Harmonizing Program Induction with Rate-Distortion Theory

Hanqi Zhou[1,2,3,4]**(hanqi.zhou@uni-tuebingen.de), David G. Nagy**[1,2,4] **& Charley M. Wu**[1,2,3,4]
[1] Human and Machine Cognition Lab, University of Tübingen, Tübingen, Germany
[2] Max Planck Institute for Biological Cybernetics [3] IMPRS-IS [4] Tübingen AI Center

Abstract

Many aspects of human learning have been proposed as a process of constructing mental programs: from acquiring symbolic number representations to intuitive theories about the world. In parallel, there is a long-tradition of using information processing to model human cognition through Rate Distortion Theory (RDT). Yet, it is still poorly understood how to apply RDT when mental representations take the form of programs. In this work, we adapt RDT by proposing a three way trade-off among rate (description length), distortion (error), and computational costs (search budget). We use simulations on a melody task to study the implications of this trade-off, and show that constructing a shared program library across tasks provides global benefits. However, this comes at the cost of sensitivity to curricula, which is also characteristic of human learners. Finally, we use methods from partial information decomposition to generate training curricula that induce more effective libraries and better generalization.

Keywords: program induction; curriculum learning; compression; compositional; resource rationality

Introduction

Human learning is not just the accumulation of knowledge, but can be seen as a process of distilling experience into compact, generalizable programs (Rule, Tenenbaum, & Piantadosi, 2020). This process requires balancing the richness of our mental representations with the cognitive cost of maintaining and processing them (Lieder & Griffiths, 2020). For example, when learning music, people typically start by learning to reproduce specific melodies verbatim, through which we develop a general understanding of musical structures. Simultaneously, we need to develop compact representations, with seasoned musicians being able to utilize regularities such as key or scale to more efficiently encode melodies without needing to memorize each note. These dual aspects of program learning and compression are crucial for efficient recognition, recall, and creation of new melodies.

Program induction is the process of inferring rules or instructions that generate an observed pattern of data (Ellis et al., 2021; Rule et al., 2020). Emerging from early computational theories (Gold, 1967; Solomonoff, 1964) formalizing learning as an algorithmic process of inferring the program that generated the observed data, it has since been refined by cognitive scientists using symbolic reasoning and logical programming as a framework to model human learning and thinking (Fodor, 1975). This approach offers a model for understanding how humans, or machines, learn underlying rules, structures, or more broadly, generative program-like representations from observed examples.

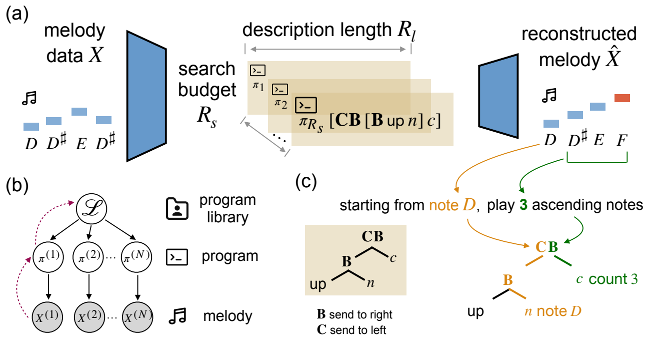

Figure 1: (a) Program induction under resource constraints using an encoder-decoder framework on melody data. The encoder compresses melodies (piano rolls X) onto a latent space (programs π), which are constrained by two bottlenecks: description length R_l and search budget R_s. (b) Each melody $X^{(i)}$ is assumed to be generated by program $\pi^{(i)}$, which is defined by a program library $\mathcal{L}$ (solid arrows). When inferring the program and library given observed melodies (dashed arrows), the goal is to find a balance between compact and easy-to-search programs, while minimizing reconstruction error $d(X, \hat{X})$. (c) Illustrative example of the tree structure and routers used in a program.

Rate distortion theory (RDT; Berger, 1971; Shannon, 1959) provides a normative framework for how to allocate limited resources to encode information in settings where perfect (i.e., lossless) compression is not possible. Since humans have limited cognitive resources, RDT has provided a powerful framework for modeling how we develop informative and structured mental representations, selectively ignoring irrelevant or redundant information to most effectively encode the most useful and relevant information. Recently, RDT has been applied to a wide number of cognitive phenomena, such as memory (Bates & Jacobs, 2020; Gershman, 2021; Nagy, Török, & Orbán, 2020), perception (Bates, Lerch, Sims, & Jacobs, 2019; Sims, 2016), and decision-making (Bhui & Gershman, 2018; Lai & Gershman, 2021).

However, it is not well understood how to apply the RDT framework when mental representations take the form of programs, and what limitations might arise. In particular, program induction implies path dependencies arising from the iterative construction of a library of programs, and there are open questions concerning how the curriculum order of learning materials influences the ease of generalizing to new data.

Goal and scope

Here, we model program induction under resource constraints with RDT in the domain of learning musical melodies. Through simulations, we show that two different dimensions of compression are relevant to learning using adaptor grammars (AGs; Johnson, Griffiths, & Goldwater, 2006). On the one hand and consistent with past algorithmic RDT approaches, there is a bias towards learning simpler programs Sablé-Meyer, Ellis, Tenenbaum, and Dehaene (2022); Sow and Eleftheriadis (2003); Vereshchagin and Vitanyi (2005). On the other hand, the amount of cognitive resources available to search for new programs adds a new dimension that is not accounted for in RDT. Together, these two dimensions influence path-dependent learning, where we show that building a more synergistic library of programs with greater overlap between stimuli allows for better generalization. Altogether, this work provides the first steps towards integrating program induction with RDT, showing new dimensions of relevance and providing guidance for future work applying this framework for modeling human learners.

Methods

In this paper, we study the interaction between compression and program learning using simulations on a melody learning task (Fig. 1). The model is presented with successive notes from a melody, which needs to be encoded to minimize distortion (i.e., error), while subject to limitations on the description length of the program (i.e., memory) and resources available for searching over programs (i.e., computational costs).

Melody learning task

Music is a universal element of human culture, transcending linguistic and cultural boundaries (Jacoby et al., 2024; Loui, Wessel, & Kam, 2010; Mehr et al., 2019). The structured and inventive nature of music makes it perfect for studying a compositional domain where simpler elements (notes, rhythms) can be combined into more complex structures (melodies, harmonies). Our dataset comes from real-world melodies in the form of monophonic piano rolls (Fig. 1a; Garcia-Valencia, Betancourt, & Lalinde-Pulido, 2020).

We first pre-process the piano rolls by 1) removing melodies with fewer than 10 notes, 2) normalizing notes into a single-octave range with 12 notes $[C, D, E, F, G, A, B, C, C\#, D\#, F\#, G\#, A\#]$, and 3) denoting pauses with a special symbol to exclude them from arithmetic operations. After pre-processing, we randomly sampled 1,000 melodies for a training set and 500 melodies for an evaluation set, with mean lengths of 50 notes in each. We then train probabilistic encoder-decoder models (Fig. 1a) to compress melodies using program-like representations and test how they generalize to new melodies. The model encodes each melody onto a latent space, by searching over discrete programs, subject to constraints on memory and computational costs. We first introduce a general Bayesian program induction framework and provide details about these constraints.

Bayesian program induction

Bayesian program induction (Correa et al., 2023; Goodman, Tenenbaum, Feldman, & Griffiths, 2008; Lake, Salakhutdinov, & Tenenbaum, 2015; Piantadosi, Tenenbaum, & Goodman, 2012; Rule, Schulz, Piantadosi, & Tenenbaum, 2018) provides a modern interpretation of Fodor's 1975 *Language of Thought* (LoT), allowing for both probabilistic representations of stimuli and the ability to simulate new scenarios given a set of learned programs. In our melody task, the goal is to find a program π that adequately represents each melody $X^{(i)} \in \{X^{(1)}, \ldots X^{(N)}\}$, described by a sequence of T notes $X^{(i)} = [x_1^{(i)}, \ldots, x_T^{(i)}]$. By specifying a prior probability distribution over all programs $p(\pi)$, and the likelihood of the observed melody under each program $p(X \mid \pi)$, Bayes' theorem can be used to compute the posterior probability of the encoding program conditioned on the observed melody:

$$p(\pi \mid X) \propto p(X \mid \pi) p(\pi) \qquad (1)$$

Here, we use *combinatory logic* (CL; Liang, Jordan, & Klein, 2010; Schönfinkel, 1924; Zhao, Lucas, & Bramley, 2023), which, although predating it, can be seen as a variant of $\lambda-$calculus (Church, 1936; Ellis et al., 2021; Piantadosi et al., 2012) that eliminates the need to keep track of variable names and scopes. This crucial feature helps identify shared substructures between programs (Liang et al., 2010). A program π in CL can be represented as a binary tree, where for example, the program for ascending by three semitones can be described as $\pi = [\mathbf{CB}, [\mathbf{B}, \mathtt{up}, n], c]$ (Fig. 1c). There are two arguments with different *types*, where the *router* **CB** directs the first argument n to the left subtree $[\mathbf{B}, \mathtt{up}, n]$ (orange branch in Fig. 1c), and the second argument c to the right subtree (green branch in Fig. 1c). The *primitive* `up` in the left subtree defines the operation on the provided arguments (e.g., note 2 and count 3). We now provide a brief overview of the various components; however due to space constraints, for a more comprehensive explanation see Liang et al. (2010).

Routers. Whereas CL originally required only two elementary combinators (**S** and **K**) to sufficiently instantiate any computable function, Liang et al. (2010) introduced higher-order combinators called *routers* $\mathcal{R}$ to make computations more practical. Routers $\mathcal{R}$ are finite sequences composed of combinators (first-order routers) $\{\mathbf{B}, \mathbf{C}, \mathbf{S}\}$ and direct incoming variables to the corresponding subtrees. For instance, in the generic program $[\mathcal{R}, \Gamma_l, \Gamma_r]$, the router **B** first directs the variable to the right subtree Γ_r with the results then sent to Γ_l; the router **C** directs the variable first to the left tree; and router **S** sends the variable to both trees. While abstract, these routers can compose complex functions from simpler ones, enabling infinitely productive program generation by combining a small set of primitives (base and function terms).

Primitives. Primitives, consisting of *base terms* and *function terms*, are both pre-defined. In our melody task, we define *base terms* as note n, count c, and time m by assuming learners have an initial library $\mathcal{L}$ over simple integers. *Function terms* are interpreted as functions that can take typed arguments as input. For example, primitive $\texttt{rep}(n,c)$ takes two arguments, note n and count c, representing repeating note n for c times (e.g., $\texttt{rep}([C],2) = [C,C]$). Another example, $\texttt{get}(n,m)$, takes two arguments, note n and time index m, returning the first m notes in note array n (e.g., $\texttt{get}([C,D,D^{\#}],2) = [C,D]$). Both *base terms* and *function terms* can work as inputs to other function terms, e.g., $\texttt{get}(\texttt{rep}(n,c),m)$.

Types. Typed CL assigns different types t to the domain (i.e., the set of possible inputs) and codomain (i.e., the set of possible outputs) of primitives and programs, so that the program's syntactical correctness can be ensured. *Base terms* (notes, counts, and times) are represented as distinct types t_n, t_c, t_m. This implies that, despite having identical numerical values (e.g., piano note 1, repeat 1 time, the first time index in an array), different integers have different physical meanings due to their distinct types. The type of a *function term* is denoted using a right arrow $\rightarrow$. For instance, the type for the function term $\texttt{get}(n,m)$ is expressed as $t_n, t_m \rightarrow t_n$, indicating the types of its input arguments and the type of its output.

Prior over programs. To define a prior over the tree representation of CL programs, we follow Liang et al. (2010) and introduce a *probabilistic context-free grammar* (PCFG). Intuitively, the nodes of the tree are successively expanded into subtrees according to pre-defined rules, and the complexity of the program is connected to the number of expansions, as well as the likelihoods of the terms. In order to generate samples from the PCFG while also maintaining syntactical correctness through types, we follow the generative process GENINDEP(t) in Liang et al. (2010) for each type t. When sampling primitives and routers, the prior over different primitives and routers are equiprobable conditioned on types. Thus, the prior probability of a program $\pi = [\mathrm{r}, x, y]$ given type t is

$$\log p(\pi \,|\, t) = \log p(x \,|\, t) + \log p(y \,|\, t) + \log p(\mathrm{r}). \qquad (2)$$

There are two kinds of possible expansions. The first method involves sampling a router (e.g., $\mathbf{B}$) and a primitive (e.g., $\texttt{up}(n)$) that meets the type requirement based on prior $p(f \,|\, t)$ and $p(\mathrm{r})$, resulting in a program $[\mathbf{B}, \texttt{up}, n]$. The second method recursively generates K intermediate types t_i along with their respective primitives and routers, ensuring the type sequence ultimately conforms to $t_n \rightarrow t_{i1} \rightarrow \ldots \rightarrow t_{iK} \rightarrow t_n$, thus yielding more nested programs.

Compression with programs

Here, we are interested in how program induction is shaped by constraints on cognitive resources. To formalize this question, we rely on *rate distortion theory* (RDT), which has been proposed as a normative framework for incorporating memory and processing constraints for biological agents (Bates & Jacobs, 2020; Nagy et al., 2020; Sims, 2016). RDT characterizes the balance between minimizing *distortion* D (the accuracy of the representation), while needing to keep the *rate* R (the required resource) below a certain capacity limit. Formally, RDT typically defines rate as the mutual information $I(X,Z)$ between the input X and its encoding Z, and minimizes the distortion subject to a constraint R on this quantity:

$$D(R) = \inf_Q D_Q, \;\; s.t.\, I(X,Z) \leq R, \qquad (3)$$

where Q represents the encoding function that defines the mapping $X \mapsto \hat{X}$, and $D = \mathbb{E}[d(X,\hat{X})]$ is the expected distortion between the original observation X and the reconstruction $\hat{X}$. Equivalently, the rate can be minimized such that the distortion is kept below the threshold D, and the minimal achievable rate for each constraint value defines the rate distortion function $R(D) = \inf_Q R_Q \;\; s.t.\, \mathbb{E}[d(X,\hat{X})] \leq D$.

Description length. Here, we adapt RDT for programs by measuring rate using the description length of the program (Sablé-Meyer et al., 2022; Sow & Eleftheriadis, 2003; Vereshchagin & Vitanyi, 2005), which can be seen as a generalization of Kolmogorov complexity K for imperfect reconstruction. This approach aligns with past theories suggesting human cognition has a preference for simpler representations (Chater & Vitányi, 2003; Feldman, 2000; Rubino, Hamidi, Dayan, & Wu, 2023) and allows us to define RD functions for individual observations:

$$R_l(D) = \min_Q K(\hat{X}^{(i)}) \;\; s.t.\, d(X^{(i)}, \hat{X}^{(i)}) \leq D. \qquad (4)$$

Under certain assumptions, these functions asymptotically correspond to the Shannon RD curves (Vereshchagin & Vitanyi, 2005). We assume that encodings of primitives are chosen optimally, and therefore measure the description length of a program π_j as its negative log probability according to the prior over programs $l(\pi) = -\log p(\pi)$ (Grünwald, 2007). For the distortion function, following Vereshchagin and Vitanyi (2005), we use Hamming distance, which counts how many of the recalled notes differ from the observed ones.

Search budget. Program induction, together with the optimization of the algorithmic RD objective (Eq. 4), implies a search over the space of possible programs, necessitating the use of approximations. We model constraints on this search process by defining an upper limit on how many programs can be considered, which we refer to as the search budget R_s. Extending Eq. 4 with this additional constraint provides a computationally bounded version of the RD objective:

$$D_{\text{bounded}}(R) = \inf_Q D_Q, \;\; s.t.\, K(\hat{X}^{(i)}) \leq R_l \wedge N_s \leq R_s, \qquad (5)$$

where N_s is the number of considered programs.

Approximate inference

Having introduced both Bayesian program induction and a computationally bounded version of algorithmic RDT, we now connect the two frameworks through approximate inference. Intuitively, both frameworks imply searching for programs that can compactly represent the observed melodies. In Bayesian inference, this is encouraged by the PCFG prior (Eq. 2), which attaches higher probabilities to shorter programs, whereas Eq. 4 imposes a direct limit on program length. To see how the search budget can be incorporated into the inference, we approximate the posterior using Monte Carlo (MC) methods via importance sampling. Here, proposed programs are sampled according to the prior $p(\pi)$, and then weighted by their likelihoods $p(X|\pi)$. This latter step corresponds to evaluating how likely each proposed program is to have generated observed melody, which from the RDT view corresponds to evaluating the distortion function on the reconstruction. Similar to other MC algorithms, the computational budget can be varied through the number of sampled proposals, which we identify with N_s in Eq. 5.

If the search over programs fails to find a sufficiently short candidate, we delete terms from the program in order of decreasing likelihood until the program satisfies the length constraint R_l, and then sample missing terms from the prior during reconstruction. We interpret this process as analogous to a human learner constructing an overly complex program and forgetting parts of it, resulting in characteristic, structured noise. For instance, for $\pi = [\mathbf{CB}, [\mathbf{B}, \texttt{up}, n_2], c_3]$ (Fig. 1a), the subject's reconstruction might ascend by four notes (instead of three). We introduce one additional approximation to make inference more tractable, where similar to how humans might listen to music, each melody can be segmented into subsequences represented by subprograms. Specifically, the program $\pi^{(i)}$ for each melody $X^{(i)}$ can be composed of $M^{(i)} \in \mathbb{N}^+$ subprograms $\pi^{(i)} = \{\pi_1, \ldots, \pi_M\}$. Each subprogram $\pi_j \in \pi^{(i)} \subseteq \Pi$ encodes a subsequence of the melody $X^{(i)}_{t_j:t'_j} = [x^{(i)}_j, \ldots, x^{(i)}_{j'}]$, with the length determined by the subprogram itself. We then evaluate proposals for these subprograms successively based on the likelihood.

Results

We first extend the RDT framework to a program induction setting, where we examine the constraints of description length and search budget. Then we explore how *adaptor grammars* (AGs) provide a method to enhance computational efficiency within these constraints. However, AGs are sensitive to order effects, resulting in large variations in generalization. Therefore, we develop a method to generate *a priori* curricula that lead to improved generalization through an information-theoretic definition of synergy.

To demonstrate how program induction is affected by the constraints of program length R_l and search budget R_s, we first compute the RD curve using a PCFG model for a range

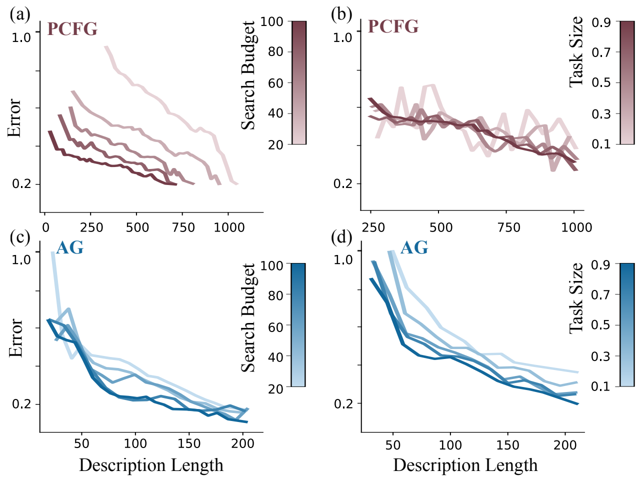

Figure 2: RD curves with (a-b) PCFGs and (c-d) AGs, given different description lengths and search budgets (left) and under different amounts of training data (right).

of settings (Fig. 2a). These curves show that lower distortion is achieved through both longer programs and larger search budgets. However, one limitation of using a PCFG is that it relies on a set of pre-specified rules and primitives, without a method for adapting them to new observations, treating each new input as a separate task. In contrast, human learning is cumulative and contextual, allowing us to build upon what we have learned from past experiences. Figure 2b illustrates how the RD curves are indifferent to different amounts of data (aside from smoothness due to random variation), with performance remaining stable (relative to different constraints).

Compression by updating the library

A key assumption of the information-theoretic notion of compression that deviates from human learning is the invariance of the compression model throughout the learning process. One way of enabling the accumulation of knowledge from past experiences is to adapt the language (i.e., compression model) to observations through the ability to define new primitives. These new primitives are composed of subprograms that have been used successfully in previous tasks and cached in a library that is shared across melodies.

For this goal, we use *Adaptor Grammars* (AGs; Johnson et al., 2006), which augment PCFGs with a capacity for building a shared library of primitives, thus breaking the strong assumptions of independence in PCFGs. Specifically, we use an AG based on the Pitman-Yor process (Teh, 2006), as defined in Liang et al. (2010). The AG model learns a distribution over grammars during training, and stores programs for future use in its cached library $\mathcal{L}^{(i)}$ given all tasks $X^{\{1:i\}}$ seen so far, updating the prior $(\pi|\mathcal{L}^{(i)})$ for future observations.

Formally, for a collection of programs C_t of type t in the library, AGs construct new programs with probability λ_1 and otherwise, it returns a cached program of type t with probability λ_2. let N_t be the number of distinct elements in C_t, and M_π be the number of times π occurs in C_t:

$$\lambda_1 = \frac{\alpha + N_t d}{\alpha + |C_t|}, \quad \lambda_2 = \frac{M_\pi - d}{|C_t| - N_t d} \tag{6}$$

Hyperparameters $\alpha > 0$ and $0 < d < 1$ govern the degree of sharing and reuse, respectively. Smaller values for α and d lead to increased sharing and reduced construction, as λ_1 is proportional to $\alpha + N_t d$. Similarly, λ_2 is proportional to M_π, indicating that more frequently cached programs receive higher weights, irrespective of their internal complexity.

Computational efficiency. We expect the augmentation of PCFGs with a library to improve the efficiency of program induction in terms of both search budget and description length. For the search process, the iteratively refined prior $p(\pi|\mathcal{L})$ serves as the proposal distribution, leading to fewer rejected samples in the MC algorithm as more data is acquired. For description length, the introduction of new primitives can facilitate more compact representations, analogously to how defining new functions that can be called later enables a programmer to reduce code length.

Compared to PFCG, AG achieves lower distortion with the same resources, as well as being less sensitive to the search budget (Fig. 2c; note the different range of the x-axes). Furthermore, as the AG model acquires more data (Fig. 2d), the new primitives added to the library facilitate even more efficient and compact representations.

To explore the intuition that an adaptive library leads to a more efficient search, we test one-shot generalization, where for a new, unobserved melody, a single program is sampled and evaluated. Figure 3a shows the results under different search budgets and exposure to other melodies from the training data. Although both models perform similarly with very little training, generalization error for PCFG is largely unaffected by more training and a greater search budget. In contrast, the AG model improves substantially with both factors.

We then analyze the degree of unique subprograms across melodies when increasing the search budget (Fig. 3b). The results show that a greater search budget increases uniqueness by developing melody-specific representations. However, greater exposure to training data decreases uniqueness, by encouraging more reuse of subprograms across different melodies. Across both patterns, AG has substantially more shared subprograms during training (50%-80%) than PCFG (20%-50%). These results suggest that the greater generalization accuracy of AG is due to developing a more domain-adapted language.

Curriculum effects

We have demonstrated that augmenting the learner with a library enables it to adapt its language to the observed data, making the AG model more efficient for both description length and search constraints. However, this comes at the cost of sensitivity to curriculum effects. Consider that the augmented learning process is performing joint inference

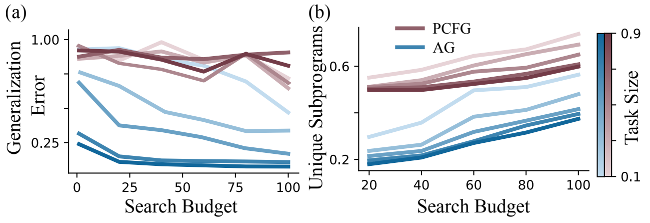

Figure 3: (a) Generalization performance given different search budgets. (b) The ratio of unique subprograms used in compressing different melodies.

over both programs and the library $p(\pi, \mathcal{L}|X^{\{1:i\}})$ given all tasks $X^{\{1:i\}}$ seen so far (Fig. 1b). Ideally, the learner would maintain a full posterior over possible libraries $p(\mathcal{L}|X^{\{1:i\}})$. However, due to the combinatorial explosion of the hypothesis space, this is not feasible. Typically the posterior is approximated with a single point estimate (or particle; Ellis et al., 2021; Zhao et al., 2023). Without access to all previously encountered tasks, this is no longer a sufficient statistic for the past. Thus, there is sensitivity to the order in which training samples are encountered (Dekker, Otto, & Summerfield, 2022; Nagy & Orban, 2016; Zhao et al., 2023).

Due to factorial growth in the number of unique stimuli orderings (i.e., curricula) as a function of the number of melodies, we ran simulations with 50 melodies across 1,000 randomly sampled curricula. Empirically, we observe high variability of learned libraries across curricula, with an average of only $3.19\% \pm 0.05\%$ shared subprograms under different curricula. To disentangle curriculum effects from the inherent stochasticity of the sampling during inference, we ran the model twice for curricula. We then compared the generalization error between matched runs of the same curriculum, compared against a random baseline with two different curricula (Fig. 4a). Matched runs were positively correlated ($r(998) = 0.286$, $p < .001$), whereas random runs were somewhat negatively correlated ($r(998) = -0.052$, $p < .001$), with the difference in performance significantly larger in matched vs. random runs ($t(999) = 7.621$, $p < .001$). These results demonstrate that the curriculum effect cannot be accounted for by stochasticity alone.

Having established that learned libraries are influenced by different curricula, we compare generalization performance between AGs' curriculum-dependent libraries against randomly generated libraries over 1,000 simulations (Fig. 4b). The random libraries consisted of randomly sampled subprograms, which are used to compress the same set of tasks. While random libraries perform worse than AG libraries on average in generalizing to new melodies, the variability of performance is quite similar, meaning unfavorable learning orders may potentially disadvantage AGs in some settings.

Synergistic curricula

While AGs have been used to explain curriculum effects in human learning (Zhao et al., 2023), previous research has typically used simple tasks with orthogonal features (Dekker et al., 2022; Rule et al., 2018), making curriculum design triv-

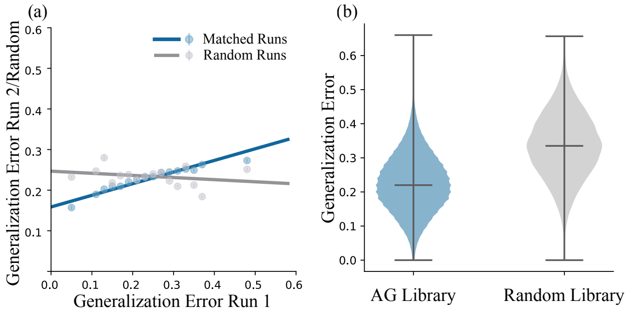

Figure 4: Curriculum effect in AGs. (a) Comparison of matched runs of the same curriculum vs. different curricula. (b) Generalization performance of AGs with a curriculum-informed library vs. a randomized library.

ially easy. Here, we explore whether we can *a priori* generate beneficial curricula for generalizing to new melodies.

We introduce a synergistic curriculum building method using principles from partial information decomposition (PID; Luppi, Rosas, Mediano, Menon, & Stamatakis, 2024; Proca et al., 2022; Williams & Beer, 2010). PID decomposes multivariate mutual information $I(\pi_1,\ldots,\pi_M;X)$ between multiple sources (i.e., the programs $\pi_m \in \mathcal{L}$) and a target variable (i.e., melodies X) into different components (Fig. 5a): *unique* $U(\pi_m;X)$ information present in exactly one program π_m, *redundant* $R(\pi_i,\ldots,\pi_j;X)$ information encoded by separately in multiple programs, and *synergistic* information $S(\pi_1,\ldots,\pi_M;X)$ jointly encoded by multiple programs. If the library $\mathcal{L}$ contains two programs, the synergy can be represented as

$$S(\mathcal{L};X) = I(\pi_1,\pi_2;X) - R(\pi_1,\pi_2;X) - U(\pi_1;X) - U(\pi_2;X) \tag{7}$$

Synergy and related quantities, such as integrated information, have been found in complex information processing (Mediano et al., 2021). We hypothesize that for both artificial and biological agents, a synergistic library is essential for being able to flexibly learn and generalize across novel problems. Here, we test whether synergy can guide the design of curricula for our AG model, with potential future applications for human learners (Zhou, Bamler, Wu, & Tejero-Cantero, 2024). Specifically, we strategically select and order melodies with the goal of maximizing synergy. Formally, given the learned library $\mathcal{L}^{(n)}$ at n-th step, we select the next presented melody X^* by maximizing the synergy of the potential library $S(\mathcal{L};X)$

$$X^* = \text{argmax}_{X^{(i)}\in X}\, S(\mathcal{L};X) \quad \text{where } \mathcal{L} \sim p(\mathcal{L}^{(n+1)} \mid \mathcal{L}^{(n)}, X^{(n)}) \tag{8}$$

The aim is to construct a sequence of training data, where each new melody is expected to add subprograms to the library that enhances the overall representational capacity. To test this method, we show that generalization performance under a synergistic curriculum is better than a random ordering ($t(999) = 6.219$, $p < .001$; Fig. 5b).

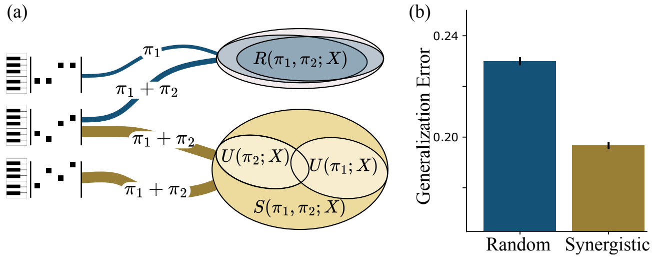

Figure 5: (a) Example libraries learned by AGs with random curricula (blue) and learned with synergistic curricula (yellow; including *R*edundant, *U*nique, and *S*ynergistic information) learned under different curricula. (b) Generalization performance given random and synergistic curricula.

Discussion

In this work, we integrated *Rate Distortion Theory* (RDT) with *Bayesian program induction* to explore how the dual constraints of description length and search complexity influence representation learning. Our simulations demonstrate that creating a shared program library across tasks facilitates more efficient learning and generalization, relative to both constraints. We also demonstrated how partial information decomposition provides an *a priori* method to design curricula that foster the development of more effective program libraries in learners, with potential future applications for human learners.

We have only begun to explore the hypothesis space in this domain, and there is still a large gap between our simulations and being able to model human learners with similar models. Nevertheless, this work presents one of the first attempts to integrate two important principles of human learning and has shed light on new patterns that were not obvious from existing theories. An important limitation that needs to be addressed for modeling human learning is that the greedy segmentation of melodies and subsequent independent generation of programs does not allow for capturing long-range structure within-melodies. This segmentation also makes targeting a specific rate difficult. While we expect that the term deletion process we introduced to enable fine-grained control over the rate might have cognitive plausibility, it may overestimate the distortion for some resource constraints. Furthermore, when defining new primitives, our model only considers the current task, and does not consider refactored versions of subprograms (Ellis et al., 2021), therefore missing chances for further sharing of components. Finally, in future experiments, we aim to explore how different design choices, such as alternative distortion functions, prior distributions, and task domains beyond melodies, would impact our results.

Overall, the integration between compression and program learning is a very promising avenue for future research, and here, we have taken the first steps in this direction.

Acknowledgments

We are grateful to Peter Dayan for helpful comments. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Hanqi Zhou. This research was supported as part of the LEAD Graduate School & Research Network, which is funded by the Ministry of Science, Research and the Arts of the state of Baden-Württemberg within the framework of the sustainability funding for the projects of the Excellence Initiative II. This work is supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–EXC2064/1–390727645, and funded by the DFG under Germany's Excellence Strategy – EXC 2117 – 422037984.

References

Bates, C. J., & Jacobs, R. A. (2020). Efficient data compression in perception and perceptual memory. *Psychological review*, *127*(5), 891.

Bates, C. J., Lerch, R. A., Sims, C. R., & Jacobs, R. A. (2019, February). Adaptive allocation of human visual working memory capacity during statistical and categorical learning. *Journal of vision*, *19*(2), 11. doi: 10.1167/19.2.11

Berger, T. (1971). *Rate-distortion theory: A mathematical basis for data compression*. Prentice-Hall.

Bhui, R., & Gershman, S. J. (2018). Decision by sampling implements efficient coding of psychoeconomic functions. *Psychological Review*, *125*(6), 985.

Chater, N., & Vitányi, P. (2003). Simplicity: a unifying principle in cognitive science? *Trends in cognitive sciences*, *7*(1), 19–22.

Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics*, *58*(2), e1008367.

Correa, C. G., Sanborn, S., Ho, M. K., Callaway, F., Daw, N. D., & Griffiths, T. L. (2023). Exploring the hierarchical structure of human plans via program generation. *arXiv preprint arXiv:2311.18644*.

Dekker, R. B., Otto, F., & Summerfield, C. (2022). Curriculum learning for human compositional generalization. *Proceedings of the National Academy of Sciences*, *119*(41), e2205582119.

Ellis, K., Wong, C., Nye, M., Sablé-Meyer, M., Morales, L., Hewitt, L., ... Tenenbaum, J. B. (2021). Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd acm sigplan international conference on programming language design and implementation* (pp. 835–850).

Feldman, J. (2000). Minimization of boolean complexity in human concept learning. *Nature*, *407*(6804), 630–633.

Fodor, I. (1975). Language reforms of the past and in the developing countries.

Garcia-Valencia, S., Betancourt, A., & Lalinde-Pulido, J. G. (2020). Sequence generation using deep recurrent networks and embeddings: A study case in music. *ArXiv e-prints*, *abs/2012.0*. Retrieved from `https://arxiv.org/abs/2012.01231`

Gershman, S. J. (2021). The rational analysis of memory. In *Oxford handbook of human memory*. Oxford, UK: Oxford University Press.

Gold, E. M. (1967). Language identification in the limit. *Information and control*, *10*(5), 447–474.

Goodman, N. D., Tenenbaum, J. B., Feldman, J., & Griffiths, T. L. (2008). A rational analysis of rule-based concept learning. *Cognitive science*, *32*(1), 108–154.

Grünwald, P. D. (2007). *The minimum description length principle*. MIT press.

Jacoby, N., Polak, R., Grahn, J. A., Cameron, D. J., Lee, K. M., Godoy, R., ... others (2024). Commonality and variation in mental representations of music revealed by a cross-cultural comparison of rhythm priors in 15 countries. *Nature Human Behaviour*, 1–32.

Johnson, M., Griffiths, T., & Goldwater, S. (2006). Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. *Advances in neural information processing systems*, *19*.

Lai, L., & Gershman, S. J. (2021). Policy compression: An information bottleneck in action selection. In *Psychology of learning and motivation* (Vol. 74, pp. 195–232). Elsevier.

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338.

Liang, P., Jordan, M. I., & Klein, D. (2010). Learning programs: A hierarchical bayesian approach. In *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 639–646).

Lieder, F., & Griffiths, T. L. (2020). Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and brain sciences*, *43*, e1.

Loui, P., Wessel, D. L., & Kam, C. L. H. (2010). Humans rapidly learn grammatical structure in a new musical scale. *Music perception*, *27*(5), 377–388.

Luppi, A. I., Rosas, F. E., Mediano, P. A., Menon, D. K., & Stamatakis, E. A. (2024). Information decomposition and the informational architecture of the brain. *Trends in Cognitive Sciences*.

Mediano, P. A., Rosas, F. E., Luppi, A. I., Carhart-Harris, R. L., Bor, D., Seth, A. K., & Barrett, A. B. (2021). Towards an extended taxonomy of information dynamics via integrated information decomposition. *arXiv preprint arXiv:2109.13186*.

Mehr, S. A., Singh, M., Knox, D., Ketter, D. M., Pickens-Jones, D., Atwood, S., ... others (2019). Universality and diversity in human song. *Science*, *366*(6468), eaax0868.

Nagy, D. G., & Orban, G. (2016). Episodic memory as a prerequisite for online updates of model structure. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society* (pp. 1–7).

Nagy, D. G., Török, B., & Orbán, G. (2020). Optimal forgetting: Semantic compression of episodic memories. *PLoS Computational Biology*, *16*(10), e1008367.

Piantadosi, S. T., Tenenbaum, J. B., & Goodman, N. D. (2012). Bootstrapping in a language of thought: A formal model of numerical concept learning. *Cognition*, *123*(2), 199–217.

Proca, A. M., Rosas, F. E., Luppi, A. I., Bor, D., Crosby, M., & Mediano, P. A. (2022). Synergistic information supports modality integration and flexible learning in neural networks solving multiple tasks. *arXiv preprint arXiv:2210.02996*.

Rubino, V., Hamidi, M., Dayan, P., & Wu, C. M. (2023). Compositionality under time pressure. In M. Goldwater, F. Anggoro, B. Hayes, & D. Ong (Eds.), *Proceedings of the 45th Annual Conference of the Cognitive Science Society.* Sydney, Australia: Cognitive Science Society. doi: 10.31234/osf.io/z2648

Rule, J. S., Schulz, E., Piantadosi, S. T., & Tenenbaum, J. B. (2018). Learning list concepts through program induction. *BioRxiv*, 321505.

Rule, J. S., Tenenbaum, J. B., & Piantadosi, S. T. (2020). The child as hacker. *Trends in cognitive sciences*, *24*(11), 900–915.

Sablé-Meyer, M., Ellis, K., Tenenbaum, J., & Dehaene, S. (2022, December). A language of thought for the mental representation of geometric shapes. *Cognitive Psychology*, *139*, 101527. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0010028522000639` doi: 10.1016/j.cogpsych.2022.101527

Schönfinkel, M. (1924, September). Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, *92*(3), 305–316. Retrieved 2024-01-31, from `https://doi.org/10.1007/BF01448013` doi: 10.1007/BF01448013

Shannon, C. E. (1959). Coding theorems for a discrete source with a fidelity criterion. *IRE Nat. Conv. Rec*, *4*(142-163), 1.

Sims, C. R. (2016). Rate–distortion theory and human perception. *Cognition*, *152*, 181–198.

Solomonoff, R. J. (1964). A formal theory of inductive inference. part i. *Information and control*, *7*(1), 1–22.

Sow, D., & Eleftheriadis, A. (2003, March). Complexity distortion theory. *IEEE Transactions on Information Theory*, *49*(3), 604–608. doi: 10.1109/TIT.2002.808135

Teh, Y. W. (2006). A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the association for computational linguistics* (pp. 985–992).

Vereshchagin, N., & Vitanyi, P. (2005). Algorithmic Rate-Distortion Theory.

Williams, P. L., & Beer, R. D. (2010). Nonnegative decomposition of multivariate information. *arXiv preprint arXiv:1004.2515*.

Zhao, B., Lucas, C. G., & Bramley, N. R. (2023). A model of conceptual bootstrapping in human cognition. *Nature Human Behaviour*, 1–12.

Zhou, H., Bamler, R., Wu, C. M., & Tejero-Cantero, (2024). Predictive, scalable and interpretable knowledge tracing on structured domains. In *The Twelfth International Conference on Learning Representations.*