www.sciencemag.org/content/350/6266/1332/suppl/DC1

Supplementary Material for

Human-level concept learning through probabilistic program induction

Brenden M. Lake,* Ruslan Salakhutdinov, Joshua B. Tenenbaum

*Corresponding author. E-mail: brenden@nyu.edu

Published 4 December 2015, *Science* **350**, 1332 (2015)
DOI: 10.1126/science.aab3050

This PDF file includes:

Human-level concept learning through probabilistic program induction – Supplemental Material

1 Omniglot dataset

This dataset contains 1623 characters spanning 50 different alphabets on www.omniglot.com. The alphabets include scripts from currently used and historic natural languages (e.g., Hebrew, Korean, Greek) as well as artificial scripts (e.g., Futurama and ULOG) invented for purposes like TV shows or video games. The alphabets were scraped from omniglot.com in printed form, and they were converted to handwritten form using human participants on Amazon Mechanical Turk (AMT). Each participant on AMT was asked to draw at least one alphabet with the option of drawing more. They were instructed to draw as accurately as possible. An alphabet's printed characters were displayed in rows on a web page with an associated drawing pad below each image. Participants could draw by holding down a mouse button and moving the mouse (or trackpad, tablet, etc.) We included "forward," "back," and "clear" buttons so drawers could easily redo their last pen stroke. The resulting data set has images paired with movies in [x, y,time] coordinates showing how the drawing was produced (and how the strokes are segmented).

2 Bayesian Program Learning

2.1 Generating character types

A character type $\psi = \{\kappa, S, R\}$ is defined by a set of κ strokes $S = \{S_1, ..., S_\kappa\}$ and spatial relations $R = \{R_1, ..., R_\kappa\}$ between strokes. The joint distribution can be written as

$$P(\psi) = P(\kappa) \prod_{i=1}^{\kappa} P(S_i) P(R_i | S_1, ..., S_{i-1}), \tag{S1}$$

and the generative process is described in Algorithm 1. The number of strokes κ is sampled from a multinomial $P(\kappa)$ estimated from the empirical frequencies (Fig. S2). All hyperparameters were learned as described in Section 2.3.

2.1.1 Strokes

Each stroke is initiated by pressing the pen down and terminated by lifting the pen up. In between, a stroke is a motor routine composed of simple movements called sub-strokes $S_i = \{s_{i1}, ..., s_{in_i}\}$ (colored curves in Fig. S1), where sub-strokes are separated by brief pauses of the pen. The number of sub-strokes n_i is sampled from the empirical frequency $P(n_i|\kappa)$ specific to the total number of strokes κ (Fig. S2), capturing the fact that characters with many strokes tend to have simpler strokes.

Given the number of sub-strokes, the process of generating a stroke is specified in Algorithm 1. Each sub-stroke s_{ij} is modeled as a uniform cubic b-spline, which can be decomposed into three variables $s_{ij} = \{z_{ij}, x_{ij}, y_{ij}\}$ with joint distribution $P(S_i) = P(z_i) \prod_{j=1}^{n_i} P(x_{ij}|z_{ij})P(y_{ij}|z_{ij})$. The discrete class $z_{ij} \in \mathbb{N}$ is an index into the library of primitives (top of Fig. S1). Its distribution $P(z_i) = P(z_{i1}) \prod_{j=2}^{n_i} P(z_{ij}|z_{i(j-1)})$ is a first-order Markov Process learned from the empirical bigrams, which can encode right angles, repetitive structure, and common motifs (Fig. S2).

The five control points $x_{ij} \in \mathbb{R}^{10}$ (small open circles in Fig. S1) are sampled from a Gaussian $P(x_{ij}|z_{ij}) = N(\mu_{z_{ij}}, \Sigma_{z_{ij}})$, and they live in an abstract space not yet embedded in the image frame. The type-level scale y_{ij} of this space, relative to the image frame, is sampled from $P(y_{ij}|z_{ij}) = \text{Gamma}(\alpha_{z_{ij}}, \beta_{z_{ij}})$. Since any possible configuration of control points can exist at the type-level, the primitive actions are more like specifications for high probability "types of parts" rather than a rigid set of parts.

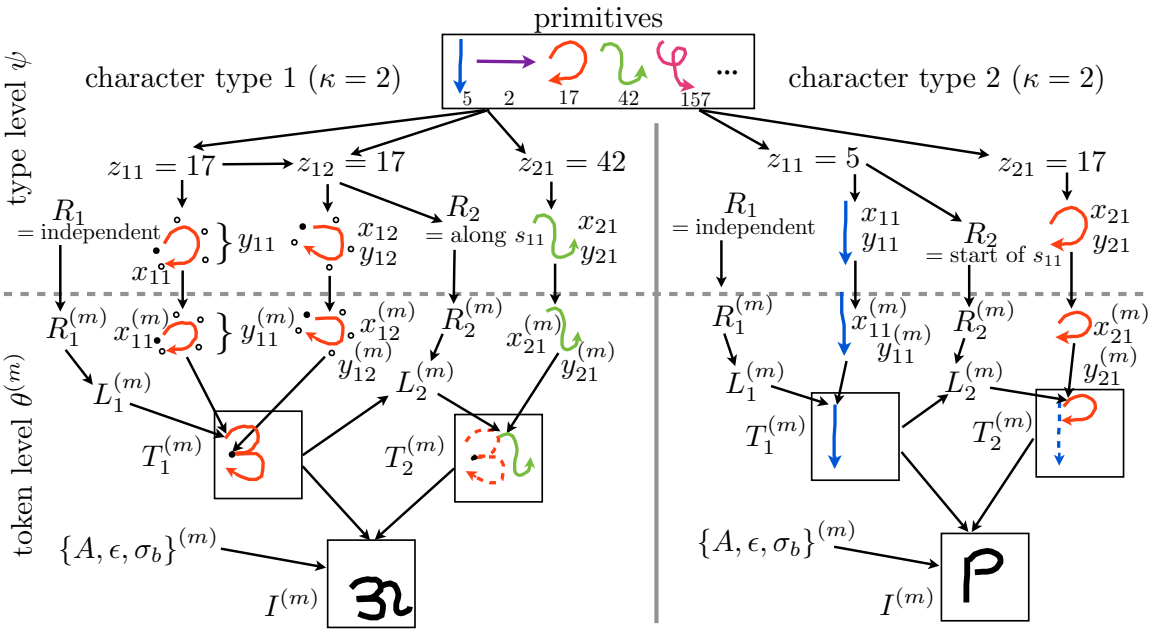

Figure S1: BPL generating two character types. Legend: number of strokes κ, relations R, primitive id z (color-coded to highlight sharing), control points x (open circles), scale y, start locations L, trajectories T, transformation A, noise ϵ and σ_b, and image I.

Algorithm 1 Generate a new character type

procedure GENERATETYPE
  $\kappa \leftarrow P(\kappa)$ ▷ Sample the number of strokes
  for $i = 1$ ... κ **do**
    $n_i \leftarrow P(n_i|\kappa)$ ▷ Sample the number of sub-strokes
    $S_i \leftarrow$ GENERATESTROKE(i, n_i) ▷ Sample stroke
    $\xi_i \leftarrow P(\xi_i)$ ▷ Sample relation to previous strokes
    $R_i \leftarrow P(R_i|\xi_i, S_1, ..., S_{i-1})$ ▷ Sample relation details
  end for
  $\psi \leftarrow \{\kappa, R, S\}$
  return @GENERATETOKEN(ψ) ▷ Return program handle
end procedure

procedure GENERATESTROKE(i,n_i)
  $z_{i1} \leftarrow P(z_{i1})$ ▷ Sample the identity of the first sub-stroke
  for $j = 2 \ldots n_i$ **do**
    $z_{ij} \leftarrow P(z_{ij}|z_{i(j-1)})$ ▷ Sample the identities of the other sub-strokes
  end for
  for $j = 1 \ldots n_i$ **do**
    $x_{ij} \leftarrow P(x_{ij}|z_{ij})$ ▷ Sample a sub-stroke's control points
    $y_{ij} \leftarrow P(y_{ij}|z_{ij})$ ▷ Sample a sub-stroke's scale
    $s_{ij} \leftarrow \{x_{ij}, y_{ij}, z_{ij}\}$
  end for
  $S_i \leftarrow \{s_{i1}, \ldots, s_{in_i}\}$ ▷ A complete stroke definition
  return S_i
end procedure

2.1.2 Relations

The spatial relation R_i specifies how the beginning of stroke S_i connects to the previous strokes $\{S_1, ..., S_{i-1}\}$, inspired by previous work on the one-shot learning of relations (*10*). Relations can come in four types, $\xi_i \in \{Independent,\ Start,\ End,\ Along\}$, with probabilities θ_R, and

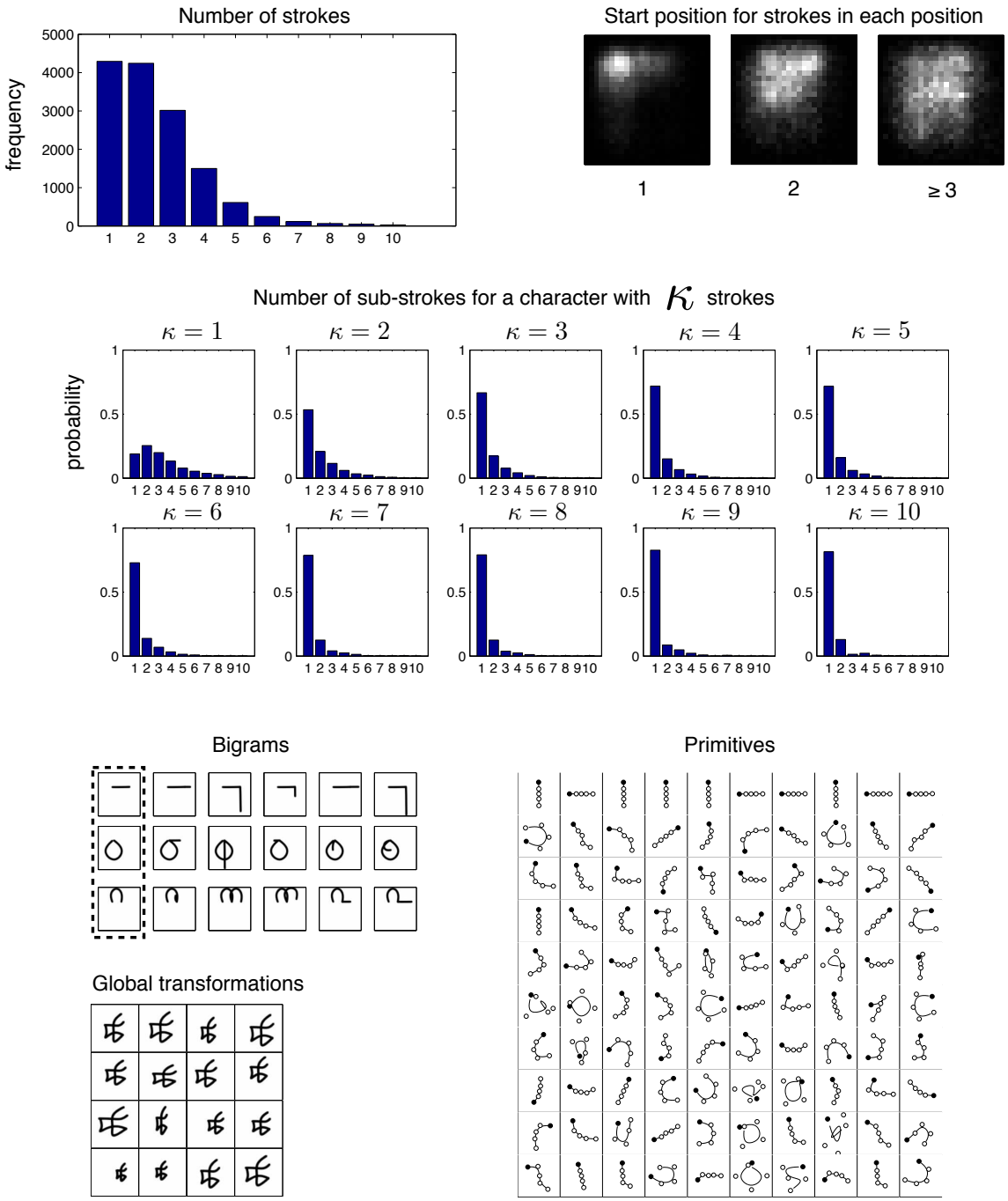

Figure S2: BPL conditional distributions. As shown, the *Start position* distribution is conditional on the stroke index in the character (label below each heatmap). The *Bigrams* illustration show seed primitive in image space (leftmost column) and then the 5 most likely continuations when sampling a second primitive. The subset of *Primitives* shows the mean of the ten most common ones in the top row followed by a random subset. The first control point (circle) is shown as filled.

each type ξ_i has different sub-variables and dimensionality (examples in Fig. S1). Here are their specifications:

- *Independent* relations, $R_i = \{\xi_i = Independent, J_i, L_i\}$, where the position of stroke i does not depend on previous strokes. The variable $J_i \in \mathbb{N}$ is drawn from $P(J_i)$, a multinomial over a 2D image grid that depends on index i (Fig. S2). Since the position $L_i \in \mathbb{R}^2$ has to be real-valued, $P(L_i|J_i)$ is then sampled uniformly at random from within the image cell J_i.
- *Start* relations, $R_i = \{\xi_i = Start, u_i\}$, where stroke i starts at the beginning of a previous stroke u_i, sampled uniformly at random from $u_i \in \{1, ..., i-1\}$.
- *End* relations, $R_i = \{\xi_i = End, u_i\}$, where stroke i starts at the end of a previous stroke u_i, sampled uniformly at random from $u_i \in \{1, ..., i-1\}$.
- *Along* relations, $R_i = \{\xi_i = Along, u_i, v_i, \tau_i\}$, where stroke i begins along previous stroke $u_i \in \{1, ..., i-1\}$ at sub-stroke $v_i \in \{1, ..., n_{u_i}\}$ at type-level spline coordinate $\tau_i \in \mathbb{R}$, each sampled uniformly at random.

2.2 Generating character tokens

The token-level variables, $\theta^{(m)} = \{L^{(m)}, x^{(m)}, y^{(m)}, R^{(m)}, A^{(m)}, \sigma_b^{(m)}, \epsilon^{(m)}\}$, are distributed as

$$P(\theta^{(m)}|\psi) = P(L^{(m)}|\theta^{(m)}_{\setminus L^{(m)}}, \psi) \prod_i P(R_i^{(m)}|R_i) P(y_i^{(m)}|y_i) P(x_i^{(m)}|x_i) P(A^{(m)}, \sigma_b^{(m)}, \epsilon^{(m)}). \tag{S2}$$

Pseudocode for sampling is provided in Algorithm 2.

Algorithm 2 Run the stochastic program of type ψ to make an image

procedure GENERATETOKEN(ψ)
  for i = 1 ... κ **do**
    $R_i^{(m)} \leftarrow R_i$ ▷ Directly copy the type-level relation
    if $\xi_i^{(m)}$ = 'along' **then**
      $\tau_i^{(m)} \leftarrow P(\tau_i^{(m)}|\tau_i)$ ▷ Add variability to the attachment along the spline
    end if
    $L_i^{(m)} \leftarrow P(L_i^{(m)}|R_i^{(m)}, T_1^{(m)}, ..., T_{i-1}^{(m)})$ ▷ Sample stroke's starting location
    for j = 1 ... n_i **do**
      $x_{ij}^{(m)} \leftarrow P(x_{ij}^{(m)}|x_{ij})$ ▷ Add variability to the control points
      $y_{ij}^{(m)} \leftarrow P(y_{ij}^{(m)}|y_{ij})$ ▷ Add variability to the sub-stroke scale
    end for
    $T_i^{(m)} \leftarrow f(L_i^{(m)}, x_i^{(m)}, y_i^{(m)})$ ▷ Compose a stroke's pen trajectory
  end for
  $A^{(m)} \leftarrow P(A^{(m)})$ ▷ Sample global image transformation
  $\epsilon^{(m)} \leftarrow P(\epsilon^{(m)})$ ▷ Sample the amount of pixel noise
  $\sigma_b^{(m)} \leftarrow P(\sigma_b^{(m)})$ ▷ Sample the amount blur
  $I^{(m)} \leftarrow P(I^{(m)}|T^{(m)}, A^{(m)}, \sigma_b^{(m)}, \epsilon^{(m)})$ ▷ Render and sample the binary image
  return $I^{(m)}$
end procedure

2.2.1 Pen trajectories

A stroke trajectory $T_i^{(m)}$ (Fig. S1) is a sequence of points in the image plane that represents the path of the pen. Each trajectory $T_i^{(m)} = f(L_i^{(m)}, x_i^{(m)}, y_i^{(m)})$ is a deterministic function of a starting location $L_i^{(m)} \in \mathbb{R}^2$, token-level control points $x_i^{(m)} \in \mathbb{R}^{10}$, and token-level scale $y_i^{(m)} \in \mathbb{R}$. The control points and scale are noisy versions of their type-level counterparts, $P(x_{ij}^{(m)}|x_{ij}) = N(x_{ij}, \sigma_x^2 I)$ and $P(y_{ij}^{(m)}|y_{ij}) \propto N(y_{ij}, \sigma_y^2)$ where the scale is truncated below 0.

To construct the trajectory $T_i^{(m)}$, the spline defined by the scaled control points $y_1^{(m)} x_1^{(m)} \in \mathbb{R}^{10}$ is evaluated to form a trajectory,[1] which is shifted in the image plane to begin at $L_i^{(m)}$. Next, the second spline $y_2^{(m)} x_2^{(m)}$ is evaluated and placed to begin at the end of the previous sub-stroke's trajectory, and so on until all sub-strokes are placed.

Token-level relations must be exactly equal to their type-level counterparts, $P(R_i^{(m)}|R_i) = \delta(R_i^{(m)} - R_i)$, except for the "along" relation which allows for token-level variability for the attachment along the spline using Gaussian $P(\tau_i^{(m)}|\tau_i) \propto N(\tau_i, \sigma_\tau^2)$ truncated at the ends of the spline. Given the pen trajectories of the previous strokes, the start position of $L_i^{(m)}$ is sampled from $P(L_i^{(m)}|R_i^{(m)}, T_1^{(m)}, ..., T_{i-1}^{(m)}) = N(g(R_i^{(m)}, T_1^{(m)}, ..., T_{i-1}^{(m)}), \Sigma_L)$. The function $g(\cdot)$ locates the stroke at an appropriate position depending on the relation. For *Independent* relations, $g(\cdot) = L_i$ where L_i is the type-level global location. For *End* relations (and analogously *Start*), $g(\cdot) = \text{end}(T_{u_i}^{(m)})$ where $\text{end}(\cdot)$ selects the last element in a sequence of trajectory points. For *Along* relations, $g(\cdot) = \text{spline-eval}(T_{u_i}^{(m)}, v_i, \tau_i)$ is the evaluation at position τ_i along the spline that defines the v_ith sub-stroke segment that makes up the stroke trajectory $T_{u_i}^{(m)}$.

2.2.2 Image

An image transformation $A^{(m)} \in R^4$ is sampled from $P(A^{(m)}) = N([1, 1, 0, 0], \Sigma_A)$, where the first two elements control a global re-scaling and the second two control a global translation of

[1] The number of spline evaluations is computed to be approximately 2 points for every 3 pixels of distance along the spline (with a minimum of 10 evaluations).

the center of mass of $T^{(m)}$. Samples from this distribution are shown in Fig. S2.

After applying $A^{(m)}$, grayscale ink is placed along the trajectories using a ink model similar to (*35*). Each point on the trajectory $T^{(m)}$ contributes up to two units of ink to the four closest pixels using bilinear interpolation, where the ink units decrease linearly from 1 to 0 if two points are less than two pixel units apart. This method creates a thin line of ink, which is expanded out by convolving the image twice with the filter $b[\frac{a}{12}, \frac{a}{6}, \frac{a}{12}; \frac{a}{6}, 1-a, \frac{a}{6}; \frac{a}{12}, \frac{a}{6}, \frac{a}{12}]$ and thresholding values greater than 1 ($a = 0.5$ and $b = 6$).

This grayscale image is then perturbed by two noise processes, making the gradient more robust during optimization and encouraging partial solutions during classification. The first noise process blurs the model's rendering of a character. Blurring is accomplished through a convolution with a Gaussian filter with standard deviation $\sigma_b^{(m)}$. The amount of noise $\sigma_b^{(m)}$ is itself a random variable, sampled from a uniform distribution on a pre-specified range,[2] allowing for the model to adaptively adjust the fidelity of its fit to an image (*30*). The second noise process stochastically flips pixels with probability $\epsilon^{(m)}$, such that the overall probability of inking a binary pixel is a Bernoulli with probability $P(I_{ij} = 1) = (1 - \epsilon^{(m)})\rho_{ij} + \epsilon^{(m)}(1 - \rho_{ij})$. The amount of noise is also a random variable that can be adaptively set during inference.[3] The grayscale pixels then parameterize 105x105 independent Bernoulli distributions, completing the full model of binary images $P(I^{(m)}|\theta^{(m)}) = P(I^{(m)}|T^{(m)}, A^{(m)}, \sigma_b^{(m)}, \epsilon^{(m)})$.

2.3 Learning-to-learn motor programs

Omniglot was randomly split into a 30 alphabet "background" set and a 20 alphabet "evaluation" set constrained such that the background set included the six most common alphabets as determined by Google hits, including Latin, Greek, Japanese, Korean, Hebrew, and Tagalog.

[2]The distribution on amount of blur is $\sigma_b^{(m)} \sim \text{uniform}(0.5, 16)$. The probability map is blurred by two convolutions with a Gaussian filter of size 11 with standard deviation $\sigma_b^{(m)}$.

[3]The distribution on the amount of pixel noise is $\epsilon^{(m)} \sim \text{uniform}(.0001, 0.5)$.

Background images paired with their motor data was used to learn the BPL hyperparameters as shown in Fig. S2 and described below.

To further examine the role of learning-to-learn, BPL was also trained on a smaller set of just 5 "background" alphabets. Two different sets of 5 were chosen, where set 1 included Latin, Greek, Korean, Early Aramaic, and Balinese and set 2 included Latin, Greek, Japanese (katakana), Sanskrit, and Tagalog.

2.3.1 Learning primitives

The learned primitive actions serve multiple roles. First, they encourage common shapes found in the background set. Second, since the primitives are directed trajectories, they implement direction preferences such as a general top-down and left-right drawing preference (*63,64*). The learned actions are position invariant. They are also encouraged to be scale-selective, where each primitive prefers a small range of sizes if supported by the background data. This is necessary for capturing interesting sequential structure like primitive repetition where the same movement (at the same scale) occurs multiple times (Fig. S2 Bigrams).

In the Omniglot dataset, the sub-stroke trajectories have non-uniform spatial and temporal sampling intervals, due to the fact that it is a synthesis of drawing data from many different web browsers and computers. To standardize the data, first, all pen trajectories were normalized in time to have a 50 millisecond sampling interval as approximated by linear interpolation. If the pen moved less than one pixel between two points, it was marked as a pause. Sub-strokes were defined as the segments extracted between pairs of pauses.

After the sub-strokes were extracted, all trajectories were normalized to have a uniform spatial resolution with one pixel distance between points, so that only the shape of the trajectory was relevant for clustering. Furthermore, sub-stroke trajectory was normalized to have zero mean and a common scale along its longest dimension. Sub-stroke trajectories with less than

10 points were removed. This resulted in about 55,000 sub-stroke trajectories. Each sub-stroke was fit with a spline and re-represented by its five control points in $\mathbb{R}^{10}$. To achieve partial scale invariance, scale was included as an additional dimension but weighted as two dimensions. A diagonal Gaussian Mixture Model (GMM) fit with expectation maximization was used to partition sub-strokes into 1250 primitive elements, removing small mixture components (350 primitives were used when there were 5 background alphabets). Given this partition, the parameters for each primitive z, μ_z, Σ_z, α_z and β_z, could be fit with maximum likelihood estimation (MLE). The transition probabilities between primitives $P(z_{ij}|z_{i(j-1)})$ were estimated by the smoothed empirical counts, after the sub-strokes were assigned to the most likely primitive, where the regularization was chosen via cross-validation by withholding 25 percent of the characters in the background set as validation data.

2.3.2 Learning start positions

The model for stroke start positions $P(L_i)$ (Section 2.1) was estimated by discretizing the image plane and fitting a separate multinomial grid model for a drawing's first and second stroke. All additional strokes share a single aggregated model (Fig. S2). The probability of each cell was estimated from the empirical frequencies, and the complexity parameters for the grid granularity, smoothing, and aggregation threshold were also chosen by cross-validation on withheld background set characters. Evidently, position is concentrated in the top-left where the concentration is stronger for earlier strokes.

2.3.3 Learning relations and token variability

The hyperparameters governing relations and token variability are less straightforward to estimate, since they cannot be directly computed from the motor data. Instead, we fit a large number of motor programs to 800 images in the background set using temporary values of these parameters. After the programs were fit to these real background characters, the values

were re-estimated based on this set of programs. The temporary hyperparameter values were chosen based on an earlier model that used a more complicated fitting procedure (*65*). For the BPL models fit to 5 alphabets, only a subset of the 800 images corresponding to those alphabets were used.

Relational parameters, including mixing probabilities θ_R and position noise Σ_L, were estimated by assuming a flat prior on relation types and examining about 1600 motor programs fit to the background set. Based on the fit statistics, the mixing proportions were estimated as 34% independent, 5% start, 11% end, and 50% along.

The token-level variability parameters for shape σ_x, scale σ_y, and attachment σ_τ govern how much the pen trajectories change from exemplar to exemplar. Thus, the 1600 motor programs were re-fit to new examples of those characters using the procedure described in Section 5.2. The variance of the relevant exemplar statics could be computed by comparing the values in the parses fit to the original image versus the new image. The token-level variance in the substroke control points, scales, and attachment positions were estimated as the expectation of the squared deviation from the mean of each pair, after removing outliers and letters from the Braille alphabet.

2.3.4 Learning image parameters

The distribution on transformations $P(A^{(m)})$ was also learned, and samples are shown in Fig. S2. For each image, the center of mass and range of the inked pixels was computed. Second, images were grouped by character, and a transformation (scaling and translation) was computed for each image so that its mean and range matched the group average. Based on this large set of approximate transformations, a covariance on transformations Σ_A could be estimated. The ink model hyperparameters $a = 0.5$ and $b = 6$ were fit with maximum likelihood given a small subset of background pairs of images and drawings. The ink model parameters were not

re-learned for the 5 alphabet training condition.

3 Inference for discovering motor programs from images

Posterior inference in this model is challenging since parsing an image $I^{(m)}$ requires exploring a large combinatorial space of different numbers and types of parts, sub-parts, and relations. In principle, generic MCMC algorithms such as the one explored in (*66*) can be used, but we have found this approach to be slow, prone to local minima, and poor at switching between different parses. Instead, inspired by the speed of human perception and approaches for faster inference in probabilistic programs (*67*), we explored bottom-up methods to compute a fast structural analysis and propose values of the latent variables in BPL. This produces a large set of possible motor programs – each approximately fit to the image of interest. The most promising motor programs are chosen and refined with continuous optimization and MCMC. The end product is a list of K high-probability parses, $\psi^{[1]}, \theta^{(m)[1]}, ..., \psi^{[K]}, \theta^{(m)[K]}$, which are the most promising candidates discovered by the algorithm. An example of the end product is illustrated in Fig. 4A.

3.1 Discrete approximation

Here is a summary of the basic approximation to the posterior. These parses approximate the posterior with a discrete distribution,

$$P(\psi, \theta^{(m)}|I^{(m)}) \approx \sum_{i=1}^{K} w_i \delta(\theta^{(m)} - \theta^{(m)[i]})\delta(\psi - \psi^{[i]}), \tag{S3}$$

where each weight w_i is proportional to parse score, marginalizing over type-level shape variables x and attachment positions τ and constraining $\sum_i w_i = 1$,

$$w_i \propto \tilde{w}_i = P(\psi^{[i]}_{\setminus x,\tau}, \theta^{(m)[i]}, I^{(m)}). \tag{S4}$$

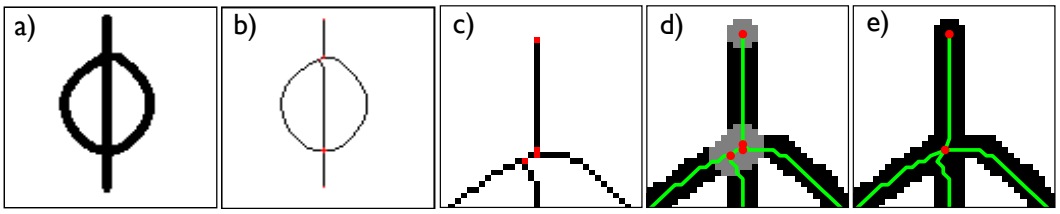

Figure S3: Extracting the character skeleton from an image. a) Original image. b) Thinned image. c) Zoom highlights the imperfect detection of critical points (red pixels). d) Maximum circle criterion applied to the spurious critical points. e) Character graph after merging.

Rather than using just a point estimate for each parse, the approximation can be improved by incorporating local variance. The token-level variables $\theta^{(m)}$ allow for little variability since they closely track the image. Also, it is relatively cheap to produce conditional samples from the type-level $P(\psi|\theta^{(m)[i]}, I^{(m)}) = P(\psi|\theta^{(m)[i]})$ as it does not require evaluating the likelihood of the image. Thus, local variance around the type-level is estimated with the token-level fixed. A Metropolis Hastings algorithm produces N samples (Section 3.6) for each parse $\theta^{(m)[i]}$, denoted by $\psi^{[i1]}, ..., \psi^{[iN]}$, where the improved approximation is

$$P(\psi, \theta^{(m)}|I^{(m)}) \approx Q(\psi, \theta^{(m)}, I^{(m)}) = \sum_{i=1}^{K} w_i \delta(\theta^{(m)} - \theta^{(m)[i]}) \frac{1}{N} \sum_{j=1}^{N} \delta(\psi - \psi^{[ij]}). \quad \text{(S5)}$$

Given an approximate posterior for a particular image, the model can evaluate the posterior predictive score of a new image by re-fitting the token-level variables (Fig. 4A), as explained in Section 5.2 on inference for one-shot classification.

3.2 Extracting the character skeleton

Search begins by applying a thinning algorithm to the raw image (Fig. S3a) that reduces the line width to one pixel (*68*) (Fig. S3b). This thinned image is used to produce candidate parses which are ultimately scored on the original image. The thinned image can provide an approximate structural analysis as an undirected graph (Fig. S3e), where edges (green) trace the ink and nodes (red) are placed at the terminal and fork (decision) points. While these decision

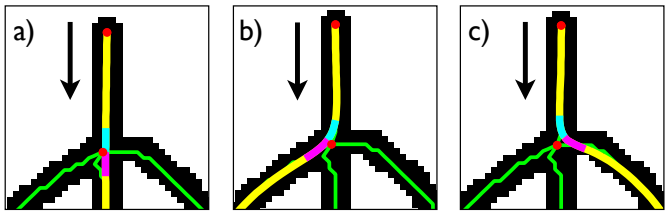

Figure S4: A random walk choosing its next move after drawing the topmost vertical edge in the direction of the arrow. There are three potential trajectory options: a) has a local angle of 0 degrees (computed between the blue and purple vectors), b) is 28 degrees, and c) is 47 degrees.

points can be detected with simple algorithms (*31*), this process is imperfect and produces too many fork points (red pixels in Fig. S3b and c). Many of these inaccuracies can be fixed by removing spurious branches and duplicate fork points with the maximum circle criterion (*69*). This algorithm places the largest possible circle on each critical point, such that the circle resides within the original ink (gray regions in Fig. S3d). All critical points with connecting circles are then merged (Fig. S3e).

3.3 Generating random parses

A candidate parse is generated by taking a random walk on the character skeleton with a "pen," visiting nodes until each edge has been traversed at least once. Since the parse space grows exponentially in the number of edges, biased random walks are necessary to explore the most interesting parts of the space for large characters. The random walker stochastically prefers actions A that minimize the local angle of the stroke trajectory around the decision point

$$P(A) \propto \exp(-\lambda\theta_A), \tag{S6}$$

where θ_A is the angle associated with action (Fig. S4) and λ is a constant. Two other possible actions, picking up the pen and re-tracing a trajectory, pay a cost of 45 and 90 degrees respectively. If the pen is in lifted position, the random walk must pick a node to place the pen, chosen in proportion to $1/b^\gamma$ where b is the number of new (unvisited) edges branching from

that node. This random walk process is repeated many times to generate a range of candidate parses. Random walks are generated until 150 parses or 100 unique strokes, shared across all of the parses, have been sampled. Different values of λ and γ are sampled before starting each random walk, producing both low and high entropy random walks as candidates.

3.4 Searching for sub-strokes

Before a parse can be scored (Eq. S4), the strokes must be sub-divided into sub-strokes. To do so, the strokes in each random walk are smoothed while enforcing that the trajectories stay within the original ink (as in Fig. S4), in order to correct for spurious curves from the thinning process (Fig. S4a). The smoothed strokes are then parsed into sub-strokes by running a simple greedy search with operators to add, remove, perturb, or replace "pauses" along the trajectory. To score the decomposition, the sub-strokes are classified as primitives z_i and scored by the generative model for strokes

$$P(x_i^{(m)}, y_i^{(m)}, z_i) = P(z_i) \prod_{j=1}^{n_i} P(y_{ij}^{(m)}|y_{ij}) P(y_{ij}|z_{ij}) \int P(x_{ij}^{(m)}|x_{ij}) P(x_{ij}|z_{ij}) \, \mathrm{d}x_{ij}, \qquad \text{(S7)}$$

where y_i is approximated by setting it equal to $y_i^{(m)}$. There is also a hard constraint that the spline approximation to the original trajectory can miss its target by no more than 3 pixels.

After the search process is run for each stroke trajectory, each candidate motor program with variables ψ and $\theta^{(m)}$ is fully-specified and tracks the image structure relatively closely. The stroke order is optimized for the best $2K$ motor programs, and then finally the prior score $P(\theta^{(m)}|\psi)P(\psi)$ is used to select the K best candidates to progress to the next stage of search, which fine-tunes the motor programs.

3.5 Optimization and fine-tuning

Holding the discrete variables fixed, the set of continuous variables (including $L^{(m)}$, $\tau^{(m)}$, $x^{(m)}$, $y^{(m)}$, $\epsilon^{(m)}$, $\sigma_b^{(m)}$) are optimized to fit the pixel image with Matlab's "active-set" constrained opti-

mization algorithm, using the full generative score as the objective function (Eq. S4). There are two simplifications to reduce the number of variables: the affine warp $A^{(m)}$ is disabled and the relations R_i are left unspecified and re-optimized during each evaluation of the objective function. After optimization finds a local maximum, the optimal joint setting of stroke directions and stroke order are chosen using exhaustive enumeration for characters with five strokes or less, while considering random subsets for more complex characters. Finally, the best scoring relations are chosen, and a greedy search to split strokes and merge strokes proceeds until the score can no longer be improved.

3.6 MCMC to estimate local variance

At this step, the algorithm has K high-probability parses $\psi^{[1]}, \theta^{(m)[1]}, ..., \psi^{[K]}, \theta^{(m)[K]}$ which have been fine-tuned to the images. Each parse spawns a separate run of MCMC to estimate the local variance around the type-level by sampling from $P(\psi|\theta^{(m)[i]})$. This is inexpensive since it does not require evaluating the likelihood of the image. Metropolis Hastings moves with simple Gaussian proposals are used for the shapes x, scales y, global positions L, and attachments τ. The sub-stroke ids z are updated with Gibbs sampling. Each chain is run for 200 iterations over variables and then sub-sampled to get $N = 10$ evenly spaced samples to form the $Q(\cdot)$ approximation to the posterior in Eq. S5.

3.7 Inference for one-shot classification

One-shot classification involves evaluating the probability of a test image $I^{(T)}$ given a single training image of a new character $I^{(c)}$ from one of $c = 1, ..., C$ classes. BPL uses a Bayesian classification rule for which an approximate solution can be computed

$$\underset{c}{\text{argmax}} \ \log P(I^{(T)}|I^{(c)}). \tag{S8}$$

Intuitively, the approximation uses the BPL search algorithm to get $K = 5$ parses of $I^{(c)}$, runs K MCMC chains to estimate the local type-level variability around each parse, and then runs K gradient-based optimization procedures to re-fit the token-level variables $\theta^{(T)}$ (all are continuous) to fit the test image $I^{(T)}$. The approximation becomes ((*70*) pg. 209 for a derivation)

$$\log P(I^{(T)}|I^{(c)}) \approx \log \int P(I^{(T)}|\theta^{(T)})P(\theta^{(T)}|\psi)Q(\theta^{(c)},\psi,I^{(c)})\,\mathrm{d}\psi\,\mathrm{d}\theta^{(c)}\,\mathrm{d}\theta^{(T)} \quad \text{(S9)}$$

$$\approx \log \sum_{i=1}^{K} w_i \max_{\theta^{(T)}} P(I^{(T)}|\theta^{(T)}) \frac{1}{N}\sum_{j=1}^{N} P(\theta^{(T)}|\psi^{[ij]}), \quad \text{(S10)}$$

where $Q(\cdot,\cdot,\cdot)$ and w_i are from Eq. S5. Fig. 4A shows examples of this classification score. While inference so far involves parses of $I^{(c)}$ refit to $I^{(T)}$, it also seems desirable to include parses of $I^{(T)}$ refit to $I^{(c)}$, namely $P(I^{(c)}|I^{(T)})$. We can re-write our classification rule (Eq. S8) to include both the forward and reverse terms (Eq. S11), which is the rule we use,

$$\underset{c}{\text{argmax}}\ \log P(I^{(T)}|I^{(c)}) = \underset{c}{\text{argmax}}\ \log P(I^{(T)}|I^{(c)})^2 = \underset{c}{\text{argmax}}\ \log[\frac{P(I^{(c)}|I^{(T)})}{P(I^{(c)})}P(I^{(T)}|I^{(c)})], \quad \text{(S11)}$$

where $P(I^{(c)}) \approx \sum_i \tilde{w}_i$ from Eq. S4. These two rules are equivalent if inference is exact, but due to our approximation, the two-way rule performs better as judged by pilot results.

3.8 Inference for one-shot generation

One-shot generation requires creating a new example image $I^{(2)}$ given another image $I^{(1)}$, and thus, it is desirable to produce samples from $P(I^{(2)},\theta^{(2)}|I^{(1)})$. Sampling from this posterior predictive distributions utilizes the approximate posterior $Q(\cdot)$ (Eq. S5). A distribution that is straightforward to sample from can be derived as follows:

$$\begin{aligned} P(I^{(2)},\theta^{(2)}|I^{(1)}) &= \int P(I^{(2)},\theta^{(2)}|\theta^{(1)},\psi)P(\theta^{(1)},\psi|I^{(1)})\,\mathrm{d}(\psi,\theta^{(1)}) \\ &= \int P(I^{(2)}|\theta^{(2)})P(\theta^{(2)}|\psi)P(\theta^{(1)},\psi|I^{(1)})\,\mathrm{d}(\psi,\theta^{(1)}) \\ &\approx \int P(I^{(2)}|\theta^{(2)})P(\theta^{(2)}|\psi)Q(\theta^{(1)},\psi,I^{(1)})\,\mathrm{d}(\psi,\theta^{(1)}) \\ &= \textstyle\sum_{i=1}^{K}\sum_{j=1}^{N} \frac{w_i}{N} P(I^{(2)}|\theta^{(2)})P(\theta^{(2)}|\psi^{[ij]}). \end{aligned}$$

The BPL inference algorithm was run to collect $K = 10$ parses of the image $I^{(1)}$. When using the above formulation directly, the model would repeatedly sample just the best-scoring parse in most cases, since even small differences in the parses can lead to large differences in weights w_i due to the high-dimensional raw data, a fact we attribute more to the generative model than the approximate inference algorithm. To avoid underestimating the variety of possible parses, the weights w_i were set to be inversely proportional to their rank order $1/\sigma(i)$ where $\sigma(\cdot)$ is the permutation function, or rank of the ith parse when sorted from highest to lowest score. The new sampling distribution is then

$$P(I^{(2)}, \theta^{(2)}|I^{(1)}) = \frac{1}{\sum_{i=1}^{K} \frac{1}{\sigma(i)}} \sum_{i=1}^{K} \frac{1}{\sigma(i)} \sum_{j=1}^{N} \frac{1}{N} P(I^{(2)}|\theta^{(2)}) P(\theta^{(2)}|\psi^{[ij]}). \tag{S12}$$

4 Alternative models

4.1 Lesioning Compositionality

Compositionality was lesioned to create a "one spline" version of BPL. The one spline model is a complete generative model, and it is shown performing classification in Fig. S5. As in BPL, there is a type and token-level distinction when generating characters. The algorithm for generating types in shown in Algorithm 3. The first step is to sample the number of control points in the spline n from the empirical distribution. The second step is to sample the control points for the spline. The first control point $P(X_1)$ is sampled from the empirical distribution of start positions. Additional control points are sampled from a first-order Markov Process given the previous control point $P(X_i|X_{i-1})$. Although not shown, the token-level model is analogous to BPL, where stroke variance is modeled as Gaussian noise on the control points $P(X_i^{(m)}|X_i) = N(X_i, \sigma_x^2 I)$. Other variables including the transformation $A^{(m)}$, pixel noise $\epsilon^{(m)}$, blur $\sigma_b^{(m)}$, and binary image $I^{(m)}$ are taken directly from BPL to facilitate a close comparison.

The hyperparameters were learned analogously to BPL. First, there is a prior on the number of control points. To estimate this, original drawings were converted into a single trajectory where a straight line was drawn between the end and beginning of consecutive strokes. This trajectory was modeled as a uniform cubic b-spline, and the number of control points was adaptively selected by adding control points until the error in the spline approximation plateaued (with a minimum of six control points). The empirical distribution on where to place a new control point given the previous was computed and modeled as a Gaussian mixture model with forty mixture components fit with expectation maximization. The distribution on the first control point was computed using the same procedure as BPL's position model for its first stroke.

Algorithm 3 One spline model: Sample a spline that defines a character concept

procedure GENERATEONESPLINETYPE
  $n \leftarrow P(n)$ ▷ Sample the number of control points
  $X_1 \leftarrow P(X_1)$ ▷ Sample the first control point
  for i = 2 ... n **do**
    $X_i \leftarrow P(X_i|X_{i-1})$ ▷ Sample the next control point given the previous
  end for
  $\psi \leftarrow \{n, X\}$
  return @GENERATEONESPLINETOKEN(ψ) ▷ Return function handle
end procedure

4.2 Lesioning Learning-to-learn: Type-level

Learning-to-learn is a key principle of BPL, and lesioning this process can help reveal the role of the model structure versus the learned parameters. This lesion disrupts the type-level parameters (Section 2.1), including the following:

- *Sub-stroke shape*. The distribution on type-level control points x_{ij}, which was previously defined by a Gaussian $P(x_{ij}|z_{ij}) = N(\mu_{z_{ij}}, \Sigma_{z_{ij}})$ for primitive identifier z_{ij}, was replaced by a uniform distribution in $\mathbb{R}^{10}$ bounded by the allowed range of the control points. This means that BPL favors no particular sub-stroke shape over any other.
- *Sub-stroke scale*. The scale of sub-strokes y_{ij}, which was previously modeled separately for each primitive as $P(y_{ij}|z_{ij}) = \text{Gamma}(\alpha_{z_{ij}}, \beta_{z_{ij}})$, was replaced by a uniform distribution between zero pixels and the entire length of the image frame.

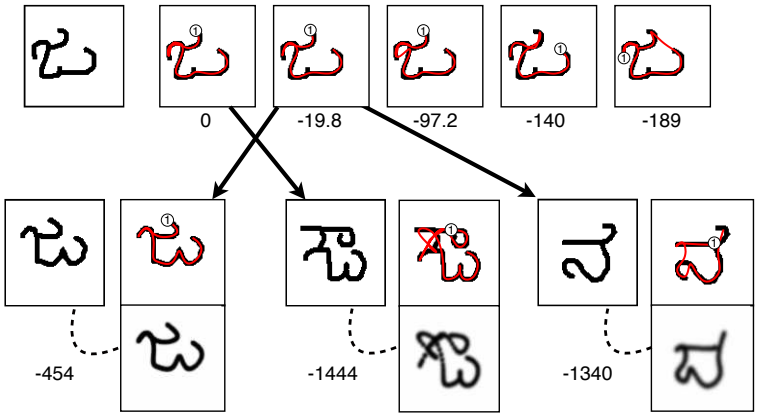

Figure S5: BPL with lesioned compositionality. A training image is fit and the five best complex splines (top row) are shown with their $\log w_j$ (Eq. S4). The white circle denotes a spline's starting position. These five parses were re-fit to three different test images of characters (left in image triplets), where the best parse (top right) and its associated image reconstruction (bottom right) are shown above its score (approximate $\log P(I^{(T)}|I^{(c)})$ in Eq. S10).

- *Stroke relations*. Since the interaction between strokes is another feature of the model learned from the background data set, the mixture of relations governed by θ_R was lesioned and only independent strokes were allowed.
- *Stroke position*. The distribution on start positions, previously fit to the empirical statistics (Fig. S2), was replaced as a uniform distribution in image space for all the strokes in a character.

The priors on $P(\kappa)$ and $P(n_i|\kappa)$ were not lesioned since they have minimal influence during inference. Taken together, these lesions remove most of the top-down influences during learning, leading to a model that attempts to fit the image as closely as possible while minimizing the number of sub-strokes. Although the model can produce any sub-stroke it needs at inference time, it has few preferences about what those trajectories should look like a priori. For example, Fig. S6 shows a diverse set of motor programs that produce a '+'-sign, where each program has the same prior probability. However, this does not imply that these programs have the same posterior probability: it is often the case that the more natural parses fit the raw pixels better than the alternatives, based strictly in the likelihood of the image. In these cases, however, most of the interesting work is done bottom-up by properties of the image rather than the prior.

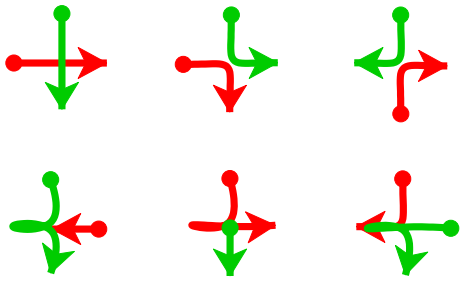

Figure S6: Program learning with no type-level learning-to-learn. Under this uninformed prior, each of these parses of a '+'-sign is equally likely a priori.

4.3 Lesioning Learning-to-learn: Token-level

This lesions disrupts the token-level parameters governing the variability across multiple instances of the same motor program (Fig. 3A-v), including parameters for spline shape σ_x, scale σ_y, and structural attachment σ_τ. Another variability parameter Σ_A governs the affine transformations. All were set to values that allow too much variability, three times their normal value, while the type-level structural parameters are left intact. The effect of having too little variability was explored in (*65*) and (*70*) through the "Affine" model.

4.4 Deep Convolutional networks

Deep convolutional networks (convnets), developed for handwritten digit recognition (*28*), are currently the state-of-the-art for objection recognition (*7, 71*). Using the Caffe software package (*72*), we adapted an architecture that achieved good performance on the MNIST digit benchmark (less than 1% test error), scaling it up to a large-scale classification task on the Omniglot background set. The convnet was trained to distinguish 964 characters based on 18 examples per class. Two examples per class were used for validation. The raw data was resized to 28 x 28 pixels and each image was centered based on its center of mass as in MNIST. We tried seven different architectures varying in depth and layer size, and we reported the model that performed best on the one-shot learning task.

For one-shot classification with 30 background alphabets for pre-training, two architectures achieved equivalent performance. The larger network is described below. There were three hidden layers: the first two were convolutional layers (120 and 300 feature maps respectively) followed by max pooling. The convolutions used a filter size of 5x5 with a stride of 1, and the pooling used a filter size of 2x2 with a stride of 2. The subsequent fully-connected layer had 3000 units and used dropout (*73*). The last layer was a 964-way softmax. All hidden layers used rectified linear activation functions. Weight decay of 0.0005 was also applied. One-shot classification was performed by computing image similarity through the feature representation in the 3000 unit hidden layer and using cosine similarity.

For one-shot classification with 5 background alphabets for pre-training, a smaller capacity model contributed the best one-shot performance. The architecture was analogous to above, except the first two convolutional layers had 30 and 75 feature maps respectively. The fully-connected layer had 750 units. The last layer had 136 or 156 softmax units depending on which set of 5 background alphabets was used.

We also report the results of (*33*) that trained a deep convolutional Siamese network to perform our task (*74*). The model was trained to make same/different judgments regarding character identity when presented with a pair of full resolution (105x105) images. The Siamese network was deeper with four convolutional layers and a standard hidden layer. The architecture and hyperparameters were optimized on a validation one-shot learning task using Bayesian optimization techniques, with training data from the 30 background training alphabets and 10 additional validation alphabets. They also used data augmentation to expand the background set by applying affine distortions to produce additional training examples. No results were reported in (*33*) for 5 alphabets of background training, but informal experiments by one of us suggest that performance would be worse (up to twice as many errors) when compared to background training with 30 alphabets.

4.5 Hierarchical Deep model

The Hierarchical Deep (HD) model is a probabilistic generative model that extends the Deep Boltzmann Machine (DBM) (*29*) so that it more elegantly handles learning new concepts from few examples. The HD model is derived by composing hierarchical nonparametric Bayesian models with the DBM. The HD model learns a hierarchical Dirichlet process (HDP) prior over the activities of the top-level features in a DBM, which allows one to represent both a layered hierarchy of increasingly abstract features and a tree-structured hierarchy of super-classes for sharing abstract knowledge among related classes. Given a new test image, the approximate posterior over class assignments can be quickly inferred as detailed in (*29*). The original images were down-sampled to 28x28 pixels with greyscale values from [0,1]. The background set was also artificially enhanced by generating slight modifications of the training images, including translations (+/- 3 pixels), rotations (+/- 5 degrees), and scales (0.9 to 1.1). This helps reduce overfitting and learn more about the 2D image topology, which is built in to some deep models like convolutional networks. The Hierarchical Deep model is more "compositional" than the deep convnet, since learning-to-learn endows it with a library of high-level object parts (*29*). However, the model lacks a abstract causal knowledge of strokes, and its internal representation is quite different than an explicit motor program.

To generate a new example of a new character, the model quickly approximately infers which super-class the new character belongs to. Given the super-class parameters, the model samples the states of the top-level DBM's features from the HDP prior, followed by computing grayscale pixel values for the bottom layer of the DBM.

5 Supplementary Methods

5.1 Collecting behavioral data from Mechanical Turk

All experiments were run on the Amazon Mechanical Turk (AMT) online platform using participants in the USA. Different experiments required unique Amazon worker ID numbers. The experiments typically took participants between 10 and 20 minutes, and we aimed to pay approximately six dollars per hour. Participants answered multiple choice questions about the instructions, and they were redirected back to the instructions until they answered all questions correctly (*75*). All visual Turing tests were binary forced choice tasks, and participants were excluded if we detected long sequences of repeated or alternating responses at the level $p < .0001$.

Some tasks asked participants to draw handwritten characters using a mouse or track pad. The raw mouse trajectories contain jitter and discretization artifacts, and thus spline smoothing was applied. Although BPL's ink model is a reasonable approximation to the actual ink model produced by the online interface, low-level image differences were entirely eliminated by re-rendering stroke trajectories in the same way for human and machine drawings.

5.2 One-shot classification

Ten alphabets were chosen from the Omniglot evaluation set to form the ten within-alphabet classification tasks (Fig. S7). Each task has 40 trials consisting of a test image compared to just one example of 20 new characters from the same alphabet. Fig. 1B shows an example trial. The images for each task were produced by four relatively typical drawers of that alphabet, and the set of 20 characters was picked to maximize diversity when alphabets had more than 20 characters. The four drawers were randomly paired to form two groups, and one drawer in each group provided the test examples for 20 trials while the other drawer provided the 20 training examples for each of these test trials. For the four alphabet tasks in Fig. S7, rows 1 and 2 were

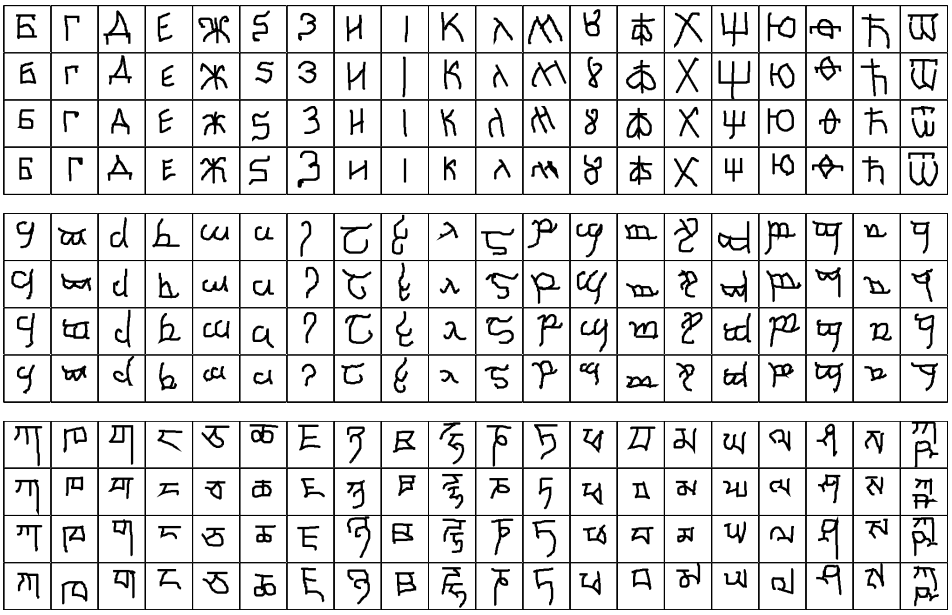

Figure S7: Three alphabets used to evaluate classification with 20 characters drawn by four people.

paired (where row 1 were the training examples) and rows 3 and 4 were paired (where row 3 was the training examples). Classification accuracy for people was estimated by evaluating forty AMT participants using a interface similar to the example trial in Fig. 1B. To ensure that participants saw each character only once, participants completed just one randomly selected trial from each of the 10 alphabet tasks. There was feedback after every trial.

5.3 Generating new examples

People and computational models were compared on the task of generating new examples of a new concept. The task used 50 randomly selected characters from the Omniglot evaluation set.

5.3.1 Production task

Eighteen human participants on AMT were asked to "draw a new example" of 25 characters, resulting in nine examples per character. To simulate drawings from nine different people, each of the models generated nine samples after seeing exactly the same images that people did.

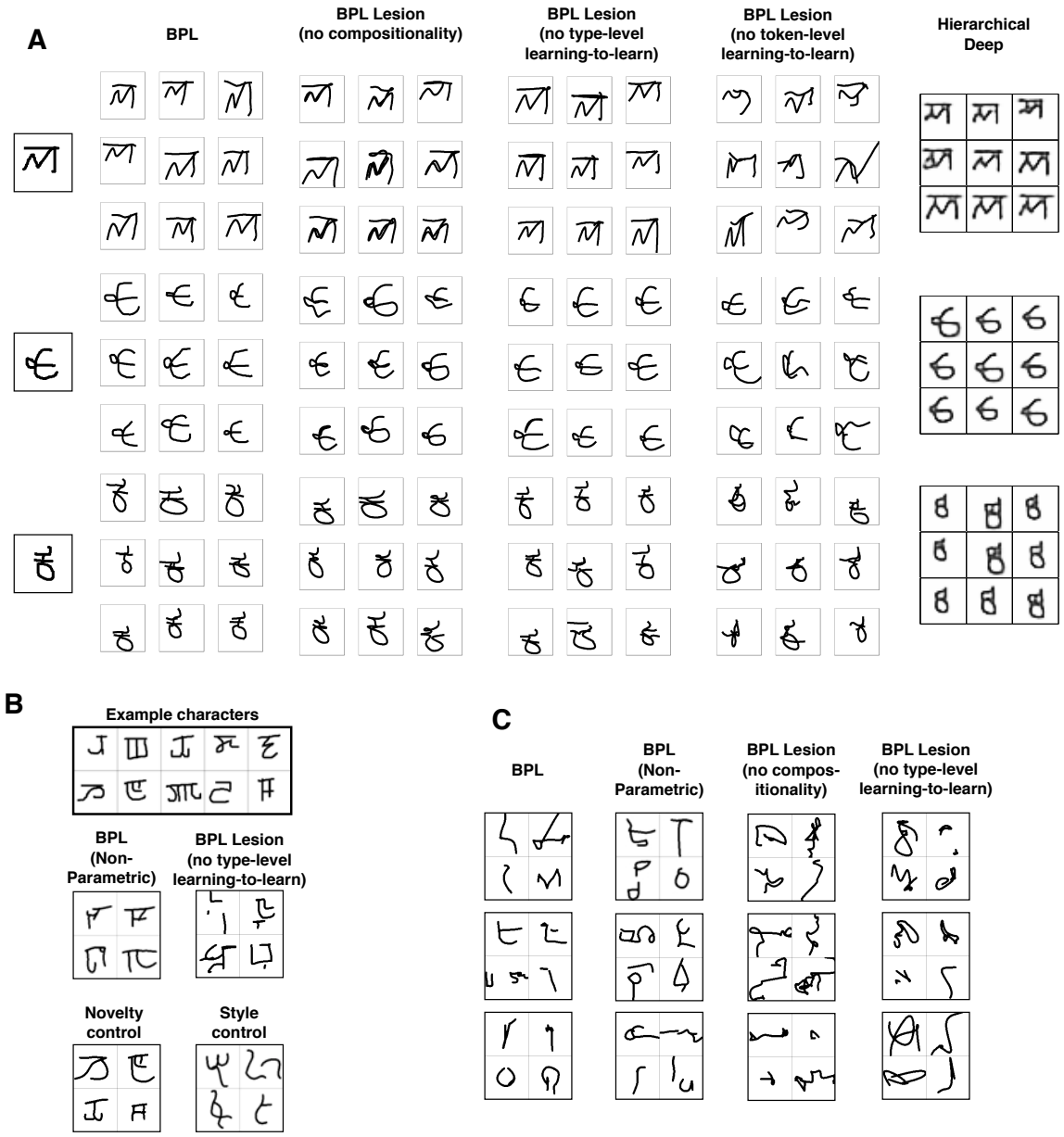

Figure S8: Various computational models (visual Turing test conditions) performing the tasks of generating A) new exemplars, B) new concepts (from type), C) new concepts (unconstrained). In A), the provided example for each new character is shown on the left. In B), the example characters of an alphabet (type) are shown above the generated examples.

Grids of generated characters for each model is shown in Fig. S8. Rather than sampling the sizes of the characters, they were manually set to match a human drawings to control for this potential low level cue.

5.3.2 Visual Turing test

Five models were compared using different between subjects conditions in the visual Turing test evaluation. The models included BPL and the three lesioned models described in Section 4. A relatively easy-to-detect "Affine" model was also compared but it is not reported due to space constraints (see (*70*) for results). The evaluation involved 150 human judges on AMT randomly assigned to conditions, and all but one condition were run at the same time.[4]

A separate AMT experiment ran 60 more participants to evaluate BPL after training from just five background alphabets. There were two conditions corresponding to the two different sets of 5 alphabets (Section S2.3). One judge participated in a previous experiment and was excluded.

Participants were told that they would see a target image and new examples of that character. There were two grids of 9 images, where one grid was drawn by people with their computer mice and the other grid was drawn by a computer program that "simulates how people draw a new character." Which grid is which? Example trials are shown in Fig. 5. The stimuli produced by the computer program varied by condition (Fig. S8A). Participants saw 49 trials[5] and ID level was revealed after each block of 10 trials. Participant responses were made by pressing the left and right arrow keys. Three participants were removed since two people pressed just one key for the whole experiment and one reported technical difficulties.

[4]The lesion of token-level learning-to-learn condition was not run with the other conditions. It was run at a later date with 50 participants randomly assigned to either that condition or the BPL condition, where the latter condition was a replication to guard against the possibility of a changing AMT population. The group average ID level was nearly the same, so the data was collapsed into a single experiment.

[5]One character was not tested since BPL's optimization procedure failed on a few of its parses.

5.4 Generating new exemplars (dynamic)

5.4.1 Visual Turing test

This was a dynamic version of the exemplar generation task in Section 5.3. Five different models were evaluated. Two factors were manipulated orthogonally: the quality of the parses (BPL vs. BPL with lesioned type-level learning-to-learn; Section 4.2) and whether dynamic properties such as stroke order and stroke direction were random or optimized given the learned model parameters. BPL was tested in three flavors: standard, with random stroke order and directions, and with reverse stroke order and direction (the drawing trajectory appears to play backwards from the original prediction). For BPL with lesioned learning-to-learn at the type level, the parse was always determined by the lesioned model. The stroke order and direction was either random or optimized based on the non-lesioned BPL hyperparameters.

There were one hundred fifty participants randomly assigned to one of the five model conditions. There were fifty trials, and a trial displayed pairs of movies instead of pairs of entire character grids. One movie showed a person drawing and one movie showed a computer program drawing, and participants were asked to indicate which movie shows the computer. The two movies were played sequentially, and each movie paused on the last frame after playing. Participants also had the option of re-watching the sequence.

Since the models do not make millisecond-scale predictions, drawings were normalized for length and drawing speed. The human and model drawing movies were rendered to be approximately five seconds long, where the time spent drawing each stroke was proportional to its length. Pauses were shown between different strokes but not different sub-strokes. As explained to the participants, the ink color changed every time the drawer lifts up the pen, making it easier to track the different pen strokes. Accuracy was displayed every ten trials. Seven participants were removed due to technical problems.

5.5 Generating new concepts (from type)

This task asked people and computational models to generate new concepts in the style of an alphabet. The ten alphabets were selected from the Omniglot evaluation set.

5.5.1 Production task

Thirty-six participants on AMT created one character for each alphabet. For each alphabet, ten example characters were shown as in Fig. 7A-i. Participants had only three seconds to draw once they started placing ink on the canvas, preventing them from drawing highly complex characters that are a clear giveaway of human design. Participants could choose to redo drawings until they were satisfied with the outcome.

5.5.2 Visual Turing test

Four different computational models were evaluated on the same task, with example productions shown in Fig. S8B. Two models included a non-parametric extension to BPL (Section 7) and a version with lesioned type-level learning-to-learn (Section 4.2). The other two models served as controls, testing that criteria that characters must be novel and stylistically consistent to pass the visual Turing test. The novelty control generated new tokens of the provided characters, and a style control used the BPL produced characters with permuted alphabet labels, so the model predictions were always assigned to the wrong alphabet.

One hundred and twenty judges were run in these four conditions. More participants were run in one condition.[6] Participants were told they would see a series of displays with ten "example characters" from a foreign alphabet, and that people were asked to design a new character that appears to belong to the same alphabet. The same task was also given to a computer

[6]Twenty of those participants were run two days later and assigned only to the style control group, since only eleven participants were randomly assigned to that group originally. There was an average ID level difference of 9% across the two phases, which approached significance ($t(29) = 2.0,\ p = 0.0502$).

program that simulates how people design new characters. Participants were given additional information about the drawers: drawers were asked not to directly copy characters, they were only given three seconds to draw each character, and they used their computer mice. Participants saw 90 displays like those in Fig. 7A where they were asked to indicate which grid of four images they thought was drawn by the computer program. They were told the other grid of four images was drawn by four different people. The displays were divided into nine blocks of ten displays, where each block had one display from each alphabet. Accuracy was displayed at the end of each block. Three participants were not analyzed, since two reported technical difficulties and another failed the guessing check (Section 5.1).

5.6 Generating new concepts (unconstrained)

5.6.1 Production task

Twenty one people on AMT participated. They were asked to invent ten plausible new characters with three seconds of drawing time per character. During the instructions, participants were shown forty Omniglot characters for inspiration, but they were informed that their drawings should not resemble known symbols or characters from the instructions. The ten characters should also be visually distinct. One participant was removed for poor quality drawings.

5.6.2 Visual Turing test

Four models were also evaluated on this task, as shown in Fig. S8A. For BPL, samples from the prior $\bar{P}(\psi)$ were taken as described in Section 2.1 and 7.2. Samples were produced using importance sampling to make 200 new characters from a base of 4000 samples from $P(\psi)$. Another way to operationalize this task is through a more explicit re-use of parts from previously learned characters. For this model, a non-parametric BPL (Section 7) was learned from the entire set of 100 characters in the previous experiment, treating this set as a single "alphabet" (Section 5.5). Two versions of the basic model were also tested with lesioned compositionality

and lesioned type-level learning-to-learn (Section 4).

There were 125 participants assigned to one of the four conditions. All but one condition were run concurrently.[7] The instructions described that people were asked to design new characters, as was a computer program that simulates how people design new characters. Participants were given additional information about the drawers: drawers were asked not to draw characters that resemble other letters or symbols they know, they were only given three seconds to draw, and they used their computer mice. Participants saw 50 displays like those in Fig. 7B. The displays were divided into five blocks where accuracy was shown after every ten trials. One participant was not analyzed after reporting technical difficulties.

6 Supplementary results and discussion

6.1 One-shot classification

Supplementing the results in Fig. 6A, a lesion to BPL applied at the type-level learning-to-learn did not change the final level of classification performance, achieving a 3.5% error rate. This suggests that a preference for the parts and sub-parts present in previous concepts is not critical for the classification task, nor is the need for explicit relations over a global position configuration. A more general notion of bottom-up parts, in the spirit of previous proposals such as (*76*), may suffice for classification, but not for some of the other more creative abilities as other tasks suggest.

[7]The experiment was run in two phases six days apart. The first phase of 75 participants included three conditions: BPL and the two lesioned conditions. The second phase of 50 participants included two conditions: non-parametric BPL and lesioned learning-to-learn, where the lesion condition was repeated so the two populations of judges could be compared. The mean accuracies in the replicated condition were a percentage point apart and not significantly different.

6.2 Generating new exemplars

The Hierarchical Deep model (Section 4.5) does not produce well-articulated strokes so it was not quantitatively analyzed, although there are clear qualitative differences between these and the human produced exemplars (Fig. S8A). As in one-shot classification, lesioning type-level learning-to-learn did not dramatically decrease performance; judges had an average identification (ID) level of 54% and only 3 of 21 were above chance, defined when their 95% confidence interval (Clopper-Pearson approximation) rules out guessing and below. Even with this lesion, BPL generated compelling new examples that fool a majority of participants. Although performance was reliably better than chance ($t(20) = 2.25, p < 0.05$), it was not reliably easier to detect than BPL ($t(67) = 0.79, p > 0.05$).

6.3 Generating new exemplars (dynamic)

First, we report some additional details regarding the results in the main text and Fig. 6B. For BPL, judges had a 59% ID level (6/30 judges reliably above chance), where the group average was significantly better than chance ($t(29) = 6.54, p < .001$). Lesioning the predicted dynamics by randomizing the stroke order and directions lead to a 71% ID level (20/29 above chance), which was significantly easier to detect than BPL ($t(57) = 4.96, p < .001$).

Second, we report some additional results not mentioned in the main text. An additional lesion visualized BPL's predicted dynamics in reverse, leading to an increased ID level of 67% (10/23 above chance) and was also significantly easier to detect than BPL ($t(51) = 2.81, p < .01$). Many participants who saw the lesioned predictions commented on these unnatural ordering choices in the survey.

These structured dynamic cues may have played a different role in models without type-level learning-to-learn. Judges had a 63% ID level on average for detecting a model with optimal dynamics (14/33) and equally accurate at 63% when this model had random dynamics (10/28).

A 2 x 2 ANOVA with learning-to-learn and random/optimal dynamics as factors showed a main effect of dynamics ($F(1, 116) = 10.0, p < .01$) and no significant main effect type-level learning-to-learn. Interestingly, there was a significant interaction ($F(1, 116) = 8.5, p < .01$), suggesting that judges were more sensitive to unrealistic dynamics when the parses were better (Fig. S9).

6.4 Generating new concepts (from type)

Supplementing the results in Fig. 6B, we report the results on two control models. The novelty control generated new tokens of the provided characters rather than new character types. Did the visual Turing test judges notice, taking creativity into account? The judges had a 68% ID level on average (13/26 above chance), which was better than chance ($t(25) = 4.89, p < .001$) and significantly easier to discriminate than BPL ($t(59) = 4.64,\ p < .001$). The novelty of the characters seemed to be an important cue to the judges, a fact that was also reflected in their self-reported strategies in the post-experiment survey.

The style control assigned the characters generated by BPL to the wrong alphabet. Interestingly, judges in this condition had a 49% ID level on average (6/31 above chance), which is about the same level of performance as BPL with the correct alphabets. To investigate why judges did not pass the visual Turing test, we analyzed the survey data regarding strategy selecting those who mentioned stylistic similarity to the example characters. For the participants noted this cue in their self-report (9/31), they were split on the direction: four participants thought the computer tended to be more different than the examples (e.g. "I figure people would be more similar to the examples" and "I thought the computer would be less accurate in regard to what looked like the alphabet."), and five participants thought the humans tended to be more different than the examples (e.g., "I thought the computer program... would more closely resemble the actual letters"). Moreover, their performance on the task reflected how they

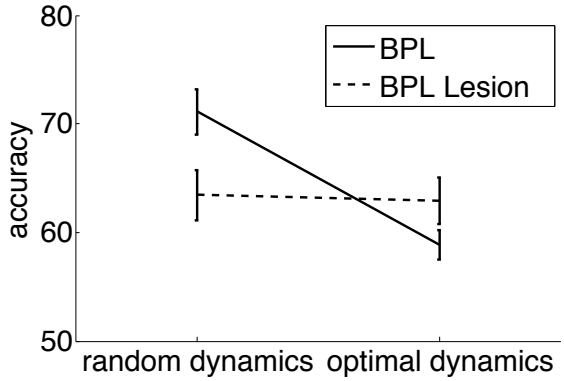

Figure S9: Interaction between type-level learning-to-learn (BPL vs. BPL Lesion) and dynamics on visual Turing test performance. Bars show the mean $\pm$ SEM.

interpreted the cue, where judges who mentioned the cue in the right direction had a 66% ID level on average compared to 43% for those who got the cue reversed ($t(7) = 7.22,\ p < .001$). A plausible interpretation is that some people pick up on the stylistic cue, but they were not given enough feedback to decide whether it was a giveaway for the humans or machines. An experiment reported in Chapter 5 of (*70*) probed this issue more deeply by using an alphabet classification task to compare the strength by which people and BPL capture alphabet style, finding that the model captured style about equally strong as people did. Finally, it is possible that another population of judges with additional expertise, such as artists or copyists, might be more sensitive to these stylistic signatures as well as other differences between the human and machine drawings.

7 Non-Parametric Extension of Bayesian Program Learning

This section describes a non-parametric extension of BPL that can generate new concepts in the style of an alphabet, adding an additional layer of hierarchy. First, an alphabet is generated and then new types are generated from that alphabet. The model from tokens down is the same as described in Section 2.2. The model schematic is shown in Fig. S10 which can be compared

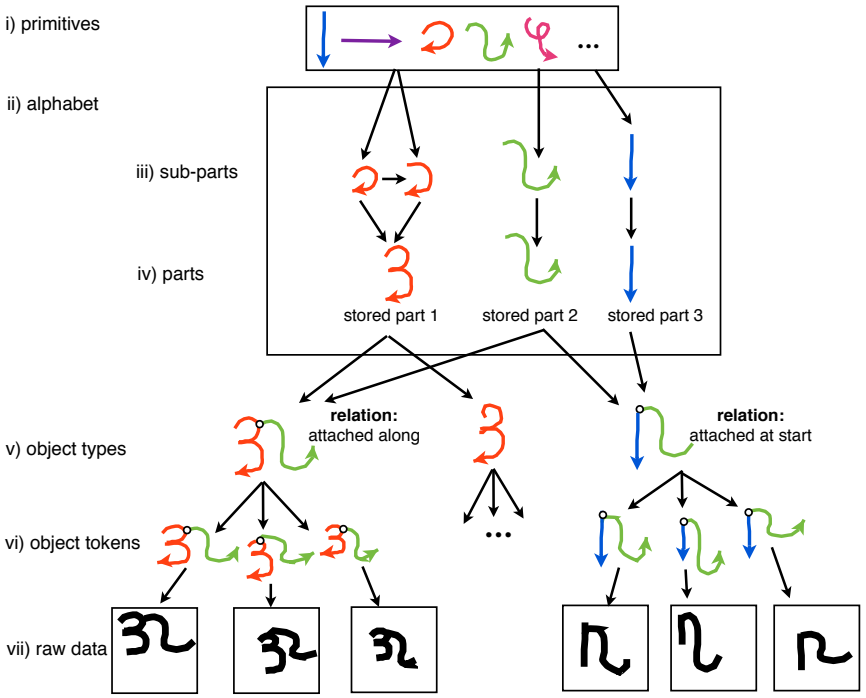

Figure S10: The generative process for producing multiple characters from a particular alphabet. New parts (strokes) are generated by choosing primitive actions from a library (i) and combining these sub-parts (iii) to make parts (iv). Objects belonging to an alphabet are created by re-combining parts to define simple programs (v). The exemplar model and image model are the same as in standard BPL (vi-vii).

to the original in Fig. 3A. By adding the alphabet level, the model exerts a pressure to re-use major structural components among a set of related characters. One of the main differences is that all of the generated parts (strokes) are explicitly stored (level ii Fig. S10), and there is a bias to re-use parts that have already been created as an alternative to generating entirely new parts. It is possible for the flat model (Fig. 3) to generate the three characters shown in Fig. S10 where entire strokes are shared across characters, but it is more likely when the prior favors re-using existing strokes.

7.1 Generating a set of related concepts

The full generative process is shown in Algorithm 4. The procedure for making alphabets does not sample concrete variables. Rather, it defines the transformation that ties the various

programs within an alphabet together, using a tool from statistics and machine learning called the Dirichlet Process (DP) (*77, 78*). The DP takes a concentration parameter α and a base distribution/procedure $P(\cdot)$ as inputs. It returns a new distribution ***P-mem***$(\cdot)$, which serves to transform the original and induce dependencies between previously independent samples. For instance, conditional samples $d_{J+1}|d_1, \ldots, d_J \sim$ ***P-mem***$(\cdot)$ are of the form

$$d_{J+1}|d_1, \ldots, d_J \sim \sum_{j=1}^{J} \frac{\delta(d_{J+1} - d_j)}{J + \alpha} + \frac{\alpha P(d_{J+1})}{J + \alpha}, \tag{S13}$$

where $\delta(\cdot)$ is a delta function and the distribution encourages the re-use of previous values. It is also related to the Chinese Restaurant Process (CRP), a metaphor where a new customer $J + 1$ comes into a restaurant where J customers are already seated around a set of tables. All customers at a given table share the same value of d_j. The new customer joins a previous table l with probability equal to $m_l/(J + \alpha)$, where m_l are the number of customers at that table. The new customer can also sit at a new table with probability $\alpha/(J+\alpha)$ where a new value for d_{J+1} is sampled from the base distribution $P(\cdot)$.

Algorithm 4 takes the procedures for sampling from various conditional probability distributions $P(\cdot)$, such as the procedure for sampling the number of strokes $P(\kappa)$ or entire strokes GENERATESTROKE(i, n_i) (Algorithm 1), and passes them through the higher-order procedure $DP(\alpha, \cdot)$ to return new procedures ***P-mem***$(\cdot)$. The new "memoized" (*79, 80*) procedures define a set of probability distributions with the CRP clustering property, used for learning the number of strokes κ, the number of sub-strokes n_i, the strokes S_i, and the relation types ξ_i that are characteristic of a particular alphabet.

Algorithm 4 Generate a new alphabet

procedure GENERATEALPHABET
  P-mem$(\kappa) \leftarrow DP(\alpha, P(\kappa))$
  for κ = 1 ... 10 **do**
    P-mem$(n_i|\kappa) \leftarrow DP(\alpha, P(n_i|\kappa))$
  end for
  for i = 1 ... 10 **do**
    for n_i = 1 ... 10 **do**
      P-mem$(S_i|n_i) \leftarrow DP(\alpha, \text{GENERATESTROKE}(i, n_i))$
    end for
  end for
  P-mem$(\xi_i) \leftarrow DP(\alpha, P(\xi_i))$
  $A \leftarrow \{$*P-mem*$(\kappa); \forall\kappa$: *P-mem*$(n_i|\kappa); \forall i, n_i$: *P-mem*$(S_i|n_i)\}$
  return @GENERATETYPE(A)
end procedure

procedure GENERATETYPE(A)
  $\kappa \leftarrow$ *P-mem*(κ) ▷ Sample the number of strokes
  for i = 1 ... κ **do**
    $n_i \leftarrow$ *P-mem*$(n_i|\kappa)$ ▷ Sample the number of sub-strokes
    $S_i \leftarrow$ *P-mem*$(S_i|n_i)$ ▷ Sample a stroke with n_i substrokes
    $\xi_i \leftarrow$ *P-mem*(ξ_i) ▷ Sample the type of a stroke's relation
    $R_i \leftarrow P(R_i|\xi_i, S_1, ..., S_{i-1})$ ▷ Sample the details of the relation
  end for
  $\psi \leftarrow \{\kappa, R, S\}$
  return @GENERATETOKEN(ψ) ▷ Return program handle
end procedure

The alphabet A is a collection of these new procedures, passed to GENERATETYPE(A) to tie the generation of new character types together, meaning that individual characters are no longer generated independently from each other. Otherwise GENERATETYPE(A) is analogous to the previous definition in Algorithm 1. The procedure GENERATETOKEN(ψ) remains unchanged from Algorithm 2. As a replacement for Equation 1, the joint probability distribution on J character types $\psi^{(1)}, \ldots, \psi^{(J)}$, with M character tokens of each type $\theta^{(1,1)}, \ldots, \theta^{(1,M)}$, $\ldots, \theta^{(J,1)}, \ldots, \theta^{(J,M)}$, and binary images $I^{(j,m)}$ is given as follows,

$$P(\psi^{(1)}, \ldots, \psi^{(J)}, \theta^{(1,\cdot)}, \ldots, \theta^{(J,\cdot)}, I^{(1,\cdot)}, \ldots, I^{(J,\cdot)}) = \prod_{j=1}^{J} P(\psi^{(j)}|\psi^{(1)}, \ldots, \psi^{(j-1)}) \prod_{m=1}^{M} P(I^{(j,m)}|\theta^{(j,m)}) P(\theta^{(j,m)}|\psi^{(j)}). \quad \text{(S14)}$$

where $\theta^{(j,\cdot)}$ is shorthand for all M examples of type j, or $\{\theta^{(j,1)}, \ldots, \theta^{(j,M)}\}$.

7.2 Global constraints

Standard BPL defines a generative model for new types through a series of simple conditional probability distributions (CPD) fit to local empirical statistics in the background dataset (Section 2.3). This setup does not guarantee that samples from the prior will match the global statistics of the training data, especially if there are additional correlations in the dataset that the prior

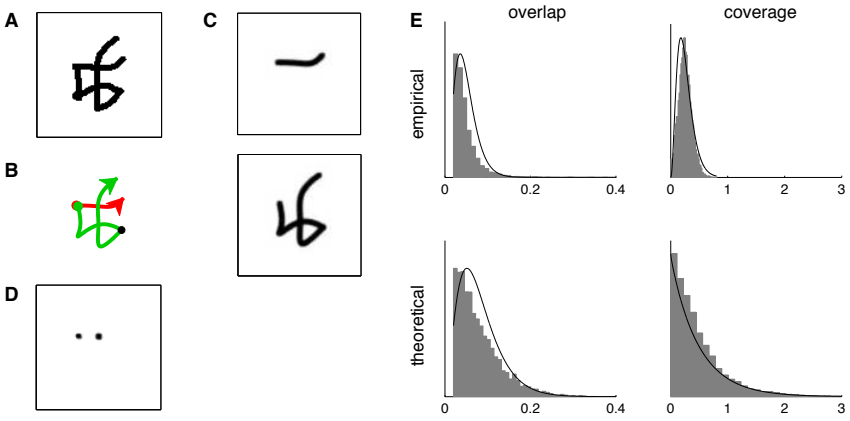

Figure S11: Computing the stroke overlap statistic. Original image (A) and the type-level footprint of its parse (B). Image footprints of the two strokes (C). Overlap image (D). Distributions on stroke overlap and coverage comparing background set (empirical) with samples from the prior (theoretical) with Gamma distribution fits (E).

cannot model. The more creative tasks test these holistic properties more directly, and thus this section describes two global properties the model misses and describes a matching method to mitigate the issue. This method was only used for the two tasks that required generating new concepts.

The solution requires defining a "type-level footprint" which is a projection of the type level variables ψ into the image canvas, which is useful for defining features that operate at the type-level. A footprint is defined by setting the token-level variables $\theta^{(m)}$ to their most likely values $\underset{\theta^{(m)}}{\text{argmax}}\, P(\theta^{(m)}|\psi)$ where the noise variables $\epsilon^{(m)}$ and $\sigma_b^{(m)}$ are set to their smallest values.

7.2.1 Stroke overlap

The first global property is the amount of overlap between different strokes (Fig. S11e). Beyond the relations that define start position, strokes are sampled independently in the prior, leading to the possibility that new strokes can overlap in unnatural ways with previous strokes. Overlap is measured by taking the probability of each pixel being turned on in the type-level footprint, called a probability map, for each stroke rendered separately (Fig. S11c). The probability maps

are added together, and the "over-determined" pixels, defined as having a value greater than 1, signal stroke overlap as shown in Fig. S11d. The overlap statistic is the total amount of over-determination, summed over pixels, divided by the total amount of ink across all strokes. For example, the character in Fig. S11 has a value of 0.04. The empirical distribution on stroke overlap, as computed from the drawings in the background set, is shown in Fig. S11e-left. The theoretical distribution is computed by sampling from $P(\psi)$ as defined in Section 2.1. Evidently, the prior produces characters with too much stroke overlap, and Section 7.2.3 describes how the model can be adapted accordingly.

7.2.2 Stroke coverage

The second statistic measures how much space the character covers in the image canvas based on its type-level footprint. The area of the bounding box of the footprint is computed, and the coverage statistic is the ratio of the area of the bounding box over the area of the image canvas. For instance, the motor program in Fig. S11 has a value of 0.2. Evidently, most real characters take up less than half the image canvas (Fig. S11e-right). On the other hand, the model can occasionally draw large and intricate characters that go outside the image canvas resulting in very large coverage values.

7.2.3 Matching the statistics

The empirical and theoretical histograms were fit with Gamma distributions.[8] Let $f_1(\psi)$ be a function that computes stroke overlap for a character ψ and returns its density under the fit empirical distribution (Fig. S11e). Likewise, let $f_2(\psi)$ return the density under the empirical coverage distribution. Let $g_1(\cdot)$ and $g_2(\cdot)$ be the analogous functions for the theoretical distributions. A new probability distribution over character types $\bar{P}(\psi)$ can be defined by taking the

[8]Since many characters have an overlap of zero, the overlap distribution was mixture model of a Gamma distribution for values over 0.02 and a uniform distribution between 0 and 0.02.

original distribution $P(\psi)$ from Section 2.1 and re-weighting it by

$$\bar{P}(\psi) \propto \frac{f_1(\psi) f_2(\psi)}{g_1(\psi) g_2(\psi)} P(\psi). \tag{S15}$$

Samples from $\bar{P}(\psi)$ will have global statistics that approximately match the empirical distributions $f_1(\cdot)$ and $f_2(\cdot)$, and they can be produced using importance sampling by sampling from $P(\psi)$ and re-weighting the samples by the ratio in Equation S15. For the more sophisticated non-parametric model, as described in Section 7, the probability of new character types can be defined analogously to better capture coverage and overlap

$$\bar{P}(\psi^{(j)} | \psi^{(1)}, \ldots, \psi^{(j-1)}) \propto \frac{f_1(\psi^{(j)}) f_2(\psi^{(j)})}{g_1(\psi^{(j)}) g_2(\psi^{(j)})} P(\psi^{(j)} | \psi^{(1)}, \ldots, \psi^{(j-1)}). \tag{S16}$$

7.3 Inference in Non-Parametric BPL

The problem of generating an example of a new concept ($I^{(J+1,1)}$) from a set of related concepts, with just one example each ($I^{(1,1)}, \ldots, I^{(J,1)}$), can be formulated as probabilistic inference where the relevant posterior predictive quantity is

$$I^{(J+1,1)} \sim P(I^{(J+1,1)} | I^{(1,1)}, \ldots, I^{(J,1)}). \tag{S17}$$

This is a difficult distribution to sample from. It contains all of the complexity of the inference problem for one-shot learning (Section 3) as well as substantial additional complexity due to the added statistical dependency between the parses of characters, although this is largely ignored in our approximate sampling procedure. We make another simplification by using the re-weighted distribution $\bar{P}(\psi)$ (Equation S15) for forward sampling of new characters (again, not for forward sampling new exemplars), while using the original $P(\psi)$ for inference.[9]

[9]The weight terms could be included during inference, at the additional expense of evaluating these weight functions for every considered setting of the type-level variables ψ. There are reasons to believe that including this term would only make a minimal difference. For instance, the coverage statistic has almost exactly the same value for all parses of a given image. Also, computational experiments suggest that for fitting an image with five parses, only in about 5% of cases does the rank order of those parses change when the model score is computed exactly with $\bar{P}(\psi)$ rather than $P(\psi)$.

This inference procedure is described in more detail below, but the gist of the procedure is described here first. The algorithm finds a motor program for each image $I^{(1,1)}, \ldots, I^{(J,1)}$ in the alphabet separately using the inference techniques in Section 3. The statistics across sets of variables in these programs are analyzed, including counts of the number of strokes, number of sub-strokes given a number of strokes, types of relations, etc. The strokes found in each program are also saved and stored, with a separate bag for the first stroke, second stroke, etc. across all of the programs. Given pressure from the Dirichlet Process priors, similar strokes can also merge their type-level parameters. To sample a new character $I^{(J+1,1)}$, rather than sampling its variables from the general prior, the number of strokes, sub-strokes, relations, and strokes themselves can be re-used from the empirical statistics of the alphabet.

There are several steps to producing a sample from Equation S17. The first goal is to produce a sample from the type-level motor programs for each image in the alphabet

$$P(\psi^{(1)}, \ldots, \psi^{(J)} | I^{(1,1)}, \ldots, I^{(J,1)}). \tag{S18}$$

Given this sample, we can sample a new type of character using Algorithm 4 and the CRP Equation S13 for each *P-mem*$(\cdot)$,

$$\psi^{(J+1)} | I^{(1,1)}, \ldots, I^{(J,1)} \sim P(\psi^{(J+1)} | \psi^{(1)}, \ldots, \psi^{(J)}). \tag{S19}$$

Finally, to produce a sample from Equation S17, we then sample from a series of conditionals to produce a new character

$$I^{(J+1,1)} | I^{(1,1)}, \ldots, I^{(J,1)} \sim P(I^{(J+1,1)} | \theta^{(J+1,1)}) P(\theta^{(J+1,1)} | \psi^{(J+1)}). \tag{S20}$$

Samples from Equation S18 are approximated as

$$\begin{aligned}
&P(\psi^{(1)}, \ldots, \psi^{(J)} | I^{(1,1)}, \ldots, I^{(J,1)}) \\
&= \int P(\psi^{(1)}, \ldots, \psi^{(J)} | \theta^{(1,1)}, \ldots, \theta^{(J,1)}) P(\theta^{(1,1)}, \ldots, \theta^{(J,1)} | I^{(1,1)}, \ldots, I^{(J,1)}) \, \mathrm{d}(\theta^{(1,1)}, \ldots, \theta^{(J,1)}) \\
&\approx P(\psi^{(1)}, \ldots, \psi^{(J)} | \theta^{(1,1)[*]}, \ldots, \theta^{(J,1)[*]})
\end{aligned}$$

where

$$\psi^{(j)[*]}, \theta^{(j,m)[*]} \leftarrow \underset{\psi^{(j)}, \theta^{(j,m)}}{\text{argmax}} \; P(\psi^{(j)}, \theta^{(j,m)} | I^{(j,m)}) \tag{S21}$$

is the approximate maximization computed using the inference techniques in Section 3. Thus, the motor programs for each character are fit independently, and then the model samples shared type-level parameters ψ conditioned on the token-level variables θ. This procedure will underestimate the amount of type-level variable sharing across characters, but it is good enough for the purposes of generating new types of characters in the style of previous ones. A more accurate inference may be needed for related tasks such as alphabet classification. Assuming this approximation, it is relatively straightforward to sample from

$$P(\psi^{(1)}, \ldots, \psi^{(J)} | \theta^{(1,1)[*]}, \ldots, \theta^{(J,1)[*]}) \tag{S22}$$

due to its many conditional independences. In fact, the number of strokes κ, number of sub-strokes n_i, and relation types ξ_i are all uniquely determined by the token-level and require no additional computation to sample. However, it is possible that similar strokes will cluster at the type-level, and samples from the joint distribution on stroke types are produced using a MCMC algorithm. The sampler is initialized by placing each unique stroke S_i at its own table (for a total of L tables), using the Chinese Restaurant Process metaphor. Initially, there is no sharing. During each iteration of MCMC, the type-level sub-stroke shape x_i and scale y_i variables at each table are updated using Metropolis Hastings moves with Gaussian proposals. The discrete sub-stroke ids z_i at each table are also re-sampled using Gibbs sampling. Finally, the table assignments of token-level strokes to type-level strokes is sampled using Gibbs sampling. The last sample after 200 iterations was then used for the purpose of sampling all of the new character types $\psi^{(J+1)}$ (Equation S19).

7.4 Generating new concepts

Examples of 36 new characters for six of the alphabets are shown in Fig. 7A. Given just a single image of J new characters, $I^{(1,1)}, \ldots, I^{(J,1)}$, new types are sampled from

$$\bar{P}(\psi^{(J+1)}|I^{(1,1)}, \ldots, I^{(J,1)}) \tag{S23}$$

using the previously discussed approximations. Importance sampling was used to sample 36 new characters for each alphabet from $\bar{P}(\psi^{(J+1)}|\psi^{(1)}, \ldots, \psi^{(J)})$ using a base of 1000 samples. Given the new type $\psi^{(J+1)}$, a token of that type can be sampled from $P(\theta^{(J+1,1)}|\psi^{(J+1)})$. Finally, rather than directly sampling a stochastic image $P(I^{(J+1,1)}|\theta^{(J+1,1)})$ which results in noisy images, the binary image was rendered using the most likely value for each pixel. Sampled characters were also selected to be unique, meaning the model could not re-use the same set of strokes twice, and the concentration parameter α was set to 0.001 so that new characters were generated almost entirely from re-used variables. Also, since the human participants were instructed not to copy the example characters, BPL was not allowed to re-use all of the inferred strokes from any of the example characters. The character was centered such that its center of mass in trajectory space was in the center of the image canvas.

References and Notes

1. B. Landau, L. B. Smith, S. S. Jones, The importance of shape in early lexical learning. *Cogn. Dev.* **3**, 299–321 (1988). doi:10.1016/0885-2014(88)90014-7

2. E. M. Markman, *Categorization and Naming in Children* (MIT Press, Cambridge, MA, 1989).

3. F. Xu, J. B. Tenenbaum, Word learning as Bayesian inference. *Psychol. Rev.* **114**, 245–272 (2007). Medline doi:10.1037/0033-295X.114.2.245

4. S. Geman, E. Bienenstock, R. Doursat, *Neural Comput.* **4**, 1 (1992).

5. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998). doi:10.1109/5.726791

6. G. E. Hinton, L. Deng, D. Yu, G. Dahl, A.- Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.* **29**, 82–97 (2012). doi:10.1109/MSP.2012.2205597

7. A. Krizhevsky, I. Sutskever, G. E. Hinton, *Adv. Neural Inf. Process. Syst.* **25**, 1097–1105 (2012).

8. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. *Nature* **521**, 436–444 (2015). Medline doi:10.1038/nature14539

9. V. Mnih *et al.*, *Nature* 518, 529 (2015).

10. J. Feldman, The structure of perceptual categories. *J. Math. Psychol.* **41**, 145–170 (1997). Medline doi:10.1006/jmps.1997.1154

11. I. Biederman, Recognition-by-components: A theory of human image understanding. *Psychol. Rev.* **94**, 115–147 (1987). Medline doi:10.1037/0033-295X.94.2.115

12. T. B. Ward, Structured imagination: The role of category structure in exemplar generation. *Cognit. Psychol.* **27**, 1–40 (1994). doi:10.1006/cogp.1994.1010

13. A. Jern, C. Kemp, A probabilistic account of exemplar and category generation. *Cognit. Psychol.* **66**, 85–125 (2013). Medline doi:10.1016/j.cogpsych.2012.09.003

14. L. G. Valiant, A theory of the learnable. *Commun. ACM* **27**, 1134–1142 (1984). doi:10.1145/1968.1972

15. D. McAllester, in *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*, Madison, WI, 24 to 26 July 1998 (Association for Computing Machinery, New York, 1998), pp. 230–234.

16. V. N. Vapnik, An overview of statistical learning theory. *IEEE Trans. Neural Netw.* **10**, 988–999 (1999). Medline doi:10.1109/72.788640

17. N. D. Goodman, J. B. Tenenbaum, T. Gerstenberg, *Concepts: New Directions*, E. Margolis, S. Laurence, Eds. (MIT Press, Cambridge, MA, 2015).

18. Z. Ghahramani, Probabilistic machine learning and artificial intelligence. *Nature* **521**, 452–459 (2015). Medline doi:10.1038/nature14541

19. P. H. Winston, *The Psychology of Computer Vision*, P. H. Winston, Ed. (McGraw-Hill, New York, 1975).

20. T. G. Bever, D. Poeppel, *Biolinguistics* **4**, 174 (2010).

21. L. B. Smith, S. S. Jones, B. Landau, L. Gershkoff-Stowe, L. Samuelson, Object name learning provides on-the-job training for attention. *Psychol. Sci.* **13**, 13–19 (2002). Medline doi:10.1111/1467-9280.00403

22. R. L. Goldstone, in *Perceptual Organization in Vision: Behavioral and Neural Perspectives*, R. Kimchi, M. Behrmann, C. Olson, Eds. (Lawrence Erlbaum, City, NJ, 2003), pp. 233–278.

23. H. F. Harlow, The formation of learning sets. *Psychol. Rev.* **56**, 51–65 (1949). Medline doi:10.1037/h0062474

24. D. A. Braun, C. Mehring, D. M. Wolpert, Structure learning in action. *Behav. Brain Res.* **206**, 157–165 (2010). Medline doi:10.1016/j.bbr.2009.08.031

25. C. Kemp, A. Perfors, J. B. Tenenbaum, Learning overhypotheses with hierarchical Bayesian models. *Dev. Sci.* **10**, 307–321 (2007). Medline doi:10.1111/j.1467-7687.2007.00585.x

26. R. Salakhutdinov, J. Tenenbaum, A. Torralba, in *JMLR Workshop and Conference Proceedings*, vol. 27**,** *Unsupervised and Transfer Learning Workshop*, I. Guyon, G. Dror, V. Lemaire, G. Taylor, D. Silver, Eds. (Microtome, Brookline, MA, 2012), pp. 195–206.

27. J. Baxter, *J. Artif. Intell. Res.* **12**, 149 (2000).

28. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten Zip code recognition. *Neural Comput.* **1**, 541–551 (1989). doi:10.1162/neco.1989.1.4.541

29. R. Salakhutdinov, J. B. Tenenbaum, A. Torralba, Learning with hierarchical-deep models. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1958–1971 (2013). Medline doi:10.1109/TPAMI.2012.269

30. V. K. Mansinghka, T. D. Kulkarni, Y. N. Perov, J. B. Tenenbaum, *Adv. Neural Inf. Process. Syst.* **26**, 1520–1528 (2013).

31. K. Liu, Y. S. Huang, C. Y. Suen, Identification of fork points on the skeletons of handwritten Chinese characters. *IEEE Trans. Pattern Anal. Mach. Intell.* **21**, 1095–1100 (1999). doi:10.1109/34.799914

32. M.-P. Dubuisson, A. K. Jain, in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 1, Conference A: Computer Vision and Image Processing*, Jerusalem, Israel, 9 to 13 October 1994 (IEEE, New York, 1994), pp. 566–568.

33. G. Koch, R. S. Zemel, R. Salakhutdinov, paper presented at ICML Deep Learning Workshop, Lille, France, 10 and 11 July 2015.

34. M. Revow, C. K. I. Williams, G. E. Hinton, Using generative models for handwritten digit recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **18**, 592–606 (1996). doi:10.1109/34.506410

35. G. E. Hinton, V. Nair, *Adv. Neural Inf. Process. Syst.* **18**, 515–522 (2006).

36. S.-C. Zhu, D. Mumford, A stochastic grammar of images. *Foundations Trends Compu.t Graphics Vision* **2**, 259–362 (2006). doi:10.1561/0600000018

37. P. Liang, M. I. Jordan, D. Klein, in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 21 to 25 June 2010 (International Machine Learning Society, Princeton, NJ, 2010), pp. 639–646.

38. P. F. Felzenszwalb, R. B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **32**, 1627–1645 (2010). Medline doi:10.1109/TPAMI.2009.167

39. I. Hwang, A. Stuhlmüller, N. D. Goodman, http://arxiv.org/abs/1110.5667 (2011).

40. E. Dechter, J. Malmaud, R. P. Adams, J. B. Tenenbaum, in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, F. Rossi, Ed., Beijing, China, 3 to 9 August 2013 (AAAI Press/International Joint Conferences on Artificial Intelligence, Menlo Park, CA, 2013), pp. 1302–1309.

41. J. Rule, E. Dechter, J. B. Tenenbaum, in *Proceedings of the 37th Annual Conference of the Cognitive Science Society*, D. C. Noelle *et al*., Eds., Pasadena, CA, 22 to 25 July 2015 (Cognitive Science Society, Austin, TX, 2015), pp. 2051–2056.

42. L. W. Barsalou, Ad hoc categories. *Mem. Cognit.* **11**, 211–227 (1983). Medline doi:10.3758/BF03196968

43. J. J. Williams, T. Lombrozo, The role of explanation in discovery and generalization: Evidence from category learning. *Cogn. Sci.* **34**, 776–806 (2010). Medline doi:10.1111/j.1551-6709.2010.01113.x

44. A. B. Markman, V. S. Makin, Referential communication and category acquisition. *J. Exp. Psychol. Gen.* **127**, 331–354 (1998). Medline doi:10.1037/0096-3445.127.4.331

45. D. N. Osherson, E. E. Smith, On the adequacy of prototype theory as a theory of concepts. *Cognition* **9**, 35–58 (1981). Medline doi:10.1016/0010-0277(81)90013-5

46. S. T. Piantadosi, J. B. Tenenbaum, N. D. Goodman, Bootstrapping in a language of thought: A formal model of numerical concept learning. *Cognition* **123**, 199–217 (2012). Medline doi:10.1016/j.cognition.2011.11.005

47. G. A. Miller, P. N. Johnson-Laird, *Language and Perception* (Belknap, Cambridge, MA, 1976).

48. T. D. Ullman, A. Stuhlmuller, N. Goodman, J. B. Tenenbaum, in *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, Quebec City, Canada, 23 to 26 July 2014 (Cognitive Science Society, Austin, TX, 2014), pp. 1640–1645.

49. B. M. Lake, C.-y. Lee, J. R. Glass, J. B. Tenenbaum, in *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, Quebec City, Canada, 23 to 26 July 2014 (Cognitive Science Society, Austin, TX, 2014), pp. 803–808.

50. U. Neisser, *Cognitive Psychology* (Appleton-Century-Crofts, New York, 1966).

51. R. Treiman, B. Kessler, *How Children Learn to Write Words* (Oxford Univ. Press, New York, 2014).

52. A. L. Ferry, S. J. Hespos, S. R. Waxman, Categorization in 3- and 4-month-old infants: An advantage of words over tones. *Child Dev.* **81**, 472–479 (2010). Medline doi:10.1111/j.1467-8624.2009.01408.x

53. N. D. Goodman, T. D. Ullman, J. B. Tenenbaum, Learning a theory of causality. *Psychol. Rev.* **118**, 110–119 (2011). Medline doi:10.1037/a0021336

54. S. Dehaene, *Reading in the Brain* (Penguin, New York, 2009).

55. M. K. Babcock, J. J. Freyd, Perception of dynamic information in static handwritten forms. *Am. J. Psychol.* **101**, 111–130 (1988). Medline doi:10.2307/1422797

56. M. Longcamp, J. L. Anton, M. Roth, J. L. Velay, Visual presentation of single letters activates a premotor area involved in writing. *Neuroimage* **19**, 1492–1500 (2003). Medline doi:10.1016/S1053-8119(03)00088-0

57. K. H. James, I. Gauthier, Letter processing automatically recruits a sensory-motor brain network. *Neuropsychologia* **44**, 2937–2949 (2006). Medline doi:10.1016/j.neuropsychologia.2006.06.026

58. K. H. James, I. Gauthier, When writing impairs reading: Letter perception's susceptibility to motor interference. *J. Exp. Psychol. Gen.* **138**, 416–431 (2009). Medline doi:10.1037/a0015836

59. C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, D. Rasmussen, A large-scale model of the functioning brain. *Science* **338**, 1202–1205 (2012). Medline doi:10.1126/science.1225266

60. A. Graves, http://arxiv.org/abs/1308.0850 (2014).

61. K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, D. Wierstra, in *Proceedings of the International Conference on Machine Learning (ICML)*, Lille, France, 6 to 11 July 2015 (International Machine Learning Society, Princeton, NJ, 2015), pp. 1462–1471.

62. J. Chung *et al*., *Adv. Neural Inf. Process. Syst.* **28** (2015).

63. J. J. Goodnow, R. A. Levine, "The Grammar of Action": Sequence and syntax in childen's copying. *Cognit. Psychol.* **4**, 82–98 (1973). doi:10.1016/0010-0285(73)90005-4

64. B. M. Lake, R. Salakhutdinov, J. B. Tenenbaum, paper presented at the 34th Annual Conference of the Cognitive Science Society, Sapporo, Japan, 1 to 4 August 2012.

65. B. M. Lake, R. Salakhutdinov, J. B. Tenenbaum, paper presented at the Advances in Neural Information Processing Systems 26, City, Country, Dates Month Year.

66. N. D. Goodman, J. B. Tenenbaum, J. Feldman, T. L. Griffiths, A rational analysis of rule-based concept learning. *Cogn. Sci.* **32**, 108–154 (2008). Medline doi:10.1080/03640210701802071

67. T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, V. Mansinghka, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 7 to 12 Jule 2015 (IEEE, New York, 2015), pp. 4390–4399.

68. L. Lam, S.-W. Lee, C. Y. Suen, Thinning methodologies - A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **14**, 869–885 (1992). doi:10.1109/34.161346

69. C.-W. Liao, J. S. Huang, Stroke segmentation by bernstein-bezier curve fitting. *Pattern Recognit.* **23**, 475–484 (1990). doi:10.1016/0031-3203(90)90068-V

70. B. M. Lake, thesis, MIT, Cambridge, MA (2014).

71. O. Russakovsky *et al*., ImageNet large scale visual recognition challenge (2014), http://arxiv.org/abs/1409.0575.

72. Y. Jia *et al*., "Caffe: Convolutional Architecture for Fast Feature Embedding," in *Proceedings of the 22nd Annual ACM International Conference on Multimedia*, Orlando, FL, 3 to 7 November 2014 (ACM, New York, 2014), pp. 675–678.

73. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).

74. S. Chopra, R. Hadsell, Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, 20 to 26 June 2005 (IEEE, New York, 2005) vol. 1, pp. 539–546.

75. M. J. C. Crump, J. V. McDonnell, T. M. Gureckis, Evaluating Amazon's Mechanical Turk as a tool for experimental behavioral research. *PLOS ONE* **8**, e57410 (2013). Medline doi:10.1371/journal.pone.0057410

76. D. D. Hoffman, W. A. Richards, Parts of recognition. *Cognition* **18**, 65–96 (1984). Medline doi:10.1016/0010-0277(84)90022-2

77. T. S. Ferguson, A Bayesian analysis of some nonparametric problems. *Ann. Stat.* **1**, 209–230 (1973). doi:10.1214/aos/1176342360

78. J. Sethuraman, A constructive definition of dirichlet priors. *Stat. Sin.* **3**, 639–650 (1994).

79. N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, J. B. Tenenbaum, "Church: A language for generative models," in *Uncertainty in Artificial Intelligence* (2008), pp. 220–229.

80. T. J. O'Donnell, thesis, Harvard University, Cambridge, MA (2011).