A MINIMALIST DATASET FOR SYSTEMATIC GENERALIZATION OF PERCEPTION, SYNTAX, AND SEMANTICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Inspired by humans' remarkable ability to master arithmetic and generalize to unseen problems, we present a new dataset, Handwritten arithmetic with INTegers (HINT), to study machines' capability of learning generalizable concepts at three levels: *perception*, *syntax*, and *semantics*. In HINT, machines are tasked to learn how concepts are perceived from raw signals such as images (*i.e.*, perception), how multiple concepts are structurally combined to form a valid expression (*i.e.*, syntax), and how concepts are realized to afford various reasoning tasks (*i.e.*, semantics), all in a weakly supervised manner. With a focus on systematic generalization, we carefully design a five-fold test set to evaluate both the *interpolation* and the *extrapolation* of learned concepts w.r.t. the three levels. We further design a few-shot learning split to test whether models can quickly learn new concepts and generalize them to more complex scenarios. To understand existing models' limitations, we conduct extensive experiments with various sequence-to-sequence models, including RNNs, Transformers, and GPT-3 (with the chain of thought prompting). The results suggest that current models still struggle in extrapolation to long-range syntactic dependency and semantics. Models show a significant gap toward human-level generalization when tested with new concepts in a few-shot setting. Moreover, we find that it is infeasible to solve HINT by simply scaling up the dataset and the model size; this strategy barely helps extrapolate over syntax and semantics. Finally, in zero-shot GPT-3 experiments, the chain of thought prompting shows impressive results and significantly boosts the test accuracy. We believe the proposed dataset and the experimental findings are of great interest to the learning community on systematic generalization.[1]

1 INTRODUCTION

Humans possess a versatile mechanism for learning concepts from data (Firestone & Scholl, 2016). For instance, imagine that we were tasked to decipher ancient Egyptian signs from some examples as depicted in Tab. 1. Given enough time, we can understand these signs by not only knowing how to recognize them—what each sign looks like at the *perceptual* level, but also knowing how to compose them into valid sequence—at the *syntactic* level, and how to predict the results—at the *semantic* level. Learning concepts heavily rely on these three-level interweaving meanings. Such observation also conforms with the classic view of human cognition, which postulates at least three distinct levels of organizations in computation systems (Pylyshyn, 1984).

Another appealing property of human concept learning is its *systematic compositionality* (Chomsky, 1957; Montague, 1970): the algebraic capacity to understand and produce a potentially infinite number of novel combinations from a finite set of known components, *i.e.*, "infinite use of finite means" (Chomsky, 1965). This type of compositionality is crucial to the human ability to make strong generalizations from simple examples to complex ones, as shown in Tab. 1.

The emerging community on learning models capturing human-like systematic compositionality has introduced various benchmarks (Lake & Baroni, 2018; Hupkes et al., 2020; Keysers et al., 2020) and methods (Lake, 2019; Gordon et al., 2019; Csordás et al., 2021). However, as collecting real data with systematic compositionality is challenging, most existing benchmarks are from artificial domains with synthetic data and tasks and only cover a partial spectrum of concept learning; see

[1] We release our dataset, code, and experiments at an anonymous website https://hint-iclr.github.io/HINT/.

Train		Test
𓂝𓂋𓏲𓂝𓂋𓂋𓂋 → 60	𓂋𓂝𓂋𓂋𓏲𓂝𓂝𓂝 → 18	𓂝𓂋𓂝𓂋𓏲𓏲𓏲𓂋𓂝𓂝𓂝 → ?
𓂝𓂋𓏲𓂝𓂝𓏲𓂝𓂝𓂝𓂋 → 100	𓂝𓂋𓂝𓂋𓏲𓏲𓏲𓂝𓂝𓂝 → 16	𓂝𓂝𓂝𓂋𓂝𓏲𓏲𓂝𓂝𓂝 → ?
𓂝𓂋𓏲𓂝𓂝𓂝 → 12	𓂝𓂋𓂝𓂋𓂋𓂋𓂝𓂝𓂝𓂝 → 41	𓂋𓂋𓂝𓂋𓂋𓂋𓂝𓂋𓂋𓂋𓂝𓂝𓂝 → ?
𓂝𓂋𓂝𓂋𓏲𓂝𓂝𓂝𓂝 → 4	𓂋𓂋𓂝𓂋𓂋𓂋𓂝𓂝𓂝𓂝 → 4	𓂝𓂋𓂝𓂋𓂋𓂋𓂝𓂋𓂋𓂋𓂝𓂝𓂝𓂝𓂝𓂝𓂝 → ?
𓂝𓂋𓂝𓂋𓂋𓂋𓂝𓂋𓂋𓂋 → 26	𓂝𓂋𓂝𓂋𓂋𓂋𓂝𓂋𓂋𓂋 → 17	𓂝𓂋𓂝𓂋𓂋𓂋𓂝𓂋𓂋𓂋𓂝𓂋𓂋𓂋𓂝𓂋𓂋𓂋𓂝𓂝𓂝𓂝 → ?

Table 1: **Can you decipher these ancient Egyptian signs from training examples and apply them to test cases?** Check https://hint-iclr.github.io/HINT/egypt.html for more training and test samples with the ground-truth meaning for each sign. We strongly encourage the readers to try this game before checking the answers.

Tab. 2 for an in-depth comparison. Critically, prior datasets often mix syntax and semantics when evaluating systematic compositionality. For example, the SCAN dataset (Lake & Baroni, 2018) is a semantic parsing task from natural language commands to action sequences; when a model fails on a longer command than the ones in the training set, the root cause could come from misunderstanding the complex syntactic relations in a long input sequence (command) or its deficiency of generating a long output sequence (actions) (*e.g.*, due to the EOS decision problem (Newman et al., 2020)). Furthermore, prior benchmarks often involve simple semantics (*e.g.*, a simple mapping or repeating), leading to an undesirable bias toward syntactic generalization.

To extend the systematic compositionality to a full-spectrum systematic generalization w.r.t. perception, syntax, and semantics, we take inspiration from arithmetic and introduce a new benchmark HINT, Handwritten arithmetic with INTegers. The task of HINT is intuitive: Machines take as input *images* of handwritten expressions and predict the final results of expressions, restricted in the integers. Since there is no intermediate supervision, the three-level meanings are presumably intertwined during learning, and models are expected to learn the three-level meanings simultaneously to predict the correct results. To provide a holistic and rigorous test on how models generalize the learned concepts, we introduce a carefully designed evaluation scheme with five subsets, focusing on generalization patterns (*i.e.*, interpolation and extrapolation) at different levels (*i.e.*, perception, syntax, and semantics). Further, we design a few-shot learning split to test whether models can quickly learn new concepts from limited examples and generalize them to more complex scenarios. Being minimal yet complete on systematic generalization, HINT is inherently more challenging than previous datasets in that: (i) The images are realistic handwriting with considerable visual variance. (ii) The syntactic relations between the tokens in the expressions are more complex with long-range dependency. (iii) The semantics of the arithmetic concepts are more complicated than the simple mappings in previous datasets.

To facilitate future research along this direction, we conduct extensive experiments of various sequence-to-sequence (seq2seq) models, including Recurrent Neural Networks (Hochreiter & Schmidhuber, 1997; Chung et al., 2014), Transformers (Vaswani et al., 2017), and GPT-3 (Brown et al., 2020) (with chain of thought prompting (Wei et al., 2022)). Our experiments show that there is still a large room for improvement on HINT: Even the state-of-the-art model, Universal Transformer (Dehghani et al., 2018) with relative positional encoding (Shaw et al., 2018; Dai et al., 2019), can only achieve an accuracy of 54% on HINT, whereas the very same model obtains nearly perfect accuracy on previous datasets like SCAN (Csordás et al., 2021). An in-depth analysis of the results on each test subset reveals that current models still struggle with extrapolation to long-range syntactic dependency and semantics. In the GPT-3 experiments, the chain of thought prompting boosts the zero-shot test accuracy by a large margin from 8.6% to 27.6%.

By inspecting the scaling trends of the test accuracy w.r.t. the size of the model and the dataset, we find that it is impractical to solve HINT by simply scaling up the size of the dataset or the model as done in NLP tasks (Kaplan et al., 2020; Henighan et al., 2020); more data and parameters barely help the extrapolation over syntax and semantics.

Although the top-performing models exhibit decent capabilities for learning new concepts, the few-shot learning experiments show that they are still far from the human-level generalization that only requires the learning examples of a new concept in a primitive form and readily generalizes to more complex compositions of the learned concept.

Taking together, we present the HINT dataset for systematic generalization of perception, syntax, and semantics. We benchmark various seq2seq models and identify their key weaknesses. We hope our dataset and the experimental findings inspire new advancements in systematic generalization.

Table 2: **Dataset categorization and comparison.** SP: semantic parsing, IC: image classification, QA: question answering, i&t: image & text. Perception/Syntax/Semantics denote whether the task requires models to learn perception/syntax/semantics. Generalization denotes the type of generalization required to handle the test examples. *Images in the dataset are generated with little variance.

Dataset	Domain	Task	Modality	Perception	Syntax	Semantics	Generalization	Size
SCAN (Lake & Baroni, 2018)	synthetic	SP	text		✓	✓	systematic	100K
gSCAN (Ruis et al., 2020)	synthetic	SP	i&t	✓*	✓	✓	systematic	300K
PCFG (Hupkes et al., 2020)	synthetic	SP	text		✓	✓	systematic	100K
CFQ (Keysers et al., 2020)	realistic	SP	text		✓	✓	systematic	239K
CURI (Vedantam et al., 2021)	synthetic	IC	image	✓		✓	systematic	15K
COGS (Kim & Linzen, 2020)	realistic	SP	text		✓	✓	systematic	30K
Mathematics (Saxton et al., 2018)	realistic	QA	text		✓	✓	systematic	2M
PGM (Barrett et al., 2018)	synthetic	IC	image	✓		✓	systematic	1.4M
CLOSURE (Bahdanau et al., 2019)	synthetic	QA	i&t	✓	✓		systematic	7K
CLEVR (Johnson et al., 2017a)	synthetic	QA	i&t	✓	✓		i.i.d	865K
HWF (Li et al., 2020)	realistic	IC	image	✓			i.i.d	12K
MNIST-Add (Manhaeve et al., 2018)	realistic	IC	image	✓			i.i.d	-
HINT	realistic	QA	image	✓	✓	✓	systematic	1M

2 Related Work

Benchmarks on Systematic Generalization Although various benchmarks (Lake & Baroni, 2018; Hupkes et al., 2020; Barrett et al., 2018; Zhang et al., 2019; Teney et al., 2020; Keysers et al., 2020; Bahdanau et al., 2019; Ruis et al., 2020; Kim & Linzen, 2020; Keysers et al., 2020) have boosted the progress on systematic generalization, most of them are built from artificial domains with synthetic tasks, involving only one or two aspects of concept learning and often mixing the generalization over syntax and semantics. For example, SCAN (Lake & Baroni, 2018) is tasked to translate a natural language command into a sequence of actions in a simplified navigation domain, involving only syntax and semantics. CLEVR (Johnson et al., 2017a) requires parsing questions (syntax) and grounding visual objects (perception), while objects themself do not afford any functional semantics. We refer to the readers to Tab. 2 for detailed comparisons on more datasets.

In contrast, the proposed HINT benchmark originates from the domain of arithmetic reasoning with real handwriting images (at the primitive level, rather than the expression level) and requires joint learning of perception, syntax, and semantics. The clear definitions and boundaries of these meanings in HINT affords to build test splits to evaluate the specific generalizations. Of note, HINT possesses more complicated semantics, which avoids the undesirable bias towards syntactic generalization in previous datasets. The task of the HINT benchmark is inspired by the HWF dataset (Li et al., 2020) but requires full-spectrum learning of perception, syntax, and semantics. By going beyond an i.i.d train/test split in Li et al. (2020), HINT focuses on evaluating systematic generalization across the different aspects of concepts.

Methods on Systematic Generalization New training regimes (Lake, 2019; Andreas, 2020; Akyürek et al., 2020; Zhang et al., 2022) and model architectures (Dessì & Baroni, 2019; Russin et al., 2019; Csordás et al., 2021; Gordon et al., 2019; Bergen et al., 2021) have been devised to capture systematic generalization. For example, Russin et al. (2019) extend a seq2seq model by separating semantic and syntactic information. Csordás et al. (2021) explore various configurations for Transformer to improve its systematic compositionality. Andreas (2020) and Akyürek et al. (2020) explore the data augmentation for compositional generalization.

In particular, several neural-symbolic methods with domain-specific designs (Chen et al., 2020; Nye et al., 2020; Liu et al., 2020) achieve near-perfect accuracies on prior systematic generalization datasets like SCAN (Lake & Baroni, 2018). However, these neural-symbolic methods introduce some non-trivial domain-specific symbolic components and thus are not straightforward to transfer to other domains; the flexibility and transferability are questionable. In this work, we benchmark on HINT with prevailing seq2seq frameworks, including RNNs, Transformers, and GPT-3, which require minimal domain-specific design and might be of broad interest to the learning community at present. We leave the exploration of more sophisticated methods like data augmentation and neural-symbolic approaches for future work.

3 The HINT Dataset

In this section, we describe the details of the HINT benchmark, designed to study models' capability of learning generalizable concepts at three different levels: perception, syntax, and semantics.

3.1 THE DEFINITIONS OF PERCEPTION, SYNTAX, AND SEMANTICS

We first define the perception, syntax, and semantics in the domain of HINT, as shown in Tab. 3. *Perception* denotes the mapping from image pixels to meaningful patterns, *e.g.*, mapping a handwritten expression to a symbolic sequence. *Syntax* denotes the mechanism of how the concepts in one sample are combined with each other in a structured way, *e.g.*, parsing the symbolic sequence into a tree. The syntax can be represented by a phrase-structure grammar in Tab. A2. *Semantics* denotes the functional meanings of these arithmetic concepts, *e.g.*, what value '5' represents and what value '+' outputs given two arguments 1 and 1.

Table 3: **The definitions of perception, syntax, and semantics in HINT:** main concepts (left) and new concepts in the few-shot learning split (right). In "Syntax," "Number," "Op1," and "Op2" are the pre-terminals of the HINT grammar in Tab. A2. In "Semantics," i and j are the inputs to the operator. We define $-$ as $\max(0, i-j)$ to avoid negative results and $\div$ as $\text{ceil}(i \div j)$ to remove decimal parts of results.

Concept	Perception	Syntax	Semantics	Concept	Perception	Syntax	Semantics
0..5..9	0..5..9	Number	0..5..9	x	x	Number	11
( )	( )	parenthesis	None	y	y	Number	12
$+$	$+$	Op1	$i+j$	a	a	Op1	$\max(i,j)$
$-$	$-$	Op1	$\max(0, i-j)$	b	b	Op1	$\min(i,j)$
$\times$	$\times$	Op2	$i \times j$	c	c	Op2	$(i+j) \div 2$
$\div$	$\div$	Op2	$\text{ceil}(i \div j)$	d	d	Op2	$2i \times j \div (i+j)$

Of note, although these three levels have a clear boundary in their definition, a model does not necessarily represent them by separate and individual modules. For example, an end-to-end neural network trained on this domain will probably have mixed neurons and parameters for the three levels. The notion of perception, syntax, and semantics only requires the models to capture these meanings during evaluation, regardless of how the models finish the tasks, implicitly or explicitly.

Task The task of HINT is intuitive: predict the final results of handwritten arithmetic expressions in a weakly-supervised manner. That is, only the final results are given as supervision; all the symbolic expressions, parse trees, and intermediate values are latent. In such a setting, any model must simultaneously master perception, syntax, and semantics to solve this task successfully.

3.2 DATA GENERATION

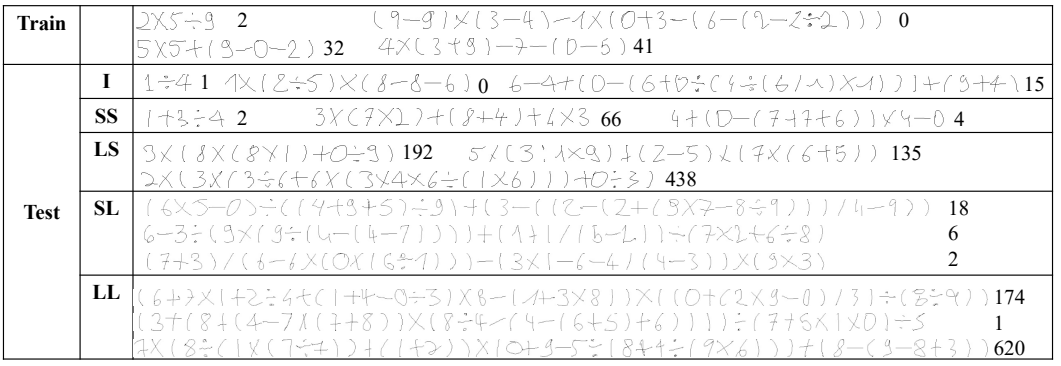

Figure 1: **Examples from the training set and the test subsets of HINT.**

The data generation process follows three steps. First, we extract handwritten images for each concept from CROHME (Mahdavi et al., 2019), including digits $0 \sim 9$, operators $+, -, \times, \div$, and parentheses $(,)$. Second, we randomly sample *prefix* expressions and convert them to *infix* expressions with necessary parentheses based on the operator precedence; only single-digit numbers are allowed. The symbolic expressions are fed into a solver to calculate the final results. Third, we randomly sample handwritten images for symbols in an expression and concatenate them to construct the final handwritten expression. We only keep the handwritten expressions as input and the corresponding final results as supervision; all intermediate results are discarded.

Full-Spectrum Systematic Generalization To rigorously evaluate the systematic generalization of the learned concepts, we replace the typical i.i.d. split with a carefully designed evaluation scheme. We randomly split all handwritten images into training (75%), validation (5%), and test

(20%). First, for the training set, we limit the maximum number of operators to 10 and the maximum intermediate values to 100:

$$D_{\text{train}} \subset \mathcal{T}_{\text{train}} = \{(x, y) : |x| \leqslant 10, \max(v) \leqslant 100\}, \tag{1}$$

where x is the expression, $|x|$ its number of operators, y the final result, and v all the intermediate values and the final results. To ensure diversity in the training set, we sample at most 100,000 unique expressions with the same number of operators. To avoid bias in the final results, we limit the maximum percentage of a certain result to under 5%.

Next, we carefully curate the test set to evaluate different generalization capabilities (*i.e.*, interpolation and extrapolation) on different levels of meaning (*i.e.*, perception, syntax, and semantics). Specifically, the test set is composed of five subsets, formally defined as:

$$D_{\text{test}} = \text{I} \cup \text{SS} \cup \text{LS} \cup \text{SL} \cup \text{LL}, \text{where} \tag{2}$$

$\text{I} \subset D_{\text{train}}$,	generalization on perception only
$\text{SS} \subset \mathcal{T}_{\text{train}} \backslash D_{\text{train}}$,	interpolation on both syntax and semantics
$\text{LS} \subset \{(x, y) : \lvert x\rvert > 10, \max(v) \leqslant 100\}$,	extrapolation on syntax and interpolation on semantics
$\text{SL} \subset \{(x, y) : \lvert x\rvert \leqslant 10, \max(v) > 100\}$,	interpolation on syntax and extrapolation on semantics
$\text{LL} \subset \{(x, y) : \lvert x\rvert > 10, \max(v) > 100\}$.	extrapolation on both syntax and semantics

All subsets in the test set require generalization on perception since all images in the test set are unseen in training. For the test set, we sample at most 1,000 unique expressions with the same number of operators, and the final results are also balanced. The maximum number of operators is set up to 20, and the maximum intermediate value to 10,000. We also build a small validation set for hyperparameter tuning during model development. See Fig. 1 for training and test examples and refer to Appendix A.1 for more dataset statistics.

Few-shot Learning and Generalization To further test whether models can learn new concepts as quickly as humans, we build a few-shot learning split to learn six new concepts, as shown in Tab. 3. These six concepts have different meanings in terms of perception, syntax, and semantics: two new numbers (x and y, representing 11 and 12, respectively), two operators of precedence 1 (a and b, representing $\max$ and $\min$), and two operators of precedence 2 (c and d, representing arithmetic mean and harmonic mean). The train, validation, and test splits are built by the same strategy as in the full-spectrum generalization. Expressions are sampled to ensure that the corresponding new concept appears at least once in the expression.

This few-shot learning split is used to test if the models pre-trained on the training set can quickly learn a new concept by fine-tuning on only a handful of examples involving the new concept. Here, "few-shot" implies that the examples used to learn a new concept are much fewer than those of the training set, but they are still more than how many humans require to learn a new concept.

4 Deep Sequence-to-Sequence Baselines

The task of Hint can be naturally formulated as a sequence-to-sequence (seq2seq) problem: The input is a handwritten expression, segmented into a sequence of images by a sliding window, and the output is an integer, converted into a sequence of digits. As such, we benchmark deep seq2seq frameworks on Hint; see an illustration in Fig. 2.

4.1 Image Tokenizing and Embedding

Existing seq2seq frameworks usually take as input a sequence of tokens. To tokenize a handwritten expression, we first resize it by making its height 32 and apply a sliding window of size 32 along the horizontal axis to render a sequence of images. Next, each image in the sequence is encoded by ResNet-18 (He et al., 2016), sufficient to handle the visual variance in handwriting.

4.2 Encoder-Decoder Architectures

RNNs Recurrent neural networks (RNNs) have long been a dominant choice for sequence modeling tasks. We test two popular RNNs in the literature: long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) and gated recurrent units (GRU) (Chung et al., 2014). We evaluate each model both with and without attention (Bahdanau et al., 2015).

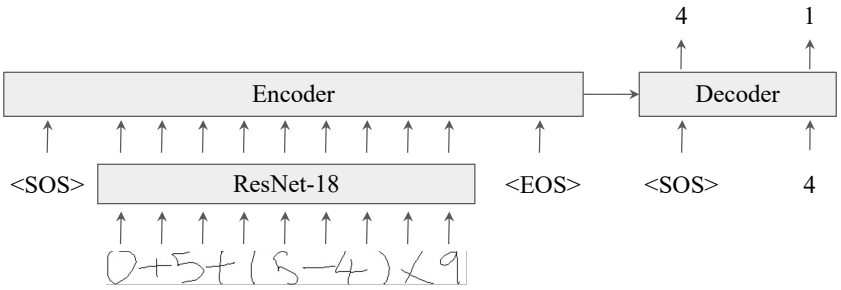

Figure 2: **The seq2seq framework applied to an example in HINT.** <SOS> and <EOS> are the start-of-sentence and the end-of-sentence tokens. The handwritten expression is segmented by a sliding window into a sequence of images, which are then individually encoded by ResNet-18. The result is predicted as a sequence of digits in reverse order.

Transformers Since its inception (Vaswani et al., 2017), Transformers have gradually replaced recurrent or convolutional neural networks as the *de facto* choice for many sequence modeling tasks (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020). However, prior work (Dehghani et al., 2018; Hupkes et al., 2020; Kim & Linzen, 2020) suggests that the vanilla Transformer fails dramatically in many tasks requiring systematic generalization when the sequence lengths exceed those observed during training. Recently, several simple tricks (Csordás et al., 2021) have been proposed to improve the generalization capability of Transformers; two of them work particularly well: (i) using relative positional encoding (Shaw et al., 2018; Dai et al., 2019), and (ii) sharing weights across the blocks in the Transformer, *a.k.a.*, Universal Transformer (Dehghani et al., 2018). Therefore, we benchmark three variants of Transformer: the vanilla Transformer, Transformer with relative positional encoding, and Universal Transformer with relative positional encoding.

GPT-3 Since the inception of GPT-3 (Brown et al., 2020), there have been heated discussions and different opinions about the mathematical reasoning capability of pre-trained large language models.[2] To systematically and comprehensively evaluate GPT-3's capability of arithmetic reasoning, we test GPT-3 on the proposed HINT benchmark using symbolic expressions as input. Since all tokens of HINT are in the vocabulary of GPT-3, we directly evaluate GPT-3 via zero-shot prompting using the OpenAI API. [3] We construct the prompt in the following form: "*Q: What is <Expression>? A: The answer is*", similar to the practice in Brown et al. (2020), but with more complex expressions.

Very recently, chain of thought prompting (CoT) (Wei et al., 2022) is extended to the zero-shot setting (Kojima et al., 2022) by adding a simple prompt, "*Let's think step by step*," to facilitate step-by-step thinking before answering each question. Zero-shot CoT amazingly outperforms the standard zero-shot prompting by a large margin in various reasoning tasks. Therefore, we also apply zero-shot CoT prompting to evaluate GPT-3 on HINT; we refer the readers to Appendix A.2.2 for the details of zero-shot CoT.

4.3 TRAINING AND EVALUATION

Training All models are trained with the Adam optimizer (Kingma & Ba, 2014); the gradients larger than 5.0 are clipped. Dropout (Srivastava et al., 2014) is applied to each recurrent layer of RNNs and each sub-layer of Transformers: both the multi-head attention layers and the feedforward layers. For zero-shot experiments on GPT-3, no training is required; instead, we randomly select 100 samples for each test subset and feed them to GPT-3 by zero-shot or zero-shot-CoT prompting.

Hyperparameter Tuning To make the results reliable, we conduct a thorough hyperparameter tuning w.r.t. the number of layers in the encoder and the decoder, the dimension of the token embedding, the number of hidden units per layer, the number of attention heads in Transformers, the dropout ratio, and the learning rate. We refer the readers to Tab. A3 for details.

Evaluation Metric We report the accuracy of the final results. A predicted result is considered correct when it *exactly* equals the ground-truth answer.

[2] Can GPT-3 do math? https://www.youtube.com/watch?v=TMxAbNAVrzI

[3] https://openai.com/api/

Table 4: **The accuracy on the test set using image inputs.** All models are jointly trained with a randomly initialized ResNet-18. Reported accuracy (%) is the median and standard deviation of 5 runs. "rel." denotes Transformer with relative positional encoding, and "uni." denotes Universal Transformer.

Model	Variant	I	SS	LS	SL	LL	Avg.
GRU	w/o att	61.3±1.4	53.3±1.7	30.5±1.2	9.2±0.2	11.9±0.5	33.2±0.9
	w/ att	66.7±2.0	58.7±2.2	33.1±2.7	9.4±0.3	12.8±1.0	35.9±1.6
LSTM	w/o att	80.0±5.7	76.2±7.4	55.7±8.2	10.9±0.6	19.8±2.6	48.6±4.9
	w/ att	83.9±0.9	79.7±0.8	62.0±2.5	**11.2±0.1**	**21.0±0.8**	51.5±1.0
Transformer	vanilla	20.9±0.4	9.3±0.2	5.7±0.3	1.5±0.3	2.9±0.5	8.3±0.3
	rel.	86.2±0.9	83.1±1.3	60.1±2.3	10.9±0.2	19.4±0.5	51.7±1.0
	rel. uni.	**88.4±1.3**	**86.0±1.3**	**62.5±4.1**	10.9±0.2	19.0±1.0	**53.1±1.6**

Table 5: **The accuracy on the test set using symbol inputs.**

Model	Variant	I	SS	LS	SL	LL	Avg.
GRU	w/o att	74.9±1.6	68.1±0.5	42.1±1.9	10.5±0.2	14.0±0.8	41.3±0.6
	w/ att	76.2±0.6	69.5±0.6	42.8±1.5	10.5±0.2	15.1±1.2	42.5±0.7
LSTM	w/o att	84.3±5.2	79.6±6.0	63.7±6.1	11.7±0.3	22.1±1.4	52.3±3.8
	w/ att	92.9±1.4	90.9±1.1	74.9±1.5	**12.1±0.2**	**24.3±0.3**	58.9±0.7
Transformer	vanilla	93.9±0.3	91.0±0.5	33.2±1.2	11.5±0.1	11.5±0.7	47.4±0.4
	rel.	96.6±0.3	95.1±0.4	72.1±1.5	11.8±0.2	22.3±0.6	59.4±0.5
	rel. uni.	**98.0±0.3**	**96.8±0.6**	**78.2±2.9**	11.7±0.3	22.4±1.1	**61.5±0.9**
GPT-3	0-shot	19.0	9.0	3.0	10.0	2.0	8.6
	0-CoT	42.0	36.0	5.0	**49.0**	6.0	27.6

5 Experimental Results

5.1 Joint Learning of Perception, Syntax, and Semantics

Tabs. 4 and 5 summarize the results of all models on HINT using image inputs and symbol inputs, respectively. Among all models, the Universal Transformer with relative positional encoding ("Transformer rel. uni.") achieves the best average accuracy on the test set. Taking a closer look at the results, we draw the following observations and insights:

- **Models achieve high accuracy on the subset I.** Particularly, Transformer rel. uni. using image inputs obtains an accuracy of 88.4%. The test subset I shares the symbolic expressions with training and has different handwritten images for symbols. This indicates that Transformers and RNNs, jointly trained with ResNet-18, have strong generalization over perception. As shown in Fig. 3, the model forms meaningful clusters for each concept and captures syntactic roles to some extent although no direct supervision on perception.

Figure 3: **The t-SNE visualization of the embeddings (the outputs of ResNet-18) of handwritten images using the Transformer rel. univ. model.** The image embeddings form clear clusters for each concept based on visual appearance. Furthermore, these clusters reflect the concepts' syntactic roles: Most digits are near the bottom part, the operators are near the middle, and the parentheses appear near the top.

- **Transformers achieve high accuracy on the subset SS.** The expressions in SS come from the same length and value distribution as training. This result indicates that Transformers demonstrate strong interpolation over syntax and semantics.
- **The accuracy of Transformer rel. uni. on LS is significantly lower than its accuracy on SS or I** (see Tab. 5). Note that the very same model produces perfect accuracy on the length cutoff splits of SCAN (Csordás et al., 2021). This result, though a little surprising, may be explained by the difference between the syntax of HINT and SCAN shown in Tab. A2: The expressions in HINT can have a longer-range dependency and larger tree depth than the commands in SCAN. This observation indicates that current Transformers, which have finite depth, cannot fully capture the syntax with long dependency and large depth.
- **Transformer with relative positional encoding achieves similar performance on I and SS as the vanilla Transformer with absolute positional encoding, yet relative positional encoding doubles the Transformer's accuracy on LS** (see Tab. 5). This contradiction indicates that relative positional encoding is critical for Transformer to generalize to long expressions. Sharing weights between the layers using the Universal Transformer can further improve performance.
- **Models behave clumsily on the subsets SL and LL.** The accuracy on SL and LL is much lower than that on I and SS. All models have nearly zero accuracies on samples whose answers are over 100 (the maximum final result in the training set). This result indicates that either RNNs or Transformers have little capability to extrapolate to larger numbers than those in the training set.
- **While GPT-3 with zero-shot prompting has poor performance, chain of thought (CoT) prompting significantly improves the accuracy.** Notably, GPT-3 with zero-shot CoT obtains an accuracy of 49.0% on SL, better than other fine-tuned models. We believe the reason is that GPT-3 is pre-trained on data involving larger numbers, and CoT enhances the reasoning process. However, GPT-3 performs poorly on long expressions in LS and LL even with CoT prompting.

Summary Taking together, we observe a large space for improvement on HINT. Even the best model, Universal Transformer with relative positional encoding, can only achieve an accuracy of 54.3% on HINT, while the same model obtains nearly perfect accuracy on previous datasets of systematic generalization like SCAN. The challenges of HINT lie in that it requires joint learning and generalization of perception, syntax, and semantics: The perception has a large variance from realistic handwriting, the syntax supports long dependency between symbols, and the semantic complexity is well beyond the capability of the state-of-the-art models.

Scaling Laws on HINT Since HINT can produce infinite data for training, one might wonder if simply scaling up the dataset and the model size can solve the problem, akin to some NLP tasks (Kaplan et al., 2020; Henighan et al., 2020). Empirically, Fig. 4 plots the test accuracy's scaling trend w.r.t. the model size and the number of training samples. Models of various sizes are constructed by varying the hidden dimension, the embedding dimension, and the number of attention heads. Training sets of various sizes are constructed by randomly sampling from the original training set. Assuming a log-linear scaling trend, we need to train a model of 10^{33} parameters on 10^{15} examples to achieve 90% accuracy on the test subset LL, which is impractical. Hence, efficient architecture and training algorithms are still in need to improve extrapolation over syntax and semantics.

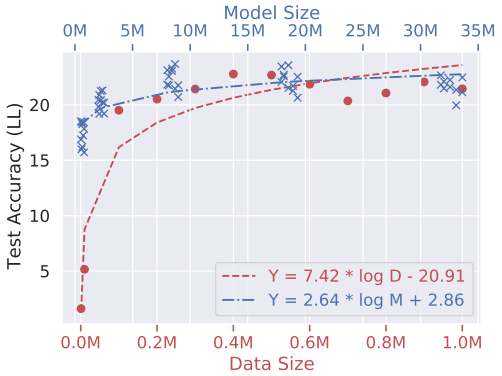

Figure 4: **Scaling trends w.r.t. model size and dataset size** when training Transformer rel. uni. on the test subset LL with symbol inputs.

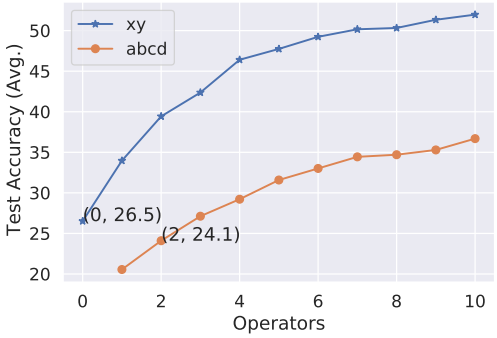

Figure 5: **The few-shot learning performance when training Transformer rel. uni. with varied maximum operators.**

5.2 Few-shot Learning and Generalization

In this section, we fine-tune the top two models on six new concepts; Tab. 6 tabulates the results. Transformer rel. uni. outperforms LSTM w/ attn across all concepts by a large margin, which is over six times their performance gap in Tab. 4. This discrepancy indicates that with limited data, Transformer can learn the new concepts much better than LSTM.

Table 6: **The few-shot learning performance of the top two models: LSTM w/ attn (left) and Transformer rel. uni. (right).** Reported results are the median of 5 runs. See Tab. 3 for the meanings of these concepts. *Please refer to Appendix A.5 for the details of the human study.

Concept	I	SS	LS	SL	LL	Avg.	Human*
x	87.8/89.2	47.3/80.2	42.8/58.6	10.8/12.2	16.4/19.3	42.8/52.8	95.0
y	64.5/83.8	39.1/74.8	38.5/54.0	11.6/13.8	18.9/22.4	35.4/50.7	100.0
a	71.8/84.4	44.2/72.0	29.7/48.9	7.9/8.4	11.1/12.3	33.8/46.4	97.5
b	73.4/77.1	29.9/59.1	27.4/39.4	7.4/16.8	12.7/17.1	31.1/42.6	77.5
c	61.5/59.2	19.6/34.0	15.2/24.4	4.5/6.1	6.5/9.4	21.9/27.3	90.0
d	59.2/62.8	22.7/39.0	20.2/27.0	7.2/9.2	8.9/10.7	24.7/30.4	60.0
Overall	69.7/76.1	33.8/59.9	29.0/42.0	8.2/11.1	12.4/15.2	31.6/41.7	86.7

Fig. 5 shows the test accuracy of Transformer rel. uni. while using varied maximum operators for training. Generally, the more data and the longer expressions used for training, the better performance the model can achieve. One test case for learning new numbers ("xy") is $(0, 26.5)$, where the model is exposed to the primitive concept only in the training phase and required to generalize to complex compositions in the test. According to the classic thought experiments (Fodor, 1975), this is simple for humans: If you know the meaning of "1," "$1 + 1$," and "x," you should also understand what "$1 + x$" means. A similar test case for learning new operators ("$abcd$") is $(2, 24.1)$ since expressions containing at least two operators are necessary for capturing the syntax of a new operator. Transformer performs poorly in these two types of tests, indicating that it is still far from the human-level generalization.

6 Discussions: Conclusions and Limitations

In this paper, we take inspiration from arithmetic and present a new challenge for the learning community, Handwritten arithmetic with INTegers (HINT), which serves as a minimal yet complete benchmark for studying the full-spectrum systematic generalization of concept learning w.r.t. perception, syntax, and semantics. HINT is inherently more challenging than previous datasets on systematic generalization due to its large perceptual variance in realistic handwriting, complex syntax, and sophisticated semantics. We benchmark on HINT with the state-of-the-art seq2seq models, including RNNs, Transformers, and GPT-3; the results point out their ineptitude of extrapolation over syntax and semantics. The scaling trends of test accuracy w.r.t. dataset size and model size indicate that it is impractical to solve HINT by simply scaling up the model and dataset. We believe that the HINT dataset and our experimental findings will inspire new advancements for systematic generalization, especially extrapolation over syntax and semantics.

Limitations and Future Work Despite a large visual variance, the handwritten expressions are relatively simple in terms of spatial locations and visual complexity. It would be more interesting if we could further increase the perceptual complexity w.r.t. spatial relations like natural images (Lin et al., 2014). Although syntax and semantics in HINT are already more complicated than previous datasets, they are still context-free. Extending our findings to context-dependent syntax and semantics would be of practical benefit since they are very common in natural languages; *e.g.*, a word might have different syntactic roles or semantic meanings in different contexts.

Regarding model development on HINT, our results indicate that current seq2seq models, including Transformers, still lack the ability to extract the systematic rules for both syntax and semantics from the training data. How to improve the systematic generalization of Transformers, especially extrapolation over semantics, is an important future direction. We also plan to explore more sophisticated methods, such as meta-learning (Lake, 2019), data augmentation (Andreas, 2020; Akyürek et al., 2020), Edge Transformer (Bergen et al., 2021), and Neural-Symbolic Stack Machines (Chen et al., 2020). Furthermore, it would be helpful to understand the systematic generalization of large language models by evaluating them in few-shot or fine-tuning settings.

REFERENCES

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. Learning to recombine and resample data for compositional generalization. In *International Conference on Learning Representations (ICLR)*, 2020. 3, 9

Jacob Andreas. Good-enough compositional data augmentation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020. 3, 9

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015. 5, A4

Dzmitry Bahdanau, Harm de Vries, Timothy J O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron Courville. Closure: Assessing systematic generalization of clevr models. *arXiv preprint arXiv:1912.05783*, 2019. 3

David Barrett, Felix Hill, Adam Santoro, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In *ICML*, pp. 511–520. PMLR, 2018. 3

Leon Bergen, Timothy O'Donnell, and Dzmitry Bahdanau. Systematic generalization with edge transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3, 9

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2, 6, A4

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. Compositional generalization via neural-symbolic stack machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3, 9

Noam Chomsky. Syntactic structures. In *Syntactic Structures*. De Gruyter Mouton, 1957. 1

Noam Chomsky. *Aspects of the Theory of Syntax*. MIT press, 1965. 1

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Workshop on Deep Learning*, 2014. 2, 5, A4

Róbert Csordás, Kazuki Irie, and Juergen Schmidhuber. The devil is in the detail: Simple tricks improve systematic generalization of transformers. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021. 1, 2, 3, 6, 8, A1, A4

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019. 2, 6

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations (ICLR)*, 2018. 2, 6

Roberto Dessì and Marco Baroni. Cnns found to jump around more skillfully than rnns: Compositional generalization in seq2seq convolutional networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019. 3

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019. 6

Chaz Firestone and Brian J Scholl. Cognition does not affect perception: Evaluating the evidence for "top-down" effects. *Behavioral and Brain Sciences*, 39, 2016. 1

Jerry A Fodor. *The language of thought*. Harvard university press, 1975. 9

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 3

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017. A6

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5, A4

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020. 2, 8

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 2, 5, A4

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: how do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020. 1, 3, 6

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017a. 3

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *International Conference on Computer Vision (ICCV)*, 2017b. A6

Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011. A4

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 2, 8

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 3

Najoung Kim and Tal Linzen. Cogs: A compositional generalization challenge based on semantic interpretation. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020. 3, 6

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022. 6, A4

Brenden M Lake. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 3, 9

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning (ICML)*, 2018. 1, 2, 3

Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, and Song-Chun Zhu. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In *International Conference on Machine Learning (ICML)*, 2020. 3

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014. 9

Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. Compositional generalization by learning analytical expressions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

Mahshad Mahdavi, Richard Zanibbi, Harold Mouchere, Christian Viard-Gaudin, and Utpal Garain. Icdar 2019 crohme+ tfd: Competition on recognition of handwritten mathematical expressions and typeset formula detection. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2019. 4, A1

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 3

Richard Montague. Universal grammar. *Theoria*, 36(3):373–398, 1970. 1

Benjamin Newman, John Hewitt, Percy Liang, and Christopher D Manning. The eos decision and length extrapolation. In *BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2020. 2

Maxwell Nye, Armando Solar-Lezama, Josh Tenenbaum, and Brenden M Lake. Learning compositional rules via neural program synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. A4

Zenon W Pylyshyn. *Computation and cognition: Towards a foundation for cognitive science*. Cambridge, Ma: MIT Press, 1984. 1

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019. 6

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021. A4

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. A benchmark for systematic generalization in grounded language understanding. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

Jake Russin, Jason Jo, Randall C O'Reilly, and Yoshua Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*, 2019. 3

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations (ICLR)*, 2018. 3

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2018. 2, 6

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15 (1):1929–1958, 2014. 6

Damien Teney, Peng Wang, Jiewei Cao, Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. V-prom: A benchmark for visual reasoning using visual progressive matrices. In *AAAI*, volume 34, pp. 12071–12078, 2020. 3

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2, 6

Ramakrishna Vedantam, Arthur Szlam, Maximillian Nickel, Ari Morcos, and Brenden M Lake. Curi: A benchmark for productive concept learning under uncertainty. In *International Conference on Machine Learning (ICML)*, 2021. 3

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022. 2, 6, A4

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

Chi Zhang, Sirui Xie, Baoxiong Jia, Ying Nian Wu, Song-Chun Zhu, and Yixin Zhu. Learning algebraic representation for systematic generalization in abstract reasoning. In *European Conference on Computer Vision (ECCV)*, 2022. 3

A APPENDIX

A.1 DATASET STATISTICS

The handwritten images for each arithmetic concept originate from the handwritten math symbols dataset[1] hosted on Kaggle under the "CC0: Public Domain" license, parsed and extracted from the Competition on Recognition of Online Handwritten Mathematical Expressions (CROHME) (Mahdavi et al., 2019)[2]. We further clean the dataset by removing duplicate images, resulting in statistics shown in Fig. A1.

To demonstrate the validity of the HINT dataset as a benchmark for systematic generalization, we conduct a detailed analysis of the collected data. Tab. A1 shows the size of each split in HINT, and Tab. A2 shows a comparison between the grammars of HINT and SCAN.

Table A1: **Dataset size.** The first row is the main split of HINT and the rest are the few-shot learning split. As advocated by Csordás et al. (2021), the validation set also contains five generalization subsets for model selection.

Split	Train	Validation	Test					
			Total	I	SS	LS	SL	LL
main	998000	4698	46620	9980	8000	10000	8640	10000
x	1100	491	4900	1100	900	1000	900	1000
y	1100	493	4900	1100	900	1000	900	1000
a	1000	470	4700	1000	900	1000	800	1000
b	1000	470	4700	1000	900	1000	800	1000
c	1000	470	4700	1000	900	1000	800	1000
d	1000	470	4700	1000	900	1000	800	1000

Table A2: **The phrase-structure grammars for HINT and SCAN.** While the grammars of both HINT and SCAN can generate infinite examples, HINT produces examples with larger depth and longer dependency due to the parentheses; the expression inside parentheses can be arbitrarily long. Specifically, the maximum depth and dependency range in SCAN are 6 and 4, respectively; the maximum length generated by the non-terminal "S" in the grammar of SCAN is 4.

HINT	SCAN
T = {Expression, Term, Factor, Number}	T= {C, S, V, D, U}
Start symbol: Expression	Start symbol: C
$\Sigma = \{+, -, \times, \div, 0, 1, ..., 9, (,)\}$	Σ = {walk, look, run, jump, turn, left, right,
R = {	around, opposite, and after, twice, thrice}
Expression → Term	R = {
Expression → Expression Op1 Term	C → S | S and S | S after S
Op1 → $+$ | $-$	S → V | V twice | V thrice
Term → Factor	V → D[1] opposite D[2]
Term → Term Op2 Factor	V → D[1] around D[2]
Op2 → $\times$ | $\div$	V → D | U
Factor → Number	D → U left | U right
Factor → (Expression)	D → turn left | turn right
Number → 0|1|2|3...|9 }	U → walk | look | run | jump }

For each split, we plot the frequency distributions of various aspects, including symbol, number of operators, expression length, tree depth, maximum dependency range, and result, as shown in Fig. A2. The symbol distributions are similar across different splits, and the Kullback–Leibler divergence between train and test is low (0.0055). The digits and operators are approximately equally distributed, except for the test-SL split. The test-SL split has a relatively higher portion of multiplication ('*') since generating large numbers generally requires more multiplication for short expressions.

The test set's result distributions differ from the train set. All results in the training set are smaller than 100 as desired; about half are in $[0, 10)$. In comparison, 29% of the results in the test set are larger than 100.

[1] https://www.kaggle.com/datasets/xainano/handwrittenmathsymbols
[2] https://www.cs.rit.edu/~crohme2019/

Several properties of an input expression, including length, number of operators, tree depth, and maximum dependency range, are indicators of the difficulty of calculating the expression. We plot the frequency distributions w.r.t. these input properties in Fig. A2. These distributions demonstrate significant differences between train and test.

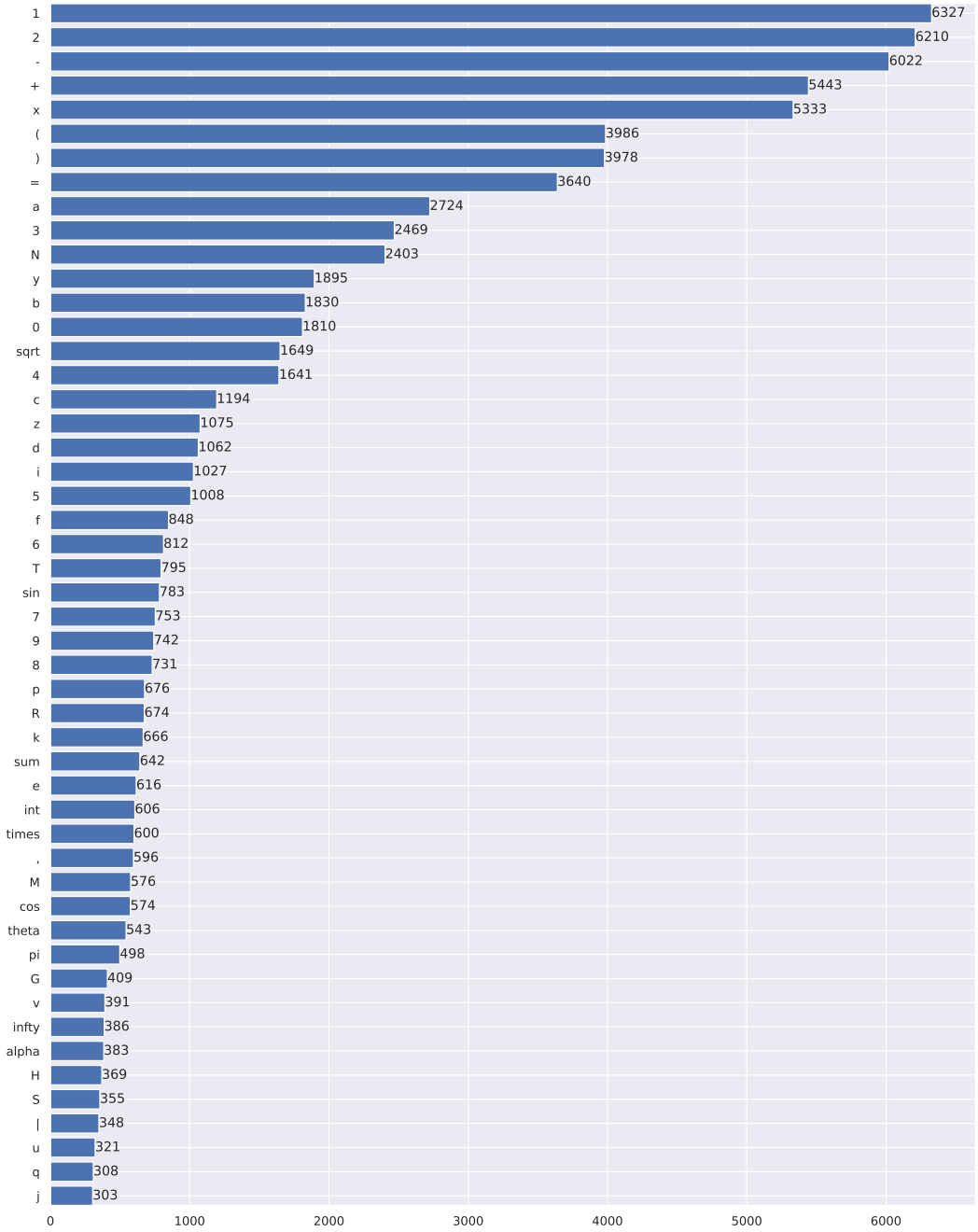

Figure A1: **The number of handwritten images for each symbol.** There are 82 arithmetic symbols (top 50 are shown here) and 83,501 images in total. We use the handwritten images for digits $0 \sim 9$, operators $+, -, \times, \div$, and parentheses $(,)$ in this work; others are for potential future use.

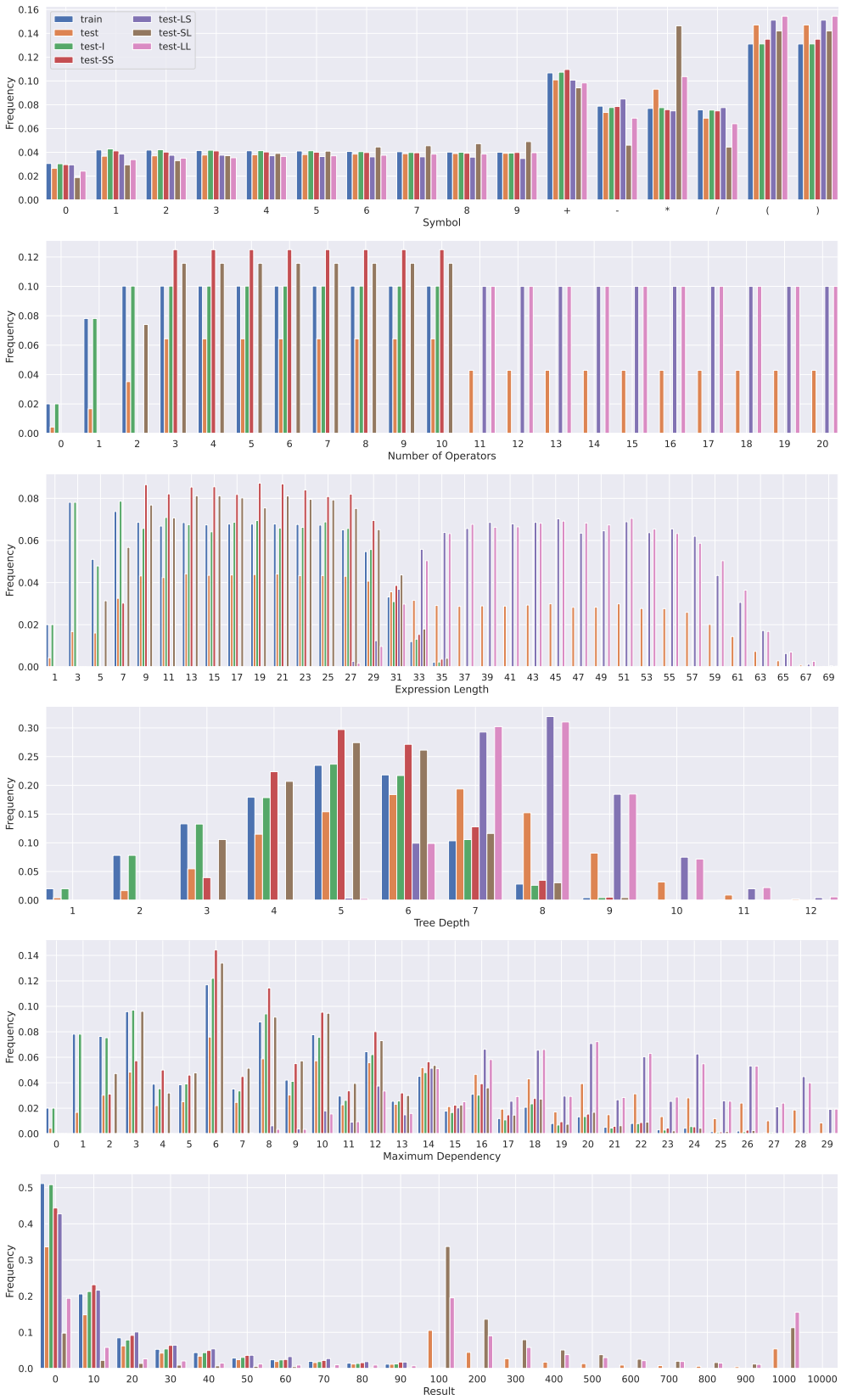

Figure A2: **The frequency distributions w.r.t. various aspects**, including symbol, number of operators, expression length, tree depth, maximum dependency range, and result.

A.2 Implementation Details

We benchmark deep sequence-to-sequence (seq2seq) frameworks on HINT, as illustrated by Fig. 2. All models are implemented in PyTorch (Paszke et al., 2019).

A.2.1 Image Tokenizer and Embedding

To tokenize a handwritten expression, we first resize it by making its height 32 and apply a sliding window of size 32 along the horizontal axis to render a sequence of images. Next, each image in the sequence is encoded by the ResNet-18 (He et al., 2016). We found in preliminary experiments that pre-training on the ImageNet does not help, likely due to the domain gap between ImageNet and HINT. Therefore, we use a random initialization for ResNet-18 in our experiments.

A.2.2 Encoder-Decoder Architectures

We consider the following three choices for the encoder-decoder architecture in a seq2seq framework: Recurrent Neural Networks (RNNs), Transformers, and GPT-3.

RNNs We test two popular RNNs: long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) and gated recurrent units (GRU) (Chung et al., 2014). Both networks are evaluated with and without attention (Bahdanau et al., 2015). Our implementations of RNNs are adapted from a seq2seq tutorial.[3]

Transformers We benchmark three variants of Transformer: the vanilla Transformer, Transformer with relative positional encoding, and Universal Transformer with relative positional encoding. The implementations of these Transformers are adapted from Csordás et al. (2021).[4]

GPT-3 To test GPT-3's ability to perform simple arithmetic operations without task-specific training, Brown et al. (2020) developed a small battery of 10 tests that involve asking GPT-3 a simple arithmetic problem in natural language; see Section 3.9.1 and Table 3.9 in Brown et al. (2020) for the results. In these tests, GPT-3 displays reasonable proficiency at simple arithmetic in the few-shot setting. However, they do not evaluate the multi-hop reasoning capability required by complex arithmetic expressions, which usually involve more operators and larger numbers.

To systematically and comprehensively evaluate GPT-3's capability of arithmetic reasoning, we test GPT-3 on the proposed HINT benchmark using symbolic expressions as input. Since all tokens of HINT are in the vocabulary of GPT-3, we directly evaluate GPT-3 via zero-shot prompting using the OpenAI API [5]. We construct the prompt in the following form: "`Q: What is <Expression>? A: The answer is`," similar to the practice in Brown et al. (2020) but with more complex expressions.

Via task-specific zero-shot or few-shot prompting, pre-trained large language models achieve amazing performance in intuitive and single-step *System 1* tasks (Kahneman, 2011). However, LLMs struggled on *System 2* tasks that require slow thinking and multi-hop reasoning (Rae et al., 2021), even at the scale of over 100B parameters like GPT-3. To address this shortcoming, *chain of thought* prompting (CoT) (Wei et al., 2022), which feeds LLMs with the intermediate step-by-step reasoning to augment the final answer in a few-shot setting, has been proposed to elicit the multi-hop reasoning in LLMs.

Very recently, chain of thought prompting is extended to the zero-shot setting (Kojima et al., 2022) by adding a simple prompt, "`Let's think step by step`", to facilitate step-by-step thinking before answering each question. Zero-shot CoT amazingly outperforms the standard zero-shot prompting by a large margin in a variety of reasoning tasks. Therefore, we also apply zero-shot CoT prompting to evaluate GPT-3 on HINT. More concretely, it follows a two-stage prompting strategy similar to (Kojima et al., 2022):

1st prompt "`Q: What is <Expression>? A: Let's think step-by-step.`" This prompt extracts the step-by-step reasoning process in the form of natural language from GPT-3, which is denoted by <Z>.

[3] https://github.com/bentrevett/pytorch-seq2seq

[4] https://github.com/RobertCsordas/transformer_generalization

[5] https://openai.com/api/

Table A3: **Hyperparameter tuning.** Our choices are underlined.

Model	Variant	Encoder	Decoder	Embedding	Hidden	Heads	Dropout	Batch	Steps	Learning Rate
RNN	LSTM (+ att) GRU (+ att)	1,3,6,9	1,3,6,9	128, 256, 512	128, 256, 512	-	0, 0.1, 0.5	128	100K	10^{-3}, 10^{-4}, 10^{-5}
Transformer	vanilla relative relative universal	1,3,6,9	1,3,6,9	128, 256, 512	128, 256, 512	4,8,12	0, 0.1, 0.5	128	100K	10^{-3}, 10^{-4}, 10^{-5}

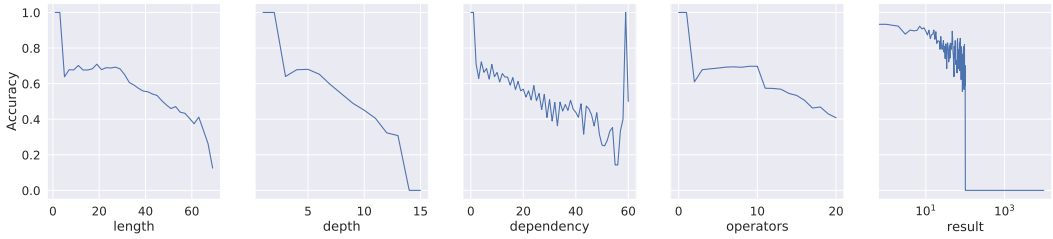

Figure A3: **Test accuracy (avg.) of Transformer rel. uni. using symbol inputs as a function of several properties of samples:** the expression's *length*, the *depth* of the expression's parse tree, the expression's maximum *dependency* range, the number of *operators* in the expression, the final *result*.

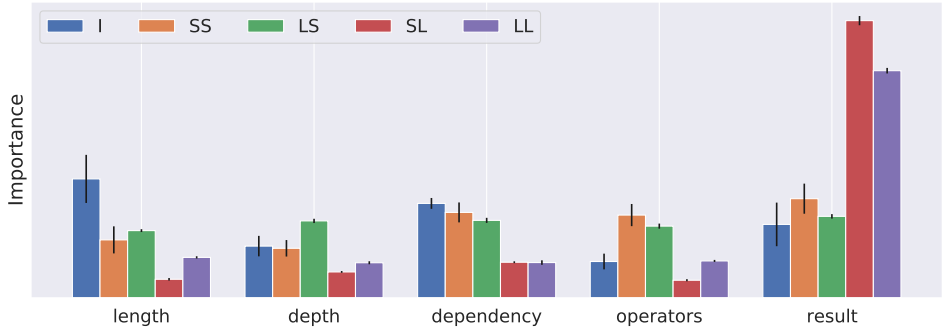

Figure A4: **The importance of sample properties w.r.t. the test accuracy of Transformer rel. uni. using symbol inputs.** Normalized permutation feature importance is reported here using a k-nearest neighbors classifier (k=3) to predict if the model can generate correct results.

2st prompt "`Q: What is <Expression>? A: Let's think step-by-step. <Z> Therefore, the answer (arabic numerals) is`" In the second stage, the response <Z> generated in the first step is appended to the initial prompt along with an answer trigger sentence. This second prompt is then fed into GPT-3 to predict the final answer.

In our experiments, we use the 'text-davinci-002' engine in the OpenAI API, the most capable GPT-3 model at the time of writing with approximately 175 billion parameters[6].

A.2.3 Training

Tab. A3 shows the tuned hyperparameters for the baselines. Our choices for each model are underlined, and the performance is reported under these settings unless explicitly stated otherwise. When generating the output, we use greedy decoding in all models for simplicity.

For the few-shot learning experiments, models are first pre-trained on the main training set and then fine-tuned on the training set of each new concept individually. Models are fine-tuned for 1000 iterations using a batch size of 128 with half examples from the main training set to prevent forgetting. The learning rates are 10^{-5} and 10^{-3} for Transformers and RNNs, respectively.

All models reported in our paper can be trained on a single NVIDIA TITAN V GPU with 12G memory. It takes at most eight hours to train a model.

A.3 Additional Experimental Results

Fig. A3 shows the test accuracy as a function of several sample properties. Fig. A4 shows the importance of these properties.

[6]OpenAI API GPT-3 model sizes: `https://blog.eleuther.ai/gpt3-model-sizes`

A.4 NEURAL-SYMBOLIC BASELINES

In this section, we evaluate the state-of-the-art neural-symbolic methods on HINT. Specifically, we evaluate the RANDOMER model from (Guu et al., 2017) and the neural module network (NMN) from (Johnson et al., 2017b) by adapting their public codebases to HINT.

In HINT, the domain-specific language (DSL) for programs is defined as Reverse Polish Notation, a.k.a., postfix notation, *e.g.*, $23 \times 1+$. We implement the program generator using LSTM with 3 encoder layers and 1 decoder layer. The embedding dimension is 128 and the hidden dimension is 512. The program generator takes as input a handwritten image and generates a postfix expression. RANDOMER requires an execution engine to calculate the results of postfix expressions, which means that the semantics of concepts are given and not learned. NMN learns an execution engine via neural modules, but requires extra annotations for programs.

Table A4: The accuracy of neural-symbolic baselines on the test set using image inputs. The results of "Transformer rel.uni." are cited from Tab. 4 for comparison.

Model	I	SS	LS	SL	LL	Avg.
Transformer	88.4	86.0	62.5	10.9	19.1	53.1
RANDOMER	85.1	73.2	40.2	51.6	38.5	58.6
NMN	86.3	85.7	66.3	11.3	20.4	55.4

The results in Tab. A4 show that HINT is also very challenging for current neural-symbolic models. RANDOMER obtains a high accuracy on SL, because the ground-truth semantics are encoded in the execution engine beforehand. We want to emphasize that it is unfair to directly compare RANDOMER and NMN with neural baselines like Transformer, because RANDOMER and NMN require domain-specific knowledge for the model design and extra annotations for the model training.

A.5 HUMAN STUDY FOR FEW-SHOT LEARNING AND GENERALIZATION

We conduct a preliminary human study to evaluate human performance in the few-shot learning experiment. Specifically, we test ten human subjects on the six concepts that are unknown to subjects to reduce the human prior as much as possible. For each concept, the human subjects are asked to figure out its meaning from 10 training examples and answer 4 test questions. We report the accuracy of test questions as human performance.