Appendices for
Modeling rapid language learning by distilling Bayesian priors into artificial neural networks

R. Thomas McCoy and Thomas L. Griffiths

A Sampling formal languages

In the metalearning phase of inductive bias distillation, our model is shown many formal languages sampled from a distribution over formal languages. Here we describe the metagrammar that we use to define this distribution, where a metagrammar is a grammar that probabilistically generates grammars, each of which defines a formal language.

A.1 Producing a grammar

The space of grammars that the metagrammar generates is based on regular expressions but with two enhancements made to increase formal expressivity. First, the recursion primitive `plus` is modified to allow center embedding (see below). Second, the grammars use a mechanism called synchrony, which enables different components of a sequence to be aligned. This ability makes possible certain formal languages that are not possible with basic regular expressions (even ones augmented to allow center embedding), such as the language $A^nB^nC^n$. To sample a grammar, we follow these steps:

1.	Select a vocabulary size $\|V\|$, which is 1 plus a value sampled from geometric(0.5)—i.e., the minimum vocab size is 2. The vocabulary is then the first $\|V\|$ non-negative integers.	$\|V\| = 3$ $V = 0, 1, 2$
2.	Select a number of slots n from geometric(0.5)	$n = 4$
3.	Produce a synchrony pattern for these slots. The synchrony pattern is a list of length n where each value in the list is sampled uniformly from the integers from 1 to n inclusive. If two slots have the same value in the synchrony pattern, then they are synchronized.	[2, 1, 2, 3]
4.	For every unique value i in the synchrony pattern, use the context-free grammar in Table S1 to sample a value for $\mathrm{S}_i[k]$, where k is the number of slots with that synchrony value. Note that the CFG contains 2 sets of probabilities: *probability at top* is used for selecting the way that the root node of a slot expands, while *probability elsewhere* is used otherwise.	$\mathrm{S}_1[1]$ = plus(2 2, 0.35, 1) $\mathrm{S}_2[2]$ = concat(0/1 Σ 0/3) $\mathrm{S}_3[1]$ = concat(or(0, 0))

The value for each $\mathrm{S}_i[k]$ is built from several primitives:

- Terminal: A k-tuple of words (where k is the number of synchrony slots that the terminal is used in).
- Σ: Can expand to any token in the vocabulary, sampled uniformly. Note that, like terminals, Σ also technically produces a k-tuple, but all of the tokens are guaranteed to be identical within any k-tuple sampled from Σ.

- ϵ: The empty string
- or(A, B): Randomly choose A or B, each with probability 0.5.
- plus(A B C ..., p, i): The first argument is a list of expressions. This list is repeated a number of times sampled from geometric(p). Each new instance is placed at index i in the list, where i could be at the start of the list or in between any two elements.
  - i is chosen uniformly from the possible options.
  - p is drawn from normal(0.33, 0.05); p is capped at 0.15 below and 0.99 above. p can be thought of as one minus the probability of recursion—so the default case uses 0.67 as the probability of recursion. The high average probability was chosen to ensure that the model gets sufficient experience with deep recursion.
  - *Example:* plus(A C, 0.5, 2) would yield tail recursion: ACACAC
  - *Example:* plus(A C, 0.5, 1) would yield center embedding: AAACCC
  - The length of the list in the first argument is drawn from geometric(0.5). This is what is achieved by the rules for the nonterminal X in Table S1.
- concat(A B C...): This concatenates a list of expressions. The list might have a length of just 1. The length of the list is drawn from geometric(0.5).

The result of sampling from an expression will be a list of n-tuples of words, where n is the number of slots that depend on this expression. The string in the i^{th} slot that depends on the expression is created by concatenating the i^{th} elements of all these n-tuples.

For example, consider the grammar created in Section A.1. We might sample the following values for each of the 3 expressions:

- S_1 = 2 2 2 2 2 2
- S_2 = 0/1 1/1 0/3
- S_3 = 0

The two strings produced for S_2 would then be 0 1 0 and 1 1 3. Since the synchrony pattern is [2, 1, 2, 3], these strings would be placed in positions 0 and 2, respectively; while position 1 would get the string from S_1, and position 3 would get the string from S_3. Concatenating it all together gives us our final string: 0 1 0 2 2 2 2 2 2 1 1 3 0.

A.2 Additional comments on the metagrammar

Minimum vocabulary size: The vocabulary size is drawn from a geometric distribution. Therefore, if we allowed a vocabulary size of 1, half of the languages would have that vocabulary size, giving little variety to the space of generated languages. To avoid this issue, we set the minimum vocabulary size to be 2. Note however that a language does not necessarily use its whole vocabulary, so it is still possible to have a language with a vocabulary size of 1; this situation is just not as pervasive as it would be if the minimum vocabulary size were 1.

Probability at the top vs. elsewhere: There are two competing concerns in designing the metagrammar. On one hand, we want every primitive to be common enough that our models can meta-learn it. On the other hand, some primitives have multiple children, which can prevent the grammar's production process from terminating properly if those primitives have high probability (Chi, 1999). As a simple example, consider the grammar given by the two rules below. If the first rule has a probability greater than 0.5, then the grammar often fails to terminate; if its probability is less than 0.5, then it is guaranteed to terminate.

(1) $S \rightarrow SS$

(2) $S \rightarrow a$

To address this issue, when a rule has many nonterminals on its righthand side, we gave it a high probability at the top of the tree (to guarantee that this rule is used reasonably frequently) but a low probability elsewhere (to avoid runaway recursion). Specifically, we set the probabilities in the first pane of Table S1 using the following formula, where p is the probability for the rule, k is the arity of the rule (the number of nonterminals on its righthand side), θ is a parameter that we set at 0.25, and α indicates whether we are determining the probability for *top* ($\alpha = -1.0$) or *elsewhere* ($\alpha = 1.0$):

$$p \propto \theta^{\alpha k} \tag{1}$$

The arities (the k values) are 0 for terminals and 2 for the other primitives. (For `plus` and `concat`, this value of 2 is an expected value, since the arity varies according to geometric(0.5) for these cases).

Extending expressivity beyond regular languages: The index argument in the `plus` primitive enables center embedding, which is not possible with the basic Kleene plus mechanism. The synchrony mechanism enables some context-sensitive structures that are not possible with just context-free grammars.

A.3 Withholding

During the inductive bias distillation process, we withheld all formal languages that are part of the formal language evaluation set. One challenge in performing such withholding is that two formal languages might have different descriptions yet be extensionally equivalent; for example, `concat(A, concat(A,B))` is equivalent to `concat(concat(A,A), B)`. With this concern in mind, the way that we performed withholding was based on comparing stringsets sampled from languages, rather than comparing language descriptions. Specifically, for each formal language sampled during the distillation process, we sampled 100,000 strings from that language. We then considered each evaluation language in turn; for each one, we sampled 100,000 strings from that evaluation language and computed the F-score between the stringsets sampled from the evaluation language and from the candidate distillation language. If the F-score was 1.0, then the language

Rule	Probability at top	Probability elsewhere
S[k] $\rightarrow$ TERMINAL$_k$	0.02	0.84
S[k] $\rightarrow$ or(S[k], S[k])	0.33	0.05
S[k] $\rightarrow$ plus(S[k] X[k], PROB, INDEX)	0.33	0.05
S[k] $\rightarrow$ concat(S[k] X[k])	0.33	0.05
TERMINAL$_k$ $\rightarrow$ WORD$_1$/WORD$_2$/.../WORD$_k$	N/A	$\frac{k}{k+2}$
TERMINAL$_k$ $\rightarrow$ Σ	N/A	$\frac{1}{k+2}$
TERMINAL$_k$ $\rightarrow$ ϵ	N/A	$\frac{1}{k+2}$
WORD $\rightarrow$ w_1	N/A	$\frac{1}{\lvert V\rvert}$
WORD $\rightarrow$ w_2	N/A	$\frac{1}{\lvert V\rvert}$
...		
WORD $\rightarrow$ $w_{\lvert V\rvert}$	N/A	$\frac{1}{\lvert V\rvert}$
X[k] $\rightarrow$ ϵ	N/A	0.5
X[k] $\rightarrow$ S[k] X[k]	N/A	0.5

Table S1: Context-free grammar used to sample expressions that are inserted into synchrony patterns. Due to rounding, some probabilities in the table do not add to 1.0.

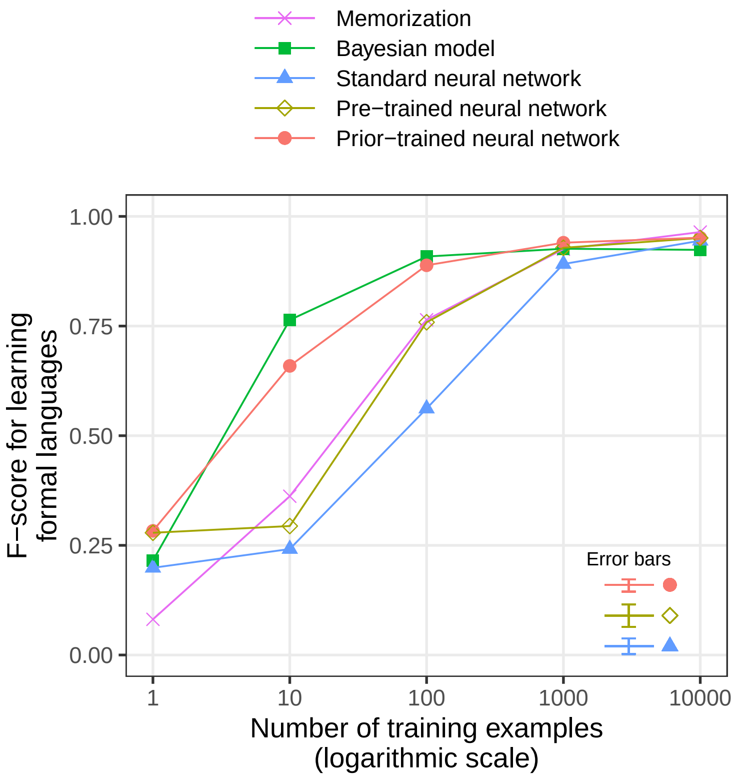


Figure S1: Expanded version of Figure 2 from the main paper, showing the ability of various models to learn formal languages. This plot averages over the 56 formal languages used by Yang and Piantadosi (2022).The Bayesian model results are taken from Yang and Piantadosi. The prior-trained neural network is our model that has undergone inductive bias distillation; the standard neural network has the same architecture but has not undergone distillation, while the pre-trained neural network has been trained on the same distillation data as the prior-trained network but using standard learning instead of meta-learning. The memorization baseline shows the performance that would be achieved by simply memorizing the training set. For each neural network condition, the plot shows the mean over 40 re-runs, with the bottom right showing error bars (giving the full range) averaged over the five training set sizes.

was withheld from distillation. The F-score used for this purpose was defined in the same way as in the Methods section of the main paper, considering only the 25 highest-probability strings from each sampled stringset, and we found that sampling 100,000 strings was generally sufficient for providing the necessary 25 distinct highest-probability strings.

Due to the probabilistic nature of this process, it is possible that the F-score-based withholding could have missed a formal language that should have been withheld. Therefore, as an additional measure, we confirmed that the set of training examples used in each episode of meta-learning was never the same as the set of training examples used for any evaluation language. Thus, we can be confident that the prior-trained network was never trained on any of the specific learning scenarios that it is being evaluated on.

B Pre-training and memorization results for formal languages

Figure S1 expands on Figure 2 from the main paper by adding two additional conditions: a memorization baseline (showing the performance that would be achieved by assigning to each string a probability equal to its frequency in the training set) and a baseline of pre-training (giving a neural network the same data that we gave to our prior-trained network, but having it learn using standard learning rather than meta-

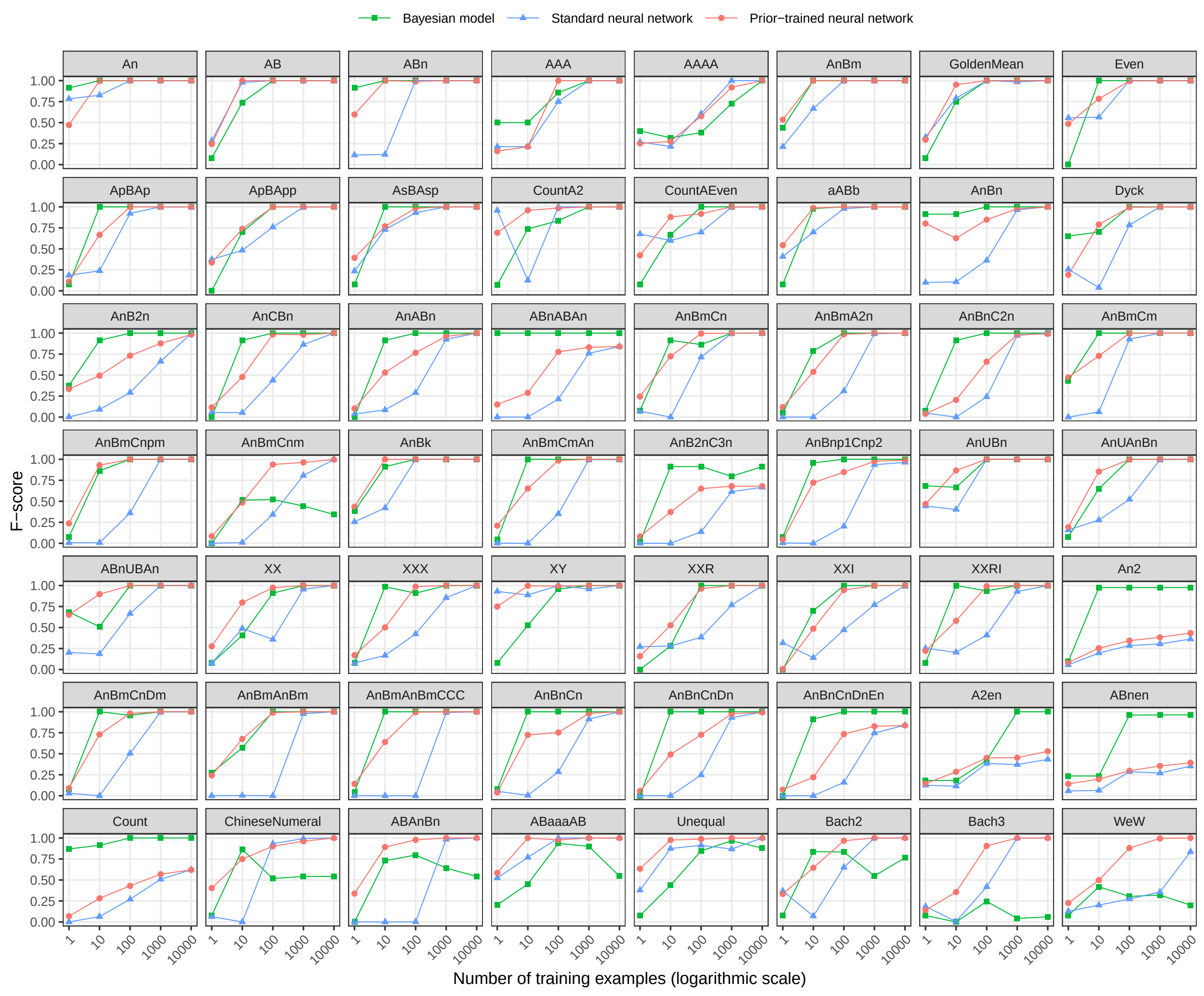


Figure S2: Formal language results broken down by formal language for the three main approaches we consider (the Bayesian model, the standard neural network, and the prior-trained neural network). Each point is the mean over 40 re-runs.

learning). The pre-trained network outperforms the standard network but still scores substantially worse than the prior-trained network.

C Fine-grained formal language results

Figure S2 shows the formal language results broken down by formal language (Figure 2 in the main paper gives these results averaged across languages). Figure S3 shows the same results but with two additional approaches included (the memorization baseline and the pre-training approach).

D Details of time comparisons

The main paper lists the training times required by a prior-trained model and a Bayesian model to learn formal languages. These times denote how long it takes to learn one language *after* inductive bias distillation is complete. To be clear, the distillation process is relatively slow, requiring approximately 2 days. We do not include this period in our time calculation because we only intend to model how languages are learned,

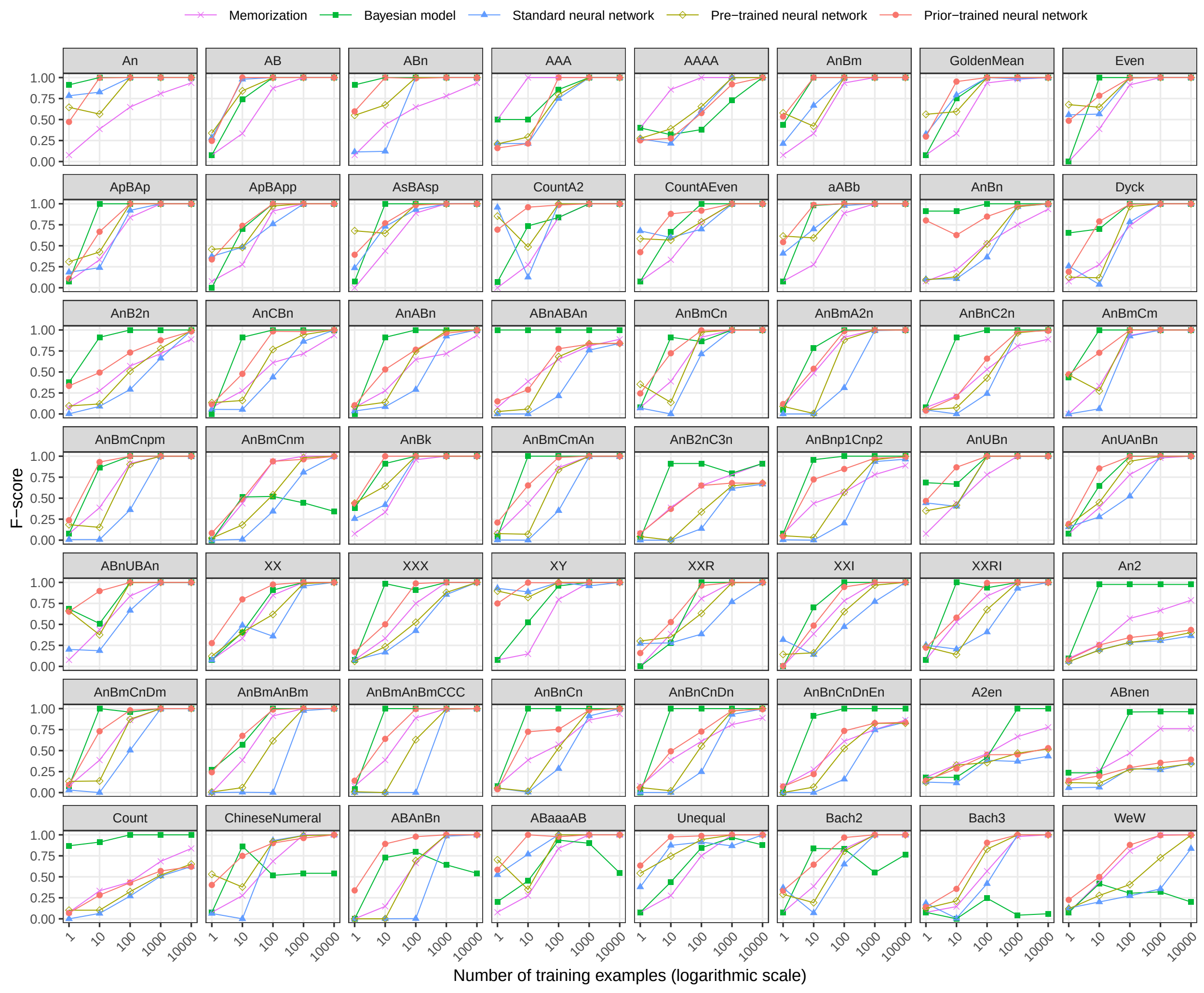


Figure S3: Formal language results broken down by formal language for an extended set of approaches: the three approaches shown in Figure S2 plus the memorization baseline and the prior-training approach. Each point is the mean over 40 re-runs.

not how inductive biases are acquired. Therefore, the distillation process is not part of the model but is rather a tool used to create the model. Even if this time were included, however, it remains less than the Bayesian learner's maximum time of 7 days. Further, it is a single up-front cost that only needs to be paid once, after which the same model can be reused for any number of languages; thus, summing across all 56 languages, the prior-trained neural network takes a total of about 2 days, compared to about 129 days for the Bayesian model. We also do not include the time it takes to sample strings from our model for the purpose of computing the F-score, because this step is not something that the model must do to learn but rather is something that we as analysts must do to figure out what the model has learned. This sampling takes from 1 to 15 minutes per language.

E Details of hyperparameter searches

E.1 Hyperparameter search for training on formal languages

For adapting our prior-trained model to a formal language, we used three stages: (i) training it using stochastic gradient descent for i epochs; (ii) training it using Adam for j epochs; and (iii) sampling sequences from the trained model using a value of p for top-p sampling and a temperature t (see main text for a description of top-p sampling and temperature). We searched over all possible combinations the following hyperparameter choices:

- Number of stochastic gradient descent epochs: 5, 10
- Number of Adam epochs: 1, 5
- p for use in top-p sampling: 0.99, 1.0
- Temperature: 0.5, 1.0

For each neural network type (standard, pre-trained, and prior-trained), we tuned the hyperparameters using a validation set of 20 formal languages that were not present in the 56-language test set from Yang & Piantadosi that we evaluated on in Figure 2 of the main paper. These 20 validation languages can be found on the project GitHub page: `https://github.com/tommccoy1/inductive-bias-distillation`. We used the same hyperparameters for all languages, only varying the hyperparameters across training set sizes. That is, we had 5 hyperparameter settings for each model type, one for each training set size:

- Standard neural network:
  - Training size 1: SGD epochs = 10, Adam epochs = 1, $p = 1.0$, temperature = 0.5
  - Training size 10: SGD epochs = 10, Adam epochs = 1, $p = 0.99$, temperature = 0.5
  - Training size 100: SGD epochs = 10, Adam epochs = 5, $p = 0.99$, temperature = 0.5
  - Training size 1,000: SGD epochs = 10, Adam epochs = 5, $p = 0.99$, temperature = 1.0
  - Training size 10,000: SGD epochs = 10, Adam epochs = 5, $p = 0.99$, temperature = 0.5
- Pre-trained neural network:
  - Training size 1: SGD epochs = 10, Adam epochs = 5, $p = 1.0$, temperature = 1.0
  - Training size 10: SGD epochs = 10, Adam epochs = 5, $p = 0.99$, temperature = 1.0
  - Training size 100: SGD epochs = 10, Adam epochs = 5, $p = 0.99$, temperature = 0.5
  - Training size 1,000: SGD epochs = 5, Adam epochs = 5, $p = 0.99$, temperature = 0.5
  - Training size 10,000: SGD epochs = 5, Adam epochs = 5, $p = 0.99$, temperature = 0.5
- Prior-trained neural network:
  - Training size 1: SGD epochs = 5, Adam epochs = 1, $p = 1.0$, temperature = 0.5

  - Training size 10: SGD epochs = 5, Adam epochs = 1, $p = 1.0$, temperature = 0.5
  - Training size 100: SGD epochs = 10, Adam epochs = 5, $p = 0.99$, temperature = 0.5
  - Training size 1,000: SGD epochs = 10, Adam epochs = 5, $p = 0.99$, temperature = 0.5
  - Training size 10,000: SGD epochs = 5, Adam epochs = 5, $p = 0.99$, temperature = 0.5

E.2 Hyperparameter search for training on CHILDES data

The performance of neural networks is highly sensitive to hyperparameter choices. Our goal in the CHILDES experiments was to test whether there are differences in the learning abilities of a standard neural network and a prior-trained neural network, but we did not want any observed differences to be the result of choosing hyperparameters that happen to be better-suited for one approach over the other. To address this concern, we performed an extensive hyperparameter search for both model types and used the hyperparameters that worked best for each model type. We chose this method as the way to be maximally charitable to both approaches. The search that we performed was identical for both model types, to ensure fairness (i.e., it would have been unfair if we tried more hyperparameter combinations for one type than the other).

We first considered only training models on the full dataset, not fractions of it. For this purpose, we trained models with all seven hidden layer sizes that we considered (16, 32, 64, 128, 256, 512, and 1024). For each size, we tried all possible combinations of the following hyperparameters (note that we search over the number of epochs, rather than fixing this number at the maximum value, because our use of a cosine learning rate scheduler means that a greater number of epochs is not guaranteed to improve performance):

- **Learning rate:** 0.01, 0.005, 0.001, 0.0005
- **Dropout:** 0.1, 0.2, 0.4
- **Epochs:** 8, 16, 32, 64, 128

We then expanded the search to include all fractions of the dataset shown in Figure 3B of the main paper, such that we conducted a hyperparameter search for each cell in the table. We searched over the same hyperparameter possibilities listed above, except for two changes:

- Epochs: We eliminated the options of 8 and 16 epochs, as these never performed the best in the initial search. We also added an option of $128/f$ epochs, where f is the fraction of the full dataset that is being trained on, in case the important number was the number of parameter updates rather than the number of epochs (varying the training set fraction causes these numbers to dissociate). For example, when training on one eighth of the full dataset, we included the option of 1024 epochs (128 divided by one eighth).
- Dropout: In the initial search, we found that certain dropout values always worked the best for certain model sizes. Therefore we restricted the dropout options by model size. For sizes 16, 32, 64, and 128, the only dropout option was 0.1; for size 256, the two options were 0.1 and 0.2; for size 512, the two options were 0.2 and 0.4; and for size 1024, the only option was 0.4.

We then identified the best hyperparameter setting for the standard model (call these hyperparameters h_s) and for the prior-trained model (h_p). We then performed 20 re-runs for both model types using both h_s and h_p. That is, we tried both h_s and h_p for the standard model, and tried both h_s and h_p for the prior-trained model. For each model type, our final hyperparameter choice—the one whose results are reported in Figure 3 of the main paper—is the one that gave the lowest mean perplexity for that model type. (We tried both h_s and h_p for both model types as an additional guarantee of fairness. We wanted to avoid a scenario where, e.g., h_p is actually better in general than h_s for both models, but h_s just happened to work better for the standard model in the run done during the hyperparameter search. Doing the full set of runs for both models with both settings addresses this possibility.) In many cases h_s and h_p were identical, meaning that the choice between these options did not need to be made. The final hyperparameter choices are shown in Table S2 for standard neural networks and Table S3 for prior-trained neural networks.

To produce different fractions of the full dataset, we divided the full dataset into 1025 pieces (1024 equal-sized pieces plus a remainder). We then sampled without replacement a number of these pieces needed

	Hidden 16	Hidden 32	Hidden 64	Hidden 128	Hidden 256	Hidden 512	Hidden 1024
1/64	0.1,0.01,8192	0.1,0.01,8192	0.1,0.005,128	0.1,0.005,64	0.2,0.001,128	0.4,0.001,32	0.4,0.0005,32
1/32	0.1,0.01,4096	0.1,0.005,4096	0.1,0.01,128	0.1,0.01,32	0.2,0.005,32	0.4,0.001,32	0.4,0.0005,32
1/16	0.1,0.01,2048	0.1,0.01,2048	0.1,0.005,2048	0.1,0.0005,2048	0.2,0.001,128	0.4,0.001,32	0.4,0.0005,32
1/8	0.1,0.01,1024	0.1,0.01,1024	0.1,0.01,1024	0.1,0.01,128	0.2,0.01,128	0.4,0.001,32	0.4,0.0005,32
1/4	0.1,0.01,512	0.1,0.01,512	0.1,0.01,512	0.1,0.01,512	0.2,0.01,512	0.4,0.005,128	0.4,0.005,32
1/2	0.1,0.005,256	0.1,0.01,256	0.1,0.01,256	0.1,0.01,256	0.2,0.005,256	0.4,0.005,256	0.4,0.005,32
Full	0.1,0.005,128	0.1,0.005,128	0.1,0.005,128	0.1,0.005,128	0.1,0.005,128	0.2,0.005,128	0.4,0.001,32

Table S2: Best hyperparameter values for training standard neural networks on CHILDES data. Each cell shows three hyperparameters: dropout, learning rate, and number of epochs (in that order). Each row shows models trained on a different proportion of the training set.

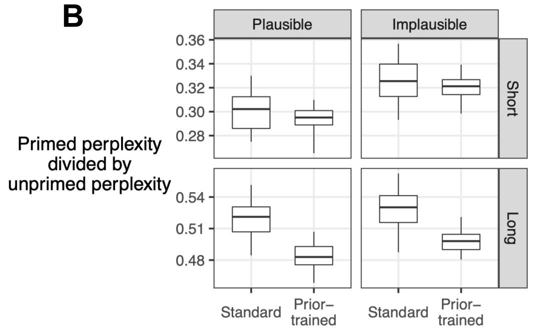
	Hidden 16	Hidden 32	Hidden 64	Hidden 128	Hidden 256	Hidden 512	Hidden 1024
1/64	0.1,0.01,8192	0.1,0.0005,8192	0.1,0.005,8192	0.1,0.001,128	0.2,0.001,128	0.4,0.01,32	0.4,0.01,32
1/32	0.1,0.01,4096	0.1,0.001,4096	0.1,0.0005,4096	0.1,0.001,128	0.2,0.0005,128	0.4,0.005,64	0.4,0.005,128
1/16	0.1,0.01,2048	0.1,0.01,2048	0.1,0.0005,2048	0.1,0.001,128	0.2,0.001,128	0.4,0.001,128	0.4,0.005,32
1/8	0.1,0.01,1024	0.1,0.01,1024	0.1,0.01,1024	0.1,0.01,64	0.2,0.005,32	0.4,0.001,128	0.4,0.001,64
1/4	0.1,0.01,512	0.1,0.005,512	0.1,0.01,512	0.1,0.01,128	0.2,0.01,128	0.4,0.001,128	0.4,0.001,64
1/2	0.1,0.005,256	0.1,0.01,256	0.1,0.01,256	0.1,0.01,256	0.2,0.005,256	0.4,0.005,256	0.4,0.0005,128
Full	0.1,0.005,128	0.1,0.005,128	0.1,0.005,128	0.1,0.005,128	0.2,0.005,128	0.2,0.005,128	0.4,0.0005,64

Table S3: Best hyperparameter values for training prior-trained neural networks on CHILDES data. Each cell shows three hyperparameters: dropout, learning rate, and number of epochs (in that order). Each row shows models trained on a different proportion of the training set.

A

	Standard accuracy	Prior-trained accuracy	Significance
Zorro	0.784	0.798	p < 10e-10
BLiMP$_{CH}$	0.663	0.675	p < 10e-23
SCaMP: Plausible	0.740	0.765	p < 10e-15
SCaMP: Implausible	0.718	0.748	p < 10e-17

B



C

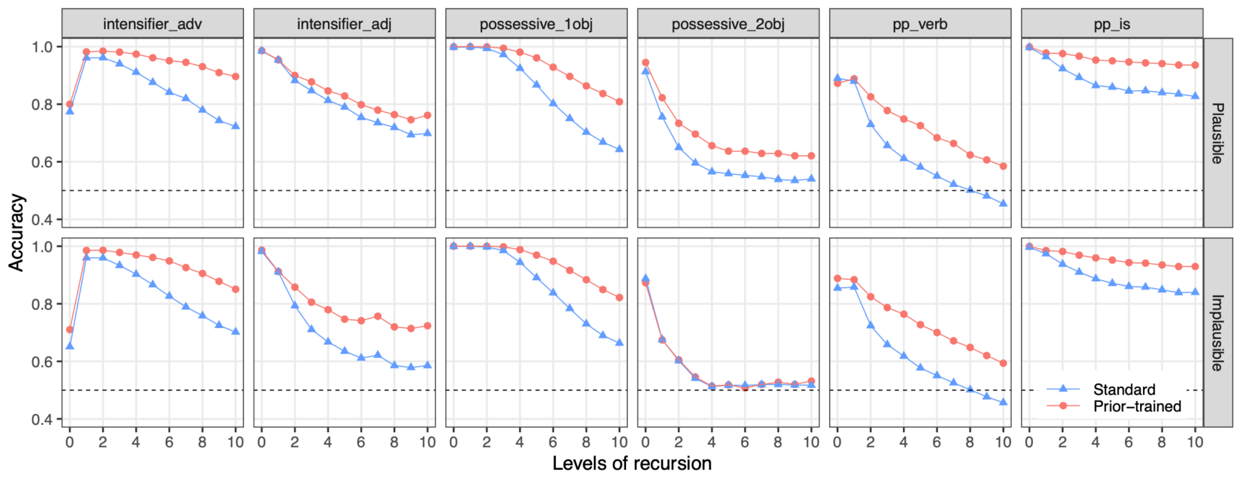


Figure S4: Reproducing Figure 4 from the main paper but using the same hyperparameters for the standard and prior-trained networks. **A.** Accuracy on 4 minimal pair datasets that each cover a broad range of syntactic phenomena. **B.** The extent to which models display priming in sentences that are either short or long and either semantically plausible or semantically implausible. The lower the value on the y-axis is, the more extensively priming has occurred. **C.** Evaluations of recursion.

to produce each fraction (for example, 256 pieces to produce half of the dataset); the remainder portion was excluded for all fractions except the full dataset. When running multiple reruns, we performed this sampling differently for each re-run, but we used the same datasets for standard and prior-trained models. For example, the seventh run with the standard model and the seventh run with the prior-trained model used exactly the same training set, but the seventh run for each one had a different training set from the sixth run of each one.

F Targeted linguistic evaluation results with different hyperparameters

As discussed in Section E.2, the best standard neural network and best prior-trained neural network were trained on CHILDES data with different hyperparameters, where *best* is defined as having the lowest perplexity. Figure 4 in the main paper shows additional results for how these systems perform on targeted linguistic evaluations, with the general finding being that the prior-trained network outperforms the standard network. Since these two systems were trained with different hyperparameters, a potential concern with this finding is that the difference may be due to their different hyperparameters rather than the presence or absence of a distilled inductive bias. To address this concern, we trained a standard neural network with the same hyperparameters as the prior-trained neural network. Figure S4 compares these new standard networks to the same prior-trained networks as before. The results are qualitatively the same as in the main paper; the main differences are that the prior-trained network's advantage on minimal pairs (Figure S4A) is slightly increased, while its advantage in priming (Figure S4B) is decreased. In some informal preliminary experiments, we also tested some smaller-scale models than the main ones we focused on (i.e., models with a hidden layer size less than 1024 and/or a training set fraction less than 1.0); we found that, in these settings, the prior-trained network outperformed the standard network less consistently on the targeted linguistic evaluations than in the larger-scale case that we focus on (where the hidden layer size is 1024 and the full training set is used). This difference may have the same cause as the diagonal-stripe pattern shown in Figure 2B of the main paper, in which the distilled inductive bias is only helpful at certain scales.

G Results for individual minimal pair phenomena

In our minimal pair evaluations, there were some phenomena for which the prior-trained model substantially outperformed the standard model and others where the reverse was true. For instance, on the collection of pairs targeting the fact that *ever* can occur in questions such as (3a) but not basic declaratives such as (3b), the prior-trained network gets an accuracy of 71.0%, compared to 57.7% for the standard network. In the other direction, the models' performance is flipped on examples targeting the fact that certain syntactic configurations are grammatical when there is a "gap"—an empty slot, such as the lack of a direct object following *stole* in (4a)—but are ungrammatical without a gap, as in (4b). On one sub-dataset containing such pairs, the standard network gets an accuracy of 63.3%, compared to 54.3% for the prior-trained network.

(3) a. Have the hats ever moved?
b. *The hats have ever moved.

(4) a. This lady won't see the ship that the teachers stole.
b. *This lady won't see what the teachers stole the ship.

These minimal pair results highlight the important fact that inductive biases are not inherently positive or negative. They simply guide how a learner generalizes, and this guidance could point a learner in the correct direction or in an incorrect direction. Overall, however, the inductive bias that we have distilled appears to be more helpful than harmful: when all phenomena are aggregated together, the prior-trained model scores statistically significantly better than the standard model on all four of the minimal pair datasets that we included.

The tables provided in the project GitHub page at `https://github.com/tommccoy1/inductive-bias-distillation/blob/main/supplement/inductive_bias_distillation_minimal_pairs.pdf` show the results for all individual phenomena within the four minimal pair evaluation sets that we used.

H Examples of individual recursion and priming categories

The recursion results in Figure 4C of the main paper involve six types of examples, each of which can be semantically plausible or semantically implausible. Below are example minimal pairs (with two levels of recursion) for each of the twelve combinations of a category and a plausibility setting. The (a) and (b) examples show the grammatically correct and grammatically incorrect member of each minimal pair, respectively. A model was deemed correct on a given minimal pair if it assigned a higher probability to the underlined portion of the grammatical sentence than the underlined portion of the ungrammatical sentence.

(5) **intensifier_adv: plausible**
a. the lady waves really really carefully .
b. *the lady is really really carefully .

(6) **intensifier_adv: implausible**
a. the box talks very very quickly .
b. *the box is very very quickly .

(7) **intensifier_adj: plausible**

a. the king was very very clever .
b. *the king cries very very clever .

(8) **intensifier_adj: implausible**

a. the bottle is really really bored .
b. *the bottle paints really really bored .

(9) **possessive_1obj: plausible**

a. the dancer thinks that my father 's doctor 's mother is tall .
b. *the dancer scares my father 's doctor 's mother is tall .

(10) **possessive_1obj: implausible**

a. the shoe thinks that my wife 's friend 's glass is helpful .
b. *the shoe talks to my wife 's friend 's glass is helpful .

(11) **possessive_2obj: plausible**

a. the person gave your wife 's teacher 's mom the rug .
b. *the person moved your wife 's teacher 's mom the rug .

(12) **possessive_2obj: implausible**

a. the window bought my cousin 's friend 's crayon the chair .
b. *the window shook my cousin 's friend 's crayon the chair .

(13) **pp_verb: plausible**

a. the book sitting on the stool in the mall is old .
b. *the book sits on the stool in the mall is old .

(14) **pp_verb: implausible**

a. the button sitting by the ship in the mall is gentle .
b. *the button sits by the ship in the mall is gentle .

(15) **pp_is: plausible**

a. the picture in the kitchen in the museum is small .
b. *the picture was in the kitchen in the museum is small .

(16) **pp_is: implausible**

a. the hat in the library in the school is excited .
b. *the hat was in the library in the school is excited .

The priming results in Figure 4B of the main paper involve two sentence lengths (short and long), each of which comes in both a semantically plausible version and a semantically implausible version. Below are examples for each of the four combinations of a length and a plausibility setting. The (a) examples show unprimed cases, and the (b) examples show primed cases. The numbers in Figure 4B were calculated by dividing the perplexity assigned to the underlined portion of each primed example by the perplexity assigned to the underlined portion of each unprimed example.

(17) **short: plausible**

a. that person can teach those doctors in the library .
b. that person can teach those doctors in the library . that person can teach those doctors in the library .

(18) **short: implausible**

a. on the table the bag is helping those cups .
b. on the table the bag is helping those cups . on the table the bag is helping those cups .

(19) **long: plausible**

a. in the library that guy has helped the adult , and those boys by the window had scared the driver .
b. in the library that guy has helped the adult , and those boys by the window had scared the driver . in the library that guy has helped the adult , and those boys by the window had scared

the driver .

(20) **long: implausible**

a. those shoes by the mirror may scare the crayon , and those forks have worried the toys in the restaurant .

b. those shoes by the mirror may scare the crayon , and those forks have worried the toys in the restaurant . those shoes by the mirror may scare the crayon , and those forks have worried the toys in the restaurant .

References

Zhiyi Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160.

Yuan Yang and Steven T Piantadosi. 2022. One model for the learning of language. *Proceedings of the National Academy of Sciences*, 119(5).