) \quad (2)$$
$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \quad (3)$$

Less Than Here we aim to learn the *lessThan* relation. Background knowledge is the same as in the tasks above. A possible solution would be:

$lessThan(X, Y) \leftarrow succ(X, Y)$
$lessThan(X, Y) \leftarrow lessThan(X, Z), lessThan(Z, Y)$

We used these templates:

$$F(X, Y) \leftarrow F(X, Y) \quad (5)$$
$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \quad (3)$$

Fizz As in the children game Fizz-Buzz, numbers that are divisible by three should be classified as Fizz. These are the template protorules used during training:

$$F(X) \leftarrow F(X) \quad (1)$$
$$F(X) \leftarrow F(Z), F(Z, X) \quad (2)$$
$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \quad (3)$$
$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \quad (3)$$

Buzz Following the same logic as in Fizz, we used the following templates:

$$F(X) \leftarrow F(X) \quad (1)$$
$$F(X) \leftarrow F(Z), F(Z, X) \quad (2)$$
$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \quad (3)$$

Member Here we aim to learn $member(X, Y)$, which is true if X is an element of list Y. The background knowledge encodes values on a list by using two predicates: $cons(X, Y)$ which is true if node Y is after list X (lists are terminated with the null node 0); and $value(X, Y)$ which is true if the value of node X is Y. One possible solution is:

$member(X, Y) \leftarrow value(Y, X)$
$member(X, Y) \leftarrow cons(Y, Z), member(X, Z)$

We used these templates:

$$F(X, Y) \leftarrow F(Y, X) \tag{9}$$
$$F(X, Y) \leftarrow F(Y, Z), F(X, Z) \tag{10}$$

Length The $length(X, Y)$ relation is true if the length of list X is Y. We represent lists in the same way as in the $Member$ task. We required at least one extra intentional predicate $pred1$. One possible solution would be:

$Length(X, X) \leftarrow zero(X)$
$Length(X, Y) \leftarrow cons(X, Z), pred1(Z, Y)$
$pred1(X, Y) \leftarrow Length(X, Z), succ(Z, Y)$

We used these templates:

$$F(X, X) \leftarrow F(X) \tag{8}$$
$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \tag{3}$$
$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \tag{3}$$

Son We aim to learn $sonOf(X, Y)$ relation from family-related facts involving $fatherOf$, $brotherOf$ and $sisterOf$. We required at least one extra intentional

predicate that learns the relation $is - male$. One possible solution would be:

$sonOf(X, Y) \leftarrow father(Y, X), isMale(X)$
$isMale(X) \leftarrow brother(X, Z)$
$isMale(X) \leftarrow father(X, Z)$

We used these templates:

$$F(X, Y) \leftarrow F(Y, X), F(X) \quad (11)$$
$$F(X) \leftarrow F(X, Z) \quad (12)$$
$$F(X) \leftarrow F(X, Z) \quad (12)$$

Grandparent The goal of this task is to infer the grandparent relation from observed mother-of and father-of facts. Our templates were:

$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \quad (3)$$
$$F(X, Y) \leftarrow F(X, Y) \quad (5)$$
$$F(X, Y) \leftarrow F(X, Y) \quad (5)$$

Relatedness $related(X, Y)$ is true if there is an undirected path between X and Y. Background knowledge contains family related facts as in the tasks Son and Grandparent. We used these templates:

$$F(X, Y) \leftarrow F(X, Y) \quad (5)$$
$$F(X, Y) \leftarrow F(X, Y) \quad (5)$$
$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \quad (3)$$
$$F(X, Y) \leftarrow F(Y, X) \quad (9)$$

Father In this task we aim to learn the *Father* relation in challenging set up (incomplete background knowledge and irrelevant facts). We used these template:

$$F(X,Y) \leftarrow F(X,Z), F(Z,Y) \tag{3}$$

Undirected Edge In this task the background knowledge is composed of several $edge(X,Y)$ facts. The goal is to learn $undirectedEdge(X,Y)$ which is true if there is an edge between nodes X and Y regardless of the direction. We used the templates:

$$F(X,Y) \leftarrow F(X,Y) \tag{5}$$
$$F(X,Y) \leftarrow F(Y,X) \tag{9}$$

Adjacent to Red In this example we extend the background knowledge of the example above with color facts: $green(C)$, $red(C)$, as well as $colour(X,C)$ which is true if node X is of colour C. We included one auxiliary predicate that learns the relation $isRed(X)$. One possible solution would be:

$adjToRed(X) \leftarrow edge(X,Y), isRed(Y)$
$isRed(X) \leftarrow colour(X,Y), red(Y)$

We used these templates:

$$F(X) \leftarrow F(X,Z), F(Z) \tag{13}$$
$$F(X) \leftarrow F(X,Z), F(Z) \tag{13}$$

Two Children Here we aimed to learn the *has-at-least-two-children(X)* predicate, which is true if there are at least two facts of the form $edge(X,Z)$. The background knowledge includes *edge* and *neq* (not equals) relations. We included one auxiliary predicate $pred1$. One possible solution would be:

$twoChildren(X) \leftarrow edge(X,Y), pred1(X,Y)$

$pred1(X, Y) \leftarrow edge(X, Z), neq(Z, Y)$

We used these templates:

$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \quad (3)$$
$$F(X, X) \leftarrow F(X, Z), F(X, Z) \quad (15)$$

Graph Colouring The task is to learn the *adj-to-same(X, Y)* which is true if nodes X, Y are of the same colour and there is an edge between them. The background knowledge is similar as in the task *Adjacent to Red*. We included an auxiliary predicate that should learn the relation *same-colour(X, Y)*.One possible solution would be:

$adjToSame(X, Y) \leftarrow edge(X, Y), sameColour(X, Y)$
$sameColour(X, Y) \leftarrow colour(X, Z), colour(Y, Z)$

$$F(X, Y) \leftarrow F(X, Z), F(Y, Z) \quad (10)$$
$$F(X, X) \leftarrow F(X, Z), F(X, Z) \quad (15)$$

Connectedness In this task we want to learn *connected(X, Y)* which is true if there is a sequence of edges connecting nodes X and Y. We used the templates:

$$F(X, Y) \leftarrow F(X, Y) \quad (5)$$
$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \quad (3)$$

Graph Cyclicity In this task the algorithm should learn the concept of cyclicity. This is true of a node when there is a path departing from it and arriving to itself.

The templates used were:

$$F(X) \leftarrow F(X, X) \tag{4}$$

$$F(X, Y) \leftarrow F(X, Y) \tag{5}$$

$$F(X, Y) \leftarrow F(X, Z), F(Z, Y) \tag{3}$$

A.2 Countries

We mimicked some of the templates found in the appendix E of [95]. In order to closely follow their approach, only for this task we enforced some predicates to be the same, so that #1 represents the same predicate across the rule. In detail for each task:

Countries S1

$$\#1(X, Y) \leftarrow \#1(Y, X)$$
$$\#1(X, Y) \leftarrow \#2(X, Z), \#2(Z, Y)$$

Countries S2

$$\#1(X, Y) \leftarrow \#1(Y, X)$$
$$\#1(X, Y) \leftarrow \#2(X, Z), \#3(Z, Y)$$

Countries S3

$$\#1(X, Y) \leftarrow \#1(Y, X)$$
$$\#1(X, Y) \leftarrow \#2(X, Z), \#3(Z, W), \#4(W, Y)$$

A.3 Taxonomy and Kinship

Taxonomy This task was described in section 2.2.4, the dataset can be seen in figure A-1. We used the following templates to perform this task:

$$F(X,Y) \leftarrow F(X,Z), F(Z,Y) \tag{3}$$

$$F(X,Y) \leftarrow F(X,Z), F(Z,Y) \tag{3}$$

$$F(X,Y) \leftarrow F(X,Z), F(Z,Y) \tag{3}$$

$$F(X,Y) \leftarrow F(X,Z), F(Z,Y) \tag{3}$$

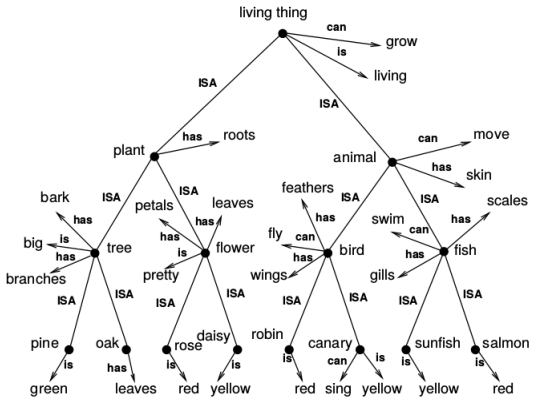


Figure A-1: Bigger animal taxonomy used for the tasks. Contains 4 predicates, 36 constants and 145 facts

Kinship This task was described in section 2.2.4. The theory is in figure A-2

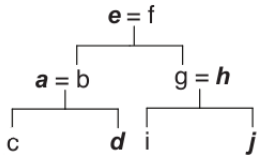


Figure A-2: Inferred family tree. Females shown in bold italics and males in ordinary font.

Appendix B

Learning with AMIGo

B.1 Full results

Tables B.1–B.4 show the final performance of the intrinsic motivation baselines trained using one of four different training regimes enumerated in Section 2.3.4. For each baseline, we train on **KCMedium** and **OMmedium**, and use the best hyperparameters for each task (for that particular baseline and training regime) to train it on the remaining harder versions of those environments (i.e. on **KChard** and **KCharder** or **OMmedhard** and **OMhard**, respectively).

Table B.1: Fully observed intrinsic reward, fully observed policy.

	Medium Difficulty Environments			**Hard Environments**		
Model	**KCmedium**	**OMmedium**	**OMmedhard**	**KChard**	**KCharder**	**OMhard**
AMIGo	.**93** ± .00	.**92** ± .00	.**83** ± .05	.**54** ± .45	.**44** ± .44	.**17** ± .34
IMPALA	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00
RND	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00
RIDE	.00 ± .00	.04 ± .04	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00
Count	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00
ICM	.00 ± .00	.01 ± .01	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00
ASP	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00	.00 ± .00

For IMPALA, the numbers reported for **KCmedium** and **OMmedium** are from the experiments in [91], while the numbers for the harder environments are presumed to be .00 because IMPALA fails to train on simpler environments.

Table B.2: Partially observed intrinsic reward, fully observed policy.

	Medium Difficulty Environments			Hard Environments		
Model	**KCmedium**	**OMmedium**	**OMmedhard**	**KChard**	**KCharder**	**OMhard**
RND	$.64 \pm .09$	$\mathbf{.01} \pm .01$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$
RIDE	$\mathbf{.84} \pm .02$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$
COUNT	$.45 \pm .26$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$
ICM	$.42 \pm .21$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$

Table B.3: Fully observed intrinsic reward, partially observed policy.

	Medium Difficulty Environments			Hard Environments		
Model	**KCmedium**	**OMmedium**	**OMmedhard**	**KChard**	**KCharder**	**OMhard**
RND	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$
RIDE	$\mathbf{.88} \pm .01$	$\mathbf{.94} \pm .00$	$\mathbf{.18} \pm .35$	$\mathbf{.19} \pm .37$	$.00 \pm .00$	$.00 \pm .00$
COUNT	$.01 \pm .01$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$
ICM	$.06 \pm .12$	$.05 \pm .06$	$.16 \pm .32$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$

Table B.4: Partially observed intrinsic reward, partially observed policy.

	Medium Difficulty Environments			Hard Environments		
Model	**KCmedium**	**OMmedium**	**OMmedhard**	**KChard**	**KCharder**	**OMhard**
RND	$.89 \pm .00$	$\mathbf{.94} \pm .00$	$\mathbf{.88} \pm .03$	$\mathbf{.23} \pm .40$	$.00 \pm .00$	$.00 \pm .00$
RIDE	$\mathbf{.90} \pm .00$	$.85 \pm .28$	$.86 \pm .06$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$
COUNT	$\mathbf{.90} \pm .00$	$.04 \pm .04$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$
ICM	$.00 \pm .00$	$.19 \pm .19$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$	$.00 \pm .00$

As a sanity check, we also verified that ASP learns successfully in easier environments not considered here, such as MiniGrid-Empty-Random-5x5-v0, and MiniGrid-KeyCorridorS3R1-v0, to validate the official PyTorch implementation.

Tables B.1–B.4 indicate that the best training regime for the intrinsic motivation baselines (for all the tasks they can reliably solve) is the one that uses a partially observed intrinsic reward and a partially observed policy (Table B.4). When the intrinsic reward is based on a full view of the environment, Count and RND will consider almost all states to be "novel" since the environment is procedurally-generated. Thus, the reward they provide will not be very helpful for the agent since it does not transfer knowledge from one episode to another (as is the case in fixed environ-

ments [8, 19]). In the case of RIDE and ICM, the change in the full view of the environment produced by one action is typically a single number in the MiniGrid observation. For ICM, this means that the agent can easily learn to predict the next state representation, so the intrinsic reward might vanish early in training leaving the agent without any guidance for exploring [91]. For RIDE, it means that the intrinsic reward will be largely uniform across all state-action pairs, thus not differentiating between more and less "interesting" states (which it can do when the intrinsic reward is based on partial observations [91]).

B.2 Hyperparameter Sweeps and Best Values

For AMIGO, we grid search over batch size for student and teacher $\in \{8, 32, 150\}$, learning rate for student $\in \{.001, .0001\}$, learning rate for teacher $\in \{.05, .01, .0001\}$ unroll length $\in \{50, 100\}$, entropy cost for student $\in \{.0005, .001, .0001\}$, entropy cost for teacher $\in \{.001, .01, .05\}$, embedding dimensions for the observations $\in \{5, 10, 20\}$, embedding dimensions for the student last linear layer $\in \{128, 256\}$, and teacher loss function parameters α and $\beta \in \{1.0, 0.7, 0.5, 0.3, 0.0\}$.

For RND, RIDE, Count, and ICM, we used learning rate 10^{-4}, batch size 32, unroll length 100, RMSProp optimizer with $\epsilon = 0.01$ and momentum 0, which were the best values found for these methods on MiniGrid tasks by [91]. We further searched over the entropy coefficient $\in \{0.0005, 0.001, 0.0001\}$ and the intrinsic reward coefficient $\in \{0.1, 0.01, 0.5\}$ on KCmedium and OMmedium. The results reported in Tables B.1, B.2 and B.3 use the best values found from these experiments, while the results reported in Table B.4 use the best parameter values reported by [91]. For ASP, we ran the authors' implementation using its reverse mode. We used the defaults for most hyperparameters, grid searching only over sp_steps $\in \{5, 10, 20\}$, sp_test_rate $\in \{.1, .5\}$, and sp_alice_entr $\in \{.003, .03\}$.

The best hyperparameters for AMIGO and each baseline are reported below:

AMIGO: a student batch size of 8, a teacher batch size of 150, a student learning rate of .001, a teacher learning rate of .001, an unroll length of 100, a student entropy

cost of .0005, a teacher entropy cost of .01, and observation embedding dimension of 5, a student last layer embedding dimension of 256, and finally, $\alpha = 0.7$ and $\beta = 0.3$.

RND: partially observed intrinsic reward, partially observed policy, entropy cost of .0005, intrinsic reward coefficient of .1

RIDE: for **KCmedium**, partially observed intrinsic reward, partially observed policy, entropy cost of .0005, intrinsic reward coefficient of .1; for **OMmedium**: fully observed intrinsic reward, partially observed policy, entropy cost of .0005, intrinsic reward coefficient of .1

COUNT: partially observed intrinsic reward, partially observed policy, entropy cost of .0005, intrinsic reward coefficient of .1

ICM: for **KCmedium**, partially observed intrinsic reward, fully observed policy, entropy cost of .0005, intrinsic reward coefficient of .1; for **OMmedium**: partially observed intrinsic reward, partially observed policy, entropy cost of .0005, intrinsic reward coefficient of .1

ASP: Best performing hyperparameters (in the easier environments) were 10 sp_steps, a sp_test_rate of .5, and Alice entropy of .003. All other hyperparameters used the defaults in the codebase.

B.3 Sample Efficiency

We show, in Figure B-1, the mean extrinsic reward over time during training for the best configuration of the various methods. The first row consists of intermediately difficult environments in which different forms of intrinsic motivation perform similarly, the first two environments require less than 30 million steps while **OMmedhard** and the three more challenging environments of the bottom rows require in the order of hundreds of millions of frames. Any plot where a method's line is not visible indicates that the method is consistently failing to reach reward states throughout its training.

The important point of note here is that on the two easiest environments, **KCmedium** and **OMmedium**, agents need about 10 million steps to converge while on the other four more challenging environments, they need an order of 100 million steps to learn

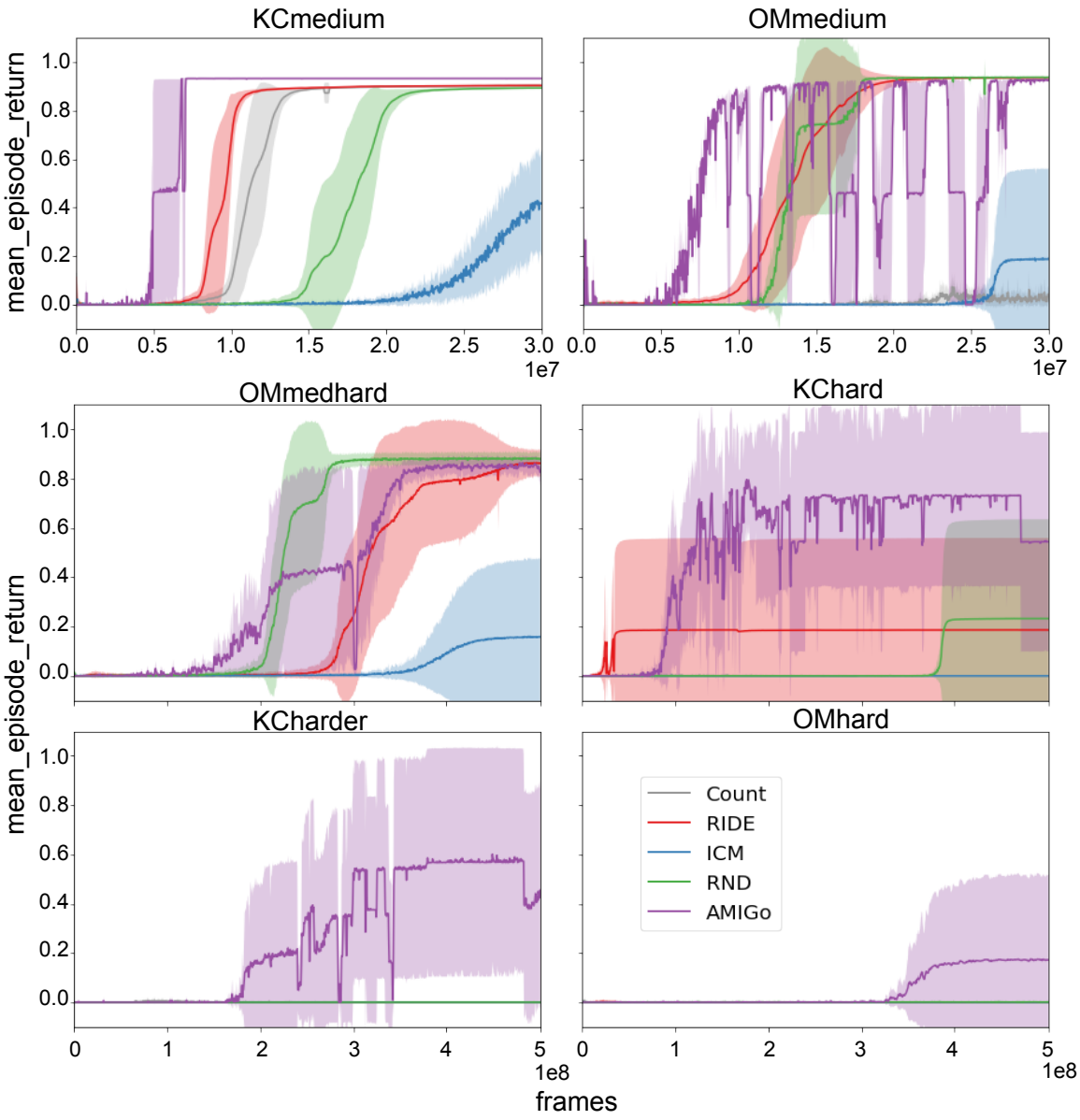


Figure B-1: Reward curves over training time comparing AMIGo to competing methods and baselines. The y-axis shows the Mean Extrinsic Reward (performance) obtained in two medium and four harder different environments, shown for 30M and 500M frames respectively.

the tasks, showcasing AMIGo's contributions not just to solving the exploration problem, but also to improving the sample complexity of agent training.

B.4 Ablation Study

To further explore the effectiveness and robustness of our method, in this subsection we investigate the alternative criteria discussed in Section 2.3.3. We compare the FULL MODEL with its ablations and alternatives consisting of removing the extrinsic bonus (NOEXTRINSIC), removing the environment change bonus (NOENVCHANGE), adding a novelty bonus(WITHNOVELTY).

We also considered two alternative reward forms for the teacher to provide a more continuous and gradual reward than the previously introduced "all or nothing" threshold. We consider a **Gaussian** $p \sim \text{Normal}(t^*, \sigma)$ reward around the target threshold t^*:

$$r^T = \begin{cases} -1 & \text{if} \quad t^+ = 0 \\ 1 + \log_p(t^+) - \log_p(t^*) & \text{otherwise} \end{cases}$$

and a **Linear-Exponential** reward which grows linearly towards the threshold and then decays exponentially as the goal proposed becomes too hard (as measured according to the number of steps):

$$r^T = \begin{cases} e^{-(t^+ - t^*)/c} & \text{if} \quad t^+ \geq t^* \\ t^+/t^* & \text{if} \quad t^+ < t^* \end{cases}$$

We report these two alternative forms of reward as (GAUSSIAN and LINEAR-EXPONENTIAL) in the study below.

Performance is shown in Table B.5 where the number of steps needed to converge (to the final extrinsic reward) is reported. A positive number means the model learned to solve the task, while 0 means the model did not manage to get any extrinsic reward. For all models we encourage goal diversity on the teacher with a high entropy coefficient of .05 (as compared to .0005 of the student).

As the table shows, removing the extrinsic reward or the environment change bonus severely hurts the model, making it unable to solve the harder environments. The novelty bonus was minimally beneficial in one of the environments (namely

Table B.5: Ablations and Alternatives. Number of steps (in millions) for models to learn to reach its final level of reward in the different environments (0 means the model did not learn to get any extrinsic reward). FULL MODEL is the main algorithm described above. NOEXTRINSIC does not provide any extrinsic reward to the teacher. NOENVCHANGE removes the reward for selecting goals that change as a result of episode resets. WITHNOVELTY adds a novelty bonus that decreases depending on the number of times an object has been successfully proposed. GAUSSIAN and LINEAR-EXPONENTIAL explore alternative reward functions for the teacher.

	Medium Difficulty Environments			**Hard Environments**		
Model	**KCmedium**	**OMmedium**	**OMmedhard**	**KChard**	**KCharder**	**OMhard**
FULL MODEL	**7M**	**8M**	**320M**	140M	**300M**	**370M**
NOEXTRINSIC	240M	50M	0	0	0	0
NOENVCHANGE	400M	37M	0	0	0	0
WITHNOVELTY	15M	20M	350M	**100M**	370M	0
GAUSSIAN	320M	60M	0	0	0	0
LINEAR-EXP	0	0	0	0	0	0

KChard) but slightly ineffective on the others. The more gradual reward forms considered are not robust to the learning dynamics and often result in the system going into rabbit holes where the algorithm learns to propose goals which provide sub-optimal rewards, thus not helping to solve the actual task. Best results across all environments in our Full Model were obtained using the simple threshold reward function along with entropy regularization and in combination with the extrinsic reward and changing bonuses, but without the novelty bonus.

B.5 Qualitative Analysis

To better understand the learning dynamics of AMIGO, Figure B-2 shows the intrinsic reward throughout training received by the student (top panel) as well as the teacher (middle panel). The bottom panel shows the difficulty of the proposed goals as measured by the target threshold t^* used by the teacher (described in Section 2.3.3). The trajectories reflect interesting and complex learning dynamics.

For visualization purposes we divide this learning period into five phases: **Phase 1**: The student slowly becomes able to reach intrinsic goals with minimal difficulty.

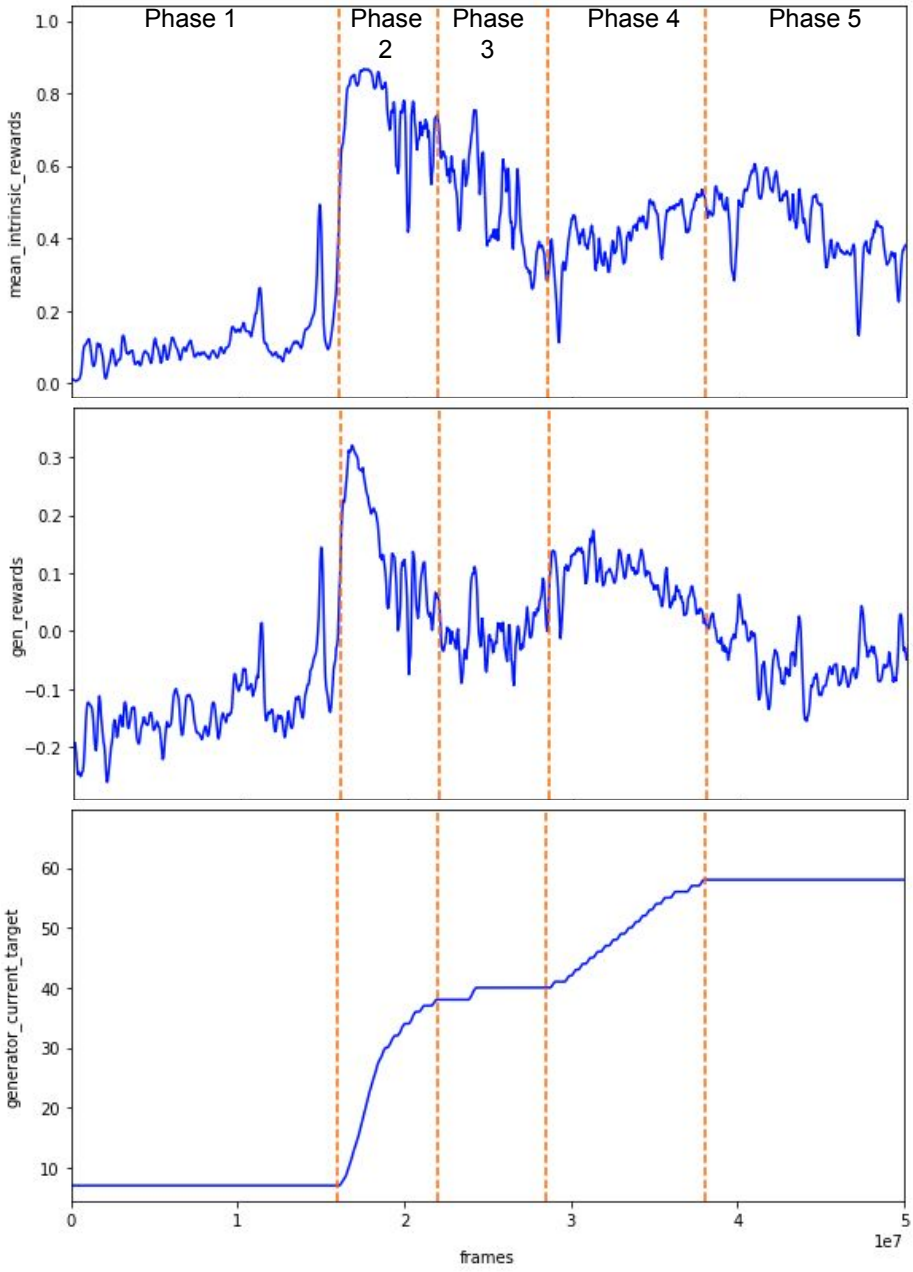


Figure B-2: An example of a learning trajectory on **OMhard**, one of the most challenging environments. Despite the lack of extrinsic reward, the panels show the dynamics of the intrinsic rewards for the student (top panel), for the teacher (middle panel), and the difficulty of the goals captured as t^* (bottom panel).

The teacher first learns to propose easy nearby goals. **Phase 2**: Once the student learns how to reach nearby goals, the adversarial dynamics cause a drop in the teacher reward which is then forced to explore and propose harder goals. **Phase 3**: An equilibrium is found in which the student is forced to learn to reach more challenging

goals. **Phase 4**: The student becomes too capable again and the teacher is forced to increase the difficulty of the proposed goals. **Phase 5**: The difficulty reaches a state where it induces a new equilibrium in which the student is unable to reach the goals and forced to improve its student.

AMIGo generates diverse and complex learning dynamics that lead to constant improvements of the agent's policy. In some phases, both components benefit from learning in the environment (as is the case during the first phase), while some phases are completely adversarial (fourth phase), and some phases require more exploration from both components (i.e. third and fifth phases).

Figure 2-10, presented in Section 2.3.4, further exemplifies a typical curriculum in which the teacher learns to propose increasingly harder goals. The examples show some of the typical goals proposed at different learning phases. First, the teacher proposes nearby goals. After some training, it learns to propose goals that involve traversing rooms and opening doors. Eventually, the teacher proposes goals which involve interacting with different objects. Despite the increasing capacity of the agent to interact with the environment, **OMhard** remains a challenging task and AMIGo learns to solve it in only one of the five runs.

B.6 Goal Examples

Figure B-3 shows examples of goals proposed by the agent during different stages of learning. Typically, in early stages the teacher learns to propose easy nearby goals. As learning progresses it is incentivized to proposed farther away goals that often involve traversing rooms and opening doors. Finally, in later stages the agent often learns to propose goals that involve removing obstacles and interacting with objects. We often observe this before the policy achieves any extrinsic reward.

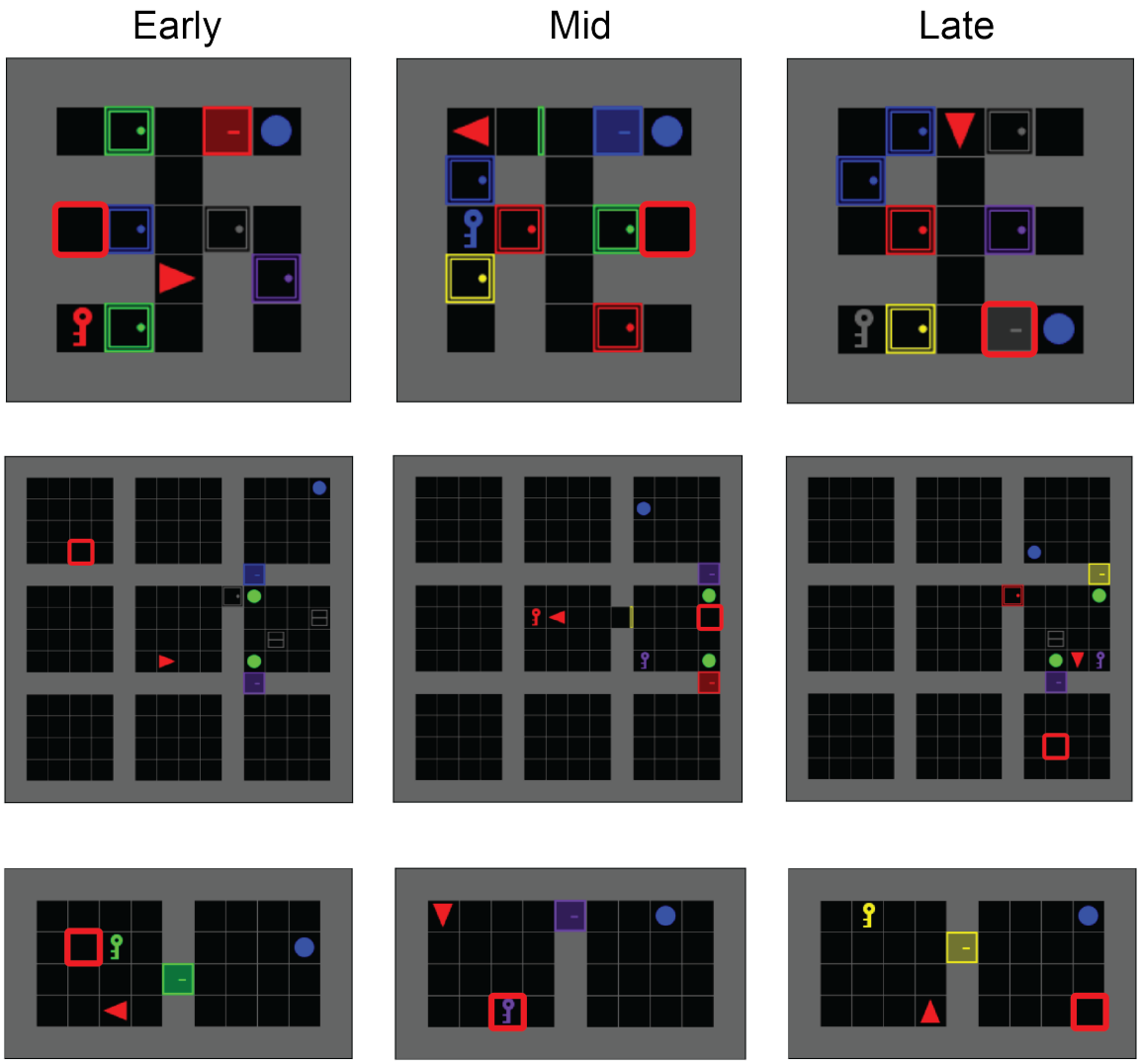


Figure B-3: Some examples of goals during early, mid, and late stages of learning (examples for **KCmedium**, **OMhard** and **OMmedium** are first, second, and third rows respectively). The red triangle is the agent, the red square is the goal proposed by the teacher, and the blue ball is the extrinsic goal.

Appendix C

The Taxonomy

The full taxonomy is in the **Open Science Framework** under the project:

Thesis Andres Campero 2021

`https://osf.io/6bdsu/?view_only=66d52c46cc664349b48540f8f37baea6`

- The full taxonomy and evolution of **Models** can also be found Here. An interactive version can be found in `https://www.stateoftheart.ai/models`

- The full taxonomy of **Tasks and Datasets** can also be found Here. An interactive version can be found in `https://www.stateoftheart.ai/tasks`

- The full **NeurIPS 2020 dataset** can also be found Here.

- The Collection of Taxonomies can also be found Here.

Appendix D

Human-AI Superintelligence

D.1 Regression Derivation

For a more sophisticated estimation of $\hat{\rho}$ we derive the generalized mixed-effects linear regression from a model where we assume the performance is a multiplicative function with the following variables: $s = \beta \frac{a}{d} fne$

s_{ijks} = speed of subject i on task j in condition k and order w

a_i = ability of subject i

d_j = difficulty of task j

g_w = effect on performance of task order w

f_k = effect on performance of condition k

e_{ijkw} = error for subject i on task j in condition k and order w

Namely, relative to some baseline speed β, speed is increased by greater ability a and greater facilitation f and by doing the task for a second time, and it is decreased by increased task difficulty d. We also assume that error is multiplicative, not additive. That is, errors are modeled as percentage increases or decreases not absolute increases or decreases.

With these assumptions, the performance of subject i on task j in condition k and order w can be modeled as follows:

$$s_{ijks} = \beta(\prod_{i=1}^{n} a_i^{q_i})(\prod_{j=1}^{2} d_j^{-T_j})(\prod_{k=1}^{2} f_k^{C_k})(\prod_{w=1}^{2} g_w^{O_w})e_{ijkw}$$

where

n = number of subjects

$$q_i = \begin{cases} 1 & for\, subject\, i \\ 0 & otherwise \end{cases}$$

$$T_j = \begin{cases} 1 & for\, task\, j \\ 0 & otherwise \end{cases}$$

$$C_k = \begin{cases} 1 & for\, condition\, k \\ 0 & otherwise \end{cases}$$

$$O_w = \begin{cases} 1 & for\, order\, w \\ 0 & otherwise \end{cases}$$

This equation can be transformed into a linear regression by taking natural logarithms on both sides:

$$ln s_{ijkw} = ln\, \beta + \sum_{i=1}^{n-1} q_i ln\, a_i + \sum_{j=1}^{m-1} T_j ln\, d_j + \sum_{k=1}^{p-1} C_k ln\, f_k + \sum_{w=1}^{r-1} O_w ln\, g_w$$

where n, m, p and r are respectively the number of sukbjects, tasks, conditions, and order numbers.

Using the observed values for s_{ijkw} and the indicator variables for T_j, C_j, and O_j for each observed value, we can use linear regression to estimate the values of $ln\, \beta$, $ln\, d_j$, $ln\, f_k$, and $ln\, g_w$ which we can relabel as B_0, B_1, B_2 and B_3. Finally, we can take the exponentials of these logs to recover the estimated values of the original variables: β, a_i, d_j, g_w and f_k. We estimate the subject random effects by taking $v_i = q_i ln\, a_i$

D.2 Study 1 Regression

Table D.1: Sudy 1 Results. 197 observations from 100 coders. Showing coefficient estimate, standard error, p-value, 95% confidence interval lower and upper limits, and the exponential of the coefficient which is the multiplicative increase due to the different independent variables

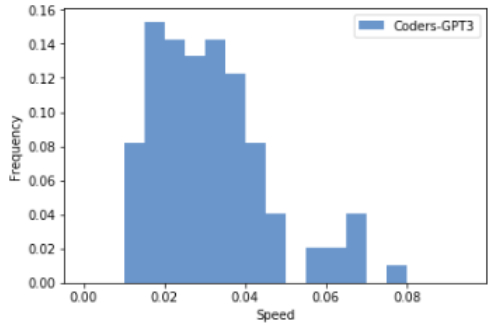
Effect	Estimate	SE	p	95% CI LL	95% CI UL	$e^{Estimate}$
Intercept	3.697	0.066	0.000	3.567	3.827	40.326
Condition	0.240	0.076	0.002	0.091	0.390	1.271
Task	-0.098	0.066	0.137	-0.227	0.031	0.376
Order	-0.329	0.059	0.000	-0.444	-0.214	0.720
Subject RE	0.046	0.063				

D.3 Coders vs Non-coders Regression

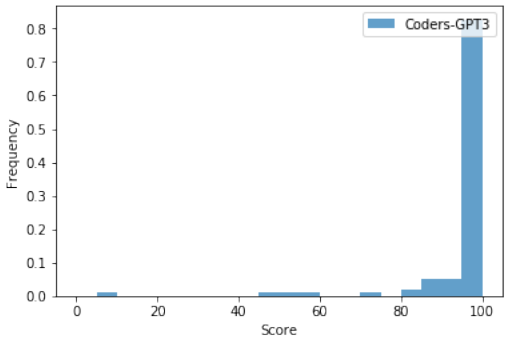
Table D.2: HC condition Regression. 200 observations from 150 subjects. Showing coefficient estimate, standard error, p-value, 95% confidence interval lower and upper limits, and the exponential of the coefficient which is the multiplicative increase due to the different independent variables

Effect	Estimate	SE	p	95% CI LL	95% CI UL	$e^{Estimate}$
Intercept	-3.792	0.099	0.000	-3.986	3.597	0.023
Condition	0.015	0.118	0.902	-0.217	0.247	1.015
Task	0.311	0.049	0.000	0.216	0.406	1.365
Order	-0.221	0.048	0.000	0.126	0.315	0.802
Subject RE	0.410	0.435				

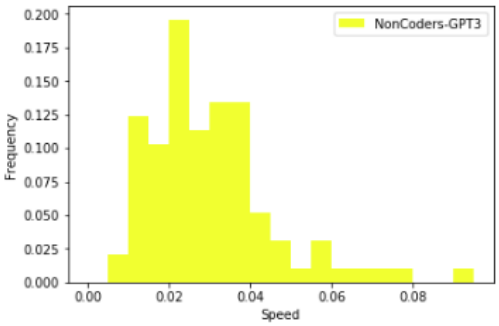
D.4 Quality and Speed Distributions



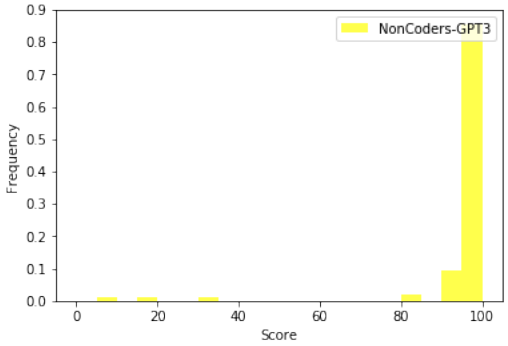
(a) Speed, Study 1 Condition HC (Human Coders + GPT-3)



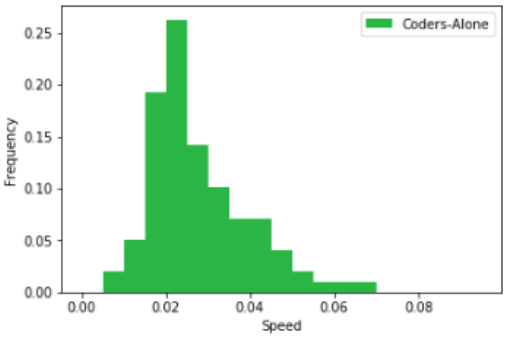
(b) Quality, Study 1 Condition HC (Human Coders + GPT-3)



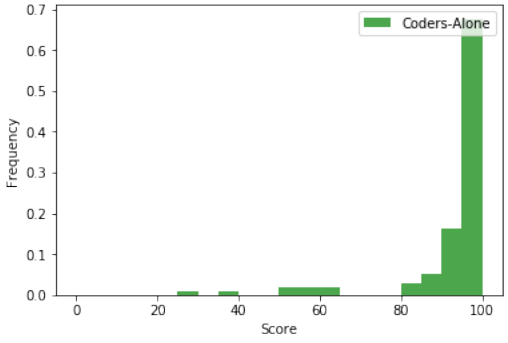
(c) Speed, Study 2 Condition HC' (Human Non-coders + GPT-3)



(d) Quality, Study 2 Condition HC' (Human Non-coders + GPT-3)



(e) Speed, Study 1 Condition H (Human Coders)



(f) Quality, Study 1 Condition H (Human Coders)

Figure D-1: Speed (tasks/min) and Quality Score (% of total points obtained on an individual submission) distribution for different populations and conditions

D.5 GPT-3 Parameters and Prompts

For all our experiments we used the Open-AI API with the Davinci engine, a temperature of 0.3 and a max_token length of 512.

Conditioning GPT-3 is conditioned by prompting where each prompt includes a natural language sentence that the user would type followed by the HTML that would be returned by GPT-3:

- "Hello world button"
  <button style="padding: 12px 24px; border: 1px solid lightgray; background-color: whitesmoke; border-radius: 4px; font-size: 14px; color: black; font-weight: normal;">Hello World</button>

- "pink button that says Banana with white text color"
  <button style="padding: 12px 24px; border: 1px solid lightgray; background-color: pink; border-radius: 4px; font-size: 12px; color: white; font-weight: normal;">Banana</button>

- "Large Google Pixel button with no background"
  <button style="padding: 12px 24px; border: 1px solid lightgray; background-color: transparent; border-radius: 4px; font-size: 24px; color: black; font-weight: normal;">Google Pixel</button>

- "red square"

- "orange rectangle"

- "paragraph that says I like apples"
  I like apples

- "large text that says Black Circle in blue"
  Black Circle

- "Air-Conditioner in brown color"
  Air-Conditioner

- "My birthday is coming up soon!"
  My birthday is coming up soon!

- "text that says Google and links to https://www.google.com/"
  Google

- "FACEBOOK and links to https://www.facebook.com/"
  FACEBOOK

- "image from https://picsum.photos/id/237/200/300"
  <img src="https://picsum.photos/id/237/200/300" style="width: 200px; object-fit: contain;"></img>

- " "

- "<input type="submit" value="Submit!"></input>"
  <input type="submit" value="Submit!"></input>

- "table that says Fruits, Apple, Lime, Blueberries in the first column and Color, Red, Green, Blue in the second column"
  <table style="width: 250px; border-collapse: collapse; border: 1px solid black; text-align: center;"><tr><th>Fruits</th><th>Color</th></tr><tr><td>Apple</td><td
  ><tr><td>Lime</td><td>Green</td></tr><tr><td>Blueberries</td><td>Blue</td>
  </table>

D.6 Study 1 Robustness Checks

Regression for only coders that passed For robustness, we run the same regression only for the population of coders that where successful, defined as obtaining at least 90% of the total possible points on the task. This regression has 169 observations. Table D.3 shows the results. The exponential of the condition coefficient remained similar improving to 1.33 with a Confidence Interval of [1.14, 1.55].

Table D.3: Regression for observations that had a score of 90%+, 169 observations from 96 coders. Showing coefficient estimate, standard error, p-value, 95% confidence interval lower and upper limits, and the exponential of the coefficient which is the multiplicative increase due to the different independent variables

Effect	Estimate	SE	p	95% CI LL	95% CI UL	$e^{Estimate}$
Intercept	3.632	0.068	0.000	3.498	3.765	37.788
Condition	0.287	0.078	0.000	0.134	0.439	1.332
Task	-0.030	0.069	0.667	-0.166	0.106	0.970
Order	-0.347	0.060	0.000	-0.465	-0.230	0.707
Subject RE	0.038	0.073				

Removing Subject Random Effects We analyze a regression without Random Effects for the population of coders using a standard Ordinary Least Squares. The regression has 197 observations from 100 coders. We see that the exponential of the condition coefficient remains almost the same at 1.26 still showing significance but with a slightly bigger variance and a Confidence Interval of [1.06, 1.51].

Table D.4: OLS Regression without Random Effects. 197 Observations from 100 coders. Showing coefficient estimate, standard error, p-value, 95% confidence interval lower and upper limits, and the exponential of the coefficient which is the multiplicative increase due to the different independent variables

Effect	Estimate	SE	p	95% CI LL	95% CI UL	$e^{Estimate}$
Intercept	3.697	0.074	0.000	3.551	3.845	40.326
Condition	0.238	0.090	0.009	0.060	0.416	1.269
Task	-0.100	0.078	0.203	-0.254	0.054	0.904
Order	-0.324	0.069	0.000	-0.461	-0.188	0.723

D.7 Cost Regressions

Table D.5: Coders Cost Regression HC vs H . 197 observations from 100 coders. Showing coefficient estimate, standard error, p-value, 95% confidence interval lower and upper limits, and the exponential of the coefficient which is the multiplicative increase due to the different independent variables

Effect	Estimate	SE	p	95% CI LL	95% CI UL	$e^{Estimate}$
Intercept	2.507	0.068	0.000	2.373	2.640	12.268
Condition	-0.116	0.072	0.109	-0.258	0.026	0.890
Task	-0.147	0.063	0.019	-0.269	-0.024	0.863
Order	-0.311	0.056	0.000	-0.420	-0.201	0.733
Subject RE	0.196	0.159				

Table D.6: Coders vs Non-coders Cost Regression HC vs HC' . HC condition Regression. 200 observations from 150 subjects. Showing coefficient estimate, standard error, p-value, 95% confidence interval lower and upper limits, and the exponential of the coefficient which is the multiplicative increase due to the different independent variables

Effect	Estimate	SE	p	95% CI LL	95% CI UL	$e^{Estimate}$
Intercept	2.135	0.108	0.000	1.922	2.347	8.457
Condition	0.337	0.130	0.010	0.082	0.591	1.401
Task	-0.351	0.046	0.000	-0.441	-0.261	0.704
Order	-0.199	0.046	0.000	-0.289	-0.110	0.819
Subject RE	0.512	0.553				

D.8 Full Postings for Recruiting

Non-Coders Needed For Web Experiment (Open to all with no coding background)

> "We are MIT researchers doing experiments on tools that help create websites. We are looking for non-coders that can help us test our interface. If you are a coder, look for our other job here `https://www.upwork.com/ab/applicants/1425936000678547456/job-details`.
>
> Your job will be to replicate a couple of website mockups. We estimate this would take 45 mins - 1.5hrs of your time for the whole job. We might also want to discuss your experience during the task to improve it.
>
> If you perform well on this job, we will prioritize you for similar future projects.
>
> To be a good fit for this project you should have no experience with coding (If you code, you should look and will do better by doing the "Website Mockup Replication in HTML (For Coders)" experiment).
>
> You will receive a base pay of $10 dollars and will receive a bonus of $10 dollars for each problem you get correctly amounting to up to $30 dollars for the two problems
>
> You will complete this job using the special interface we provide. The link to the interface will be sent to you once we have accepted you for the job.
>
> If you are interested in this project, please submit a proposal."

Frontend Web Developer Needed For Experiment

> "We are MIT researchers doing experiments on tools that help create websites. We are looking for coders with experience in HTML and CSS to test our interface.
>
> Your job will be to replicate a couple of website mockups. We estimate this would take 45 mins - 1.5 hrs of your time for the whole job. We may also want to discuss your experience with the task to improve it.

If you perform well on this job, we will prioritize you for similar future projects.

To be a good fit for this project you should have experience coding in HTML and CSS.

You will receive a base pay of $10 dollars and will receive a bonus of $10 dollars for each problem you get correctly amounting to up to $30 dollars

You will complete this job using the special interface we provide. The link to the interface will be sent to you once we have accepted you for the job.

If you are interested in this project, please submit a proposal."

Bibliography

[1] Daron Aceomglu, Michael Jordan, and Glen Weyl. How ai fails us. *WIRED*, 2021.

[2] Abdullah Almaatouq, Mohammed Alsobay, Ming Yin, and Duncan J Watts. Task complexity moderates group synergy. *Proceedings of the National Academy of Sciences*, 118(36), 2021.

[3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30:5048–5058, 2017.

[4] Andrew G Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer, 2013.

[5] Rahul Basole and AI Accenture. Visualizing the evolution of the ai ecosystem. In *HICSS*, pages 1–10, 2021.

[6] Rahul C Basole and AI Accenture. The ecosystem of machine learning methods. In *HICSS*, pages 1–10, 2021.

[7] Reinhold Behringer, Sundar Sundareswaran, Brian Gregory, Richard Elsley, Bob Addison, Wayne Guthmiller, Robert Daily, and David Bevly. The darpa grand challenge-development of an autonomous vehicle. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 226–231. IEEE, 2004.

[8] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[9] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[10] T. Besold, A. Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K. Kühnberger, L. Lamb, D. Lowd, P. Lima, et al. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv:1711.03902*, 2017.

[11] Luís Bettencourt, David Kaiser, Jasleen Kaur, Carlos Castillo-Chavez, and David Wojick. Population modeling of the emergence and development of scientific fields. *Scientometrics*, 75(3):495–518, 2008.

[12] Luís MA Bettencourt, Ariel Cintrón-Arias, David I Kaiser, and Carlos Castillo-Chávez. The power of a good idea: Quantitative modeling of the spread of ideas from epidemiological models. *Physica A: Statistical Mechanics and its Applications*, 364:513–536, 2006.

[13] Luís MA Bettencourt, David I Kaiser, and Jasleen Kaur. Scientific discovery and topological transitions in collaboration networks. *Journal of Informetrics*, 3(3):210–221, 2009.

[14] G. Bouchard, S. Singh, and T. Trouillon. On approximate reasoning capabilities of low-rank vector spaces. *AAAI Spring Syposium on Knowledge Representation and Reasoning (KRR)*, 2015.

[15] Rodney Brooks. The seven deadly sins of al predictions, 2017.

[16] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[17] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. springer, 2009.

[18] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019.

[19] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[20] Andres Campero, Andrew Francl, and J. B Tenenbaum. Learning to learn visual object categories by integrating deep learning with hierarchical bayes. *CogSci*, 2017.

[21] Andres Campero, Aldo Pareja, Tim Klinger, Josh Tenenbaum, and Sebastian Riedel. Logical rule induction and theory learning using neural theorem proving. *arXiv preprint arXiv:1809.02193*, 2018.

[22] Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B. Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals, 2020.

[23] S. Carey. *The origin of concepts*. Oxford University Press, 2009.

[24] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for OpenAI gym. `https://github.com/maximecb/gym-minigrid`, 2018.

[25] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

[26] William W. Cohen. Tensorlog: A differentiable deductive database. *CoRR*, abs/1605.06523, 2016.

[27] A.M. Collins and M.R. Quillian. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 8:240–248, 1969.

[28] A. Cropper and S. Muggleton. Learning higher-order logic programs through abstraction and invention. In *IJCAI*, pages 1418–1424, 2016.

[29] L. De Raedt and K Kersting. Probabilistic inductive logic programming. In *Probabilistic Inductive Logic Programming*, pages 1–27. Springer, 2008.

[30] Luc De Raedt. Logical settings for concept-learning. *Artif. Intell.*, 95(1):187–201, August 1997.

[31] I. Donadello, L. Serafini, and AS. D'Avila Garcez. Logic tensor networks for semantic image interpretation. In *IJCAI*, pages 1596–1602, 2017.

[32] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. *ArXiv*, abs/1604.06778, 2016.

[33] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B. Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning, 2020.

[34] DC Engelbert. A conceptual framework for the augumentation of man's intellect. *Vistas in information handling*, 1963.

[35] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. IMPALA: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

[36] R. Evans and E. Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

[37] Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 12623–12634, 2019.

[38] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[39] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*, 2017.

[40] John Foley, Emma Tosch, Kaleigh Clary, and David Jensen. Toybox: Better atari environments for testing reinforcement learning agents. *arXiv preprint arXiv:1812.02850*, 2018.

[41] Santo Fortunato, Carl T Bergstrom, Katy Börner, James A Evans, Dirk Helbing, Staša Milojević, Alexander M Petersen, Filippo Radicchi, Roberta Sinatra, Brian Uzzi, et al. Science of science. *Science*, 359(6379):eaao0185, 2018.

[42] Artur d'Avila Garcez, Marco Gori, Luis C Lamb, Luciano Serafini, Michael Spranger, and Son N Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *arXiv preprint arXiv:1905.06088*, 2019.

[43] A. Gaunt, M. Brockschmidt, R. Singh, N. Kushman, P. Kohli, J. Taylor, and D. Tarlow. Terpret: A probabilistic programming language for program induction. *arXiv preprint arXiv:1608.04428*, 2016.

[44] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[45] Noah D Goodman, Joshua B Tenenbaum, and Tobias Gerstenberg. Concepts in a probabilistic language of thought. Technical report, Center for Brains, Minds and Machines (CBMM), 2014.

[46] Thomas Griffiths and Joshua B Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16:17, 2004.

[47] Matthew Groh, Ziv Epstein, Chaz Firestone, and Rosalind Picard. Deepfake detection by human crowds, machines, and machine-informed crowds. *Proceedings of the National Academy of Sciences*, 119(1), 2022.

[48] Barbara Grosz. What question would turing pose today? *AI magazine*, 33(4):73–73, 2012.

[49] G. Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, 1986.

[50] Jeremy Howard. Is github copilot a blessing, or a curse? *Fast.ai*, 2021.

[51] David Kaiser. Booms, busts, and the world of ideas: Enrollment pressures and the challenge of specialization. *Osiris*, 27(1):276–302, 2012.

[52] Antonis C Kakas, Robert A. Kowalski, and Francesca Toni. Abductive logic programming. *Journal of logic and computation*, 2(6):719–770, 1992.

[53] Y. Katz, N. Goodman, K. Kersting, C. Kemp, and J. Tenenbaum. Modeling semantic cognition as logical dimensionality reduction. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2008.

[54] Seyed Mehran Kazemi and David Poole. Relnn: a deep neural model for relational learning. *arXiv preprint arXiv:1712.02831*, 2017.

[55] Charles Kemp, Amy Perfors, and Joshua B Tenenbaum. Learning overhypotheses with hierarchical bayesian models. *Developmental science*, 10(3):307–321, 2007.

[56] Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135. IEEE, 2005.

[57] Bernard Koch, Daniele Silvestro, and Jacob G Foster. The evolutionary dynamics of cultural change (as told through the birth and brutal, blackened death of metal music). *SocArXiv*, 2020.

[58] Pushmeet Kohli, Toby Sharp, Tom Minka, John Winn, Jamie Shotton, and Antonio Criminisi. Microsoft research in cambridge image dataset, 2005.

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[60] Heinrich Küttler, Nantas Nardelli, Thibaut Lavril, Marco Selvatici, Viswanath Sivakumar, Tim Rocktäschel, and Edward Grefenstette. TorchBeast: A PyTorch platform for distributed RL. *arXiv preprint arXiv:1910.03552*, 2019.

[61] Heinrich Küttler, Nantas Nardelli, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The NetHack Learning Environment. In *Workshop on Beyond Tabula Rasa in Reinforcement Learning (BeTR-RL)*, 2020.

[62] Nicolas Lair, Cédric Colas, Rémy Portelas, Jean-Michel Dussoux, Peter Ford Dominey, and Pierre-Yves Oudeyer. Language grounding through social interactions and curiosity-driven multi-goal learning. *arXiv preprint arXiv:1911.03219*, 2019.

[63] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

[64] James R Larson. *In search of synergy in small group performance*. Psychology Press, 2010.

[65] S. Laurence and E. Margolis. Concepts and cognitive science. *Concepts: core readings*, pages 3–81, 1999.

[66] Joseph CR Licklider. Man-computer symbiosis. *IRE transactions on human factors in electronics*, (1):4–11, 1960.

[67] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[68] Computing Machinery. Computing machinery and intelligence-am turing. *Mind*, 59(236):433, 1950.

[69] Thomas Malone, Daniela Rus, and Robert Laubacher. Artificial intelligence and the future of work. *MIT Work of the Future RB17-2020*, 2020.

[70] Thomas W Malone. *Superminds: The surprising power of people and computers thinking together*. Little, Brown Spark, 2018.

[71] Thomas W Malone, Kevin Crowston, and George Arthur Herman. *Organizing business knowledge: The MIT process handbook*. MIT press, 2003.

[72] R. Manhaeve, S. Duman, A. Kimmig, T. Demeester, and L. De Raedt. Deepproblog: Neural probabilistic logic programming. *arXiv preprint arXiv:1805.10872*, 2018.

[73] David Marr. Vision: A computational investigation into the human representation and processing of visual information. mit press. *Cambridge, Massachusetts*, 1982.

[74] Fernando Martinez-Plumed, Pablo Barredo, Sean O Heigeartaigh, and Jose Hernandez-Orallo. Research community dynamics behind popular ai benchmarks. *Nature Machine Intelligence*, 3(7):581–589, 2021.

[75] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *IEEE transactions on neural networks and learning systems*, 2017.

[76] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[77] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[78] Matej Moravcík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. DeepStack: Expert-level artificial intelligence in no-limit poker. *ArXiv*, abs/1701.01724, 2017.

[79] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint arXiv:2003.04960*, 2020.

[80] Allen Newell. You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. 1973.

[81] Michael Nielsen. *Reinventing discovery*. Princeton University Press, 2020.

[82] Georg Ostrovski, Marc G Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2721–2730. JMLR. org, 2017.

[83] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[84] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[85] Deepak Pathak, Pulkit Agrawal, Alexei Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[86] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

[87] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pages 835–853. PMLR, 2020.

[88] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

[89] J. R. Quinlan and R. M. Cameron-Jones. Induction of logic programs: FOIL and related systems. *New Generation Computing*, 13(3,4):287–312, 1995.

[90] Sebastien Racaniere, Andrew K Lampinen, Adam Santoro, David P Reichert, Vlad Firoiu, and Timothy P Lillicrap. Automated curricula through setter-solver interactions. *arXiv preprint arXiv:1909.12892*, 2019.

[91] Roberta Raileanu and Tim Rocktäschel. {RIDE}: Rewarding impact-driven exploration for procedurally-generated environments. In *International Conference on Learning Representations*, 2020.

[92] Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pages 6550–6561, 2017.

[93] Jürgen Renn. *The Evolution of Knowledge: Rethinking Science for the Anthropocene*. Princeton University Press, 2020.

[94] Sebastian Risi and Julian Togelius. Procedural content generation: From automatically generating game levels to increasing generality in machine learning. *arXiv preprint arXiv:1911.13071*, 2019.

[95] T. Rocktäschel and S. Riedel. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems*, pages 3791–3803, 2017.

[96] T. Rogers and J. McClelland. *Semantic cognition: A parallel distributed processing approach*. MIT Press, Cambridge, MA, 2004.

[97] Anselm Rothe, Alexander S Rich, and Zhi-Wei Li. Topics and trends in cognitive science (2000–2017). In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 1175–1180. Cognitive Science Society Austin, TX, 2018.

[98] Christopher Rouff and Mike Hinchey. *Experience from the DARPA urban challenge*. Springer Publishing Company, Incorporated, 2011.

[99] Ruslan Salakhutdinov, Joshua B Tenenbaum, and Antonio Torralba. One-shot learning with a hierarchical nonparametric bayesian model. In *ICML Unsupervised and Transfer Learning*, pages 195–206, 2012.

[100] Ruslan Salakhutdinov, Joshua B Tenenbaum, and Antonio Torralba. Learning with hierarchical-deep models. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1958–1971, 2013.

[101] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

[102] Jürgen Schmidhuber. Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. In *Front. Psychol.*, 2011.

[103] Lauren A Schmidt. *Meaning and compositionality as statistical induction of categories and constraints*. PhD thesis, Citeseer, 2009.

[104] Laura Schulz. Finding new facts; thinking new thoughts. In *Advances in child development and behavior*, volume 43, pages 269–294. Elsevier, 2012.

[105] D. Selsam. The terpret problem and the limits of sgd. *On Machine Intelligence, https://dselsam.github.io/the-terpret-problem/*, 2018.

[106] L. Serafini and A. Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*, 2016.

[107] Yash Sherry and Neil C. Thompson. How fast do algorithms improve? [point of view]. *Proceedings of the IEEE*, 109(11):1768–1777, 2021.

[108] Divya Siddarth, Daron Acemoglu, Danielle Allen, Kate Crawford, James Evans, Michael Jordan, and Glen Weyl. The turing test is bad for business. *Harvard, Justice, Health and Democracy Impact Initiative*, 2021.

[109] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[110] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[111] Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pages 2601–2606. Cognitive Science Society, 2009.

[112] Linda B Smith, Susan S Jones, Barbara Landau, Lisa Gershkoff-Stowe, and Larissa Samuelson. Object name learning provides on-the-job training for attention. *Psychological Science*, 13(1):13–19, 2002.

[113] Jonathan Sorg, Satinder P Singh, and Richard L Lewis. Internal rewards mitigate agent boundedness. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 1007–1014, 2010.

[114] G. Sourek, V. Aschenbrenner, F. Zelezny, and O. Kuzelka. Lifted relational neural networks. *arXiv preprint arXiv:1508.05128*, 2015.

[115] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[116] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[117] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

[118] S. Tran and AS. D'Avila Garcez. Deep logic networks: Inserting and extracting knowledge from deep belief networks. *IEEE transactions on neural networks and learning systems*, 2016.

[119] Philipp Tschandl, Christoph Rinner, Zoe Apalla, Giuseppe Argenziano, Noel Codella, Allan Halpern, Monika Janda, Aimilios Lallas, Caterina Longo, Josep Malvehy, et al. Human–computer collaboration for skin cancer recognition. *Nature Medicine*, 26(8):1229–1234, 2020.

[120] Pedro A Tsividis, Thomas Pouncy, Jaqueline L Xu, Joshua B Tenenbaum, and Samuel J Gershman. Human learning in atari. In *2017 AAAI Spring Symposium Series*, 2017.

[121] T. Ullman, N. Goodman, and J. Tenenbaum. Theory learning as stochastic search in the language of thought. *Cognitive Development*, 27(4):455–480, 2012.

[122] Michelle Vaccaro and Jim Waldo. The effects of mixing machine learning and human judgment. *Communications of the ACM*, 62(11):104–110, 2019.

[123] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[124] W. Wang, K. Mazaitis, and W. Cohen. A soft version of predicate invention based on structured sparsity. In *IJCAI*, pages 3918–3924, 2015.

[125] Fei Xu and Joshua B Tenenbaum. Word learning as bayesian inference. *Psychological review*, 114(2):245, 2007.

[126] F. Yang, Z. Yang, and W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pages 2316–2325, 2017.

[127] K. Yarden, N. Goodman, K. Kersting, C. Kemp, and J. Tenenbaum. Modeling semantic cognition as logical dimensionality reduction. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2008.

[128] Chang Ye, Ahmed Khalifa, Philip Bontrager, and Julian Togelius. Rotation, translation, and cropping for zero-shot generalization. *CoRR*, abs/2001.09908, 2020.

[129] Ilker Yildirim. Bayesian inference: Metropolis-hastings sampling. *University of Rochester, NY*, 2012.

[130] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

[131] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement, 2020.

[132] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. In *Advances in Neural Information Processing Systems*, pages 4644–4654, 2018.

[133] Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. RTFM: generalising to new environment dynamics via reading. In *International Conference on Learning Representations*, 2020.