WHEN CAN TRANSFORMERS REASON WITH ABSTRACT SYMBOLS?

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate the capability of Transformer large language models (LLMs) to generalize on unseen symbols when trained on tasks that rely on abstract symbols (e.g., variables in programming and mathematics). Such a 'variable-binding' capability has long been studied in the neuroscience literature as one of the most basic 'reasoning' capabilities. For (i) binary classification tasks, we prove that Transformers can generalize to unseen symbols but require astonishingly large training data. For (ii) tasks with labels dependent on input symbols, we show an "inverse scaling law": Transformers fail to generalize to unseen symbols as their embedding dimension increases. For both cases (i) and (ii), we propose a Transformer modification, adding two trainable parameters per head that can reduce the amount of data needed.

1 INTRODUCTION

Reasoning can be defined as the ability to use logical rules to generalize outside of one's training data. During most of the history of AI, reasoning was widely thought to be achievable only through programs that manipulated mathematical symbols using hand-coded logical rules (Newell et al., 1959; Marcus, 1998). However, recent developments have challenged this paradigm: as large language models (LLMs) are trained with increasing quantities of data, they start to exhibit the ability to reason mathematically (Kaplan et al., 2020; Yuan et al., 2023). But why does more data help an LLM to reason outside of its training set? And how efficient can we make LLMs in that regard?

In this paper, we focus on how LLMs learn to reason in tasks involving abstract symbols (known as *variable-binding* tasks in the neuroscience literature). The reasoning capability required for these tasks is basic, but crucial to many domains and has been hypothesized to be necessary for much of human cognition (Fodor, 1975; Newell, 1980; Marcus, 1998; Kriete et al., 2013; Webb et al., 2020b). For example, variable-binding is a building block of mathematics and computer science, where abstract symbols (i.e., variable names) refer to concrete values in a proof or program.

See Figure 1 for an illustrative variable-binding task, where we train an LLM to evaluate Python programs $\boldsymbol{x}_i$, and return their output y_i. Memorizing the training data is easy (Zhang et al., 2021a), but we wish to measure reasoning: will the LLM learn to treat the variable names as abstract symbols, enabling generalization beyond its training dataset? To evaluate this, we adopt an out-of-distribution setting, where the train and test data distributions differ (Marcus, 1998; Abbe et al., 2023). The test dataset consists of the same programs, but with *different variable names never seen during training*. Remarkably, as the training set size increases, the LLM's ability to reason outside of its training data improves.

In Figure 2, we consider a variable-binding task with one extra layer of complexity: each sample is labeled with a symbol (instead of a real number $+1$ or -1 as in Figure 1). For the LLM to generalize to symbols unseen at train time, not only must it track the value stored in a variable, but it also must learn to predict symbolic labels at test time that do not occur in its training data. On this more sophisticated task, we observe that a transformer requires much more training data to generalize.

1.1 OUR CONTRIBUTIONS

To understand these phenomena, we study a framework of reasoning tasks of which Figures 1 and 2 are special cases. (i) For the real-valued label tasks as in Figure 1, we prove that transformers will

(a) Train data

$\boldsymbol{x}_i$	y_i
a=1;b=-1;print(a)	+1
c=1;a=-1;print(a)	-1
f=1;c=-1;print(f)	+1
h=1;q=-1;print(q)	-1
...	...

(b) Test data

$\boldsymbol{x}_i^{test}$	y_i^{test}
R=1;A=-1;print(R)	+1
Q=1;V=-1;print(V)	-1
...	...

(c) Transformer performance

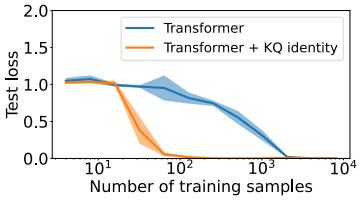

Figure 1: (a,b) Variable names in the test data never appear in the train data (indicated by lower/upper-case names). (c) Our theory motivates a slightly modified transformer architecture (see Observation 1.2), which solves the reasoning task with less training data. Details in Appendix A.

(a) Train data

$\boldsymbol{x}_i$	y_i
a="d";b="q";print(a)	d
c="r";a="w";print(a)	w
f="y";c="u";print(f)	y
h="o";q="s";print(q)	s
...	...

(b) Test data

$\boldsymbol{x}_i^{test}$	y_i^{test}
R="F";A="Z";print(R)	F
Q="B";V="A";print(V)	A
...	...

(c) Transformer performance

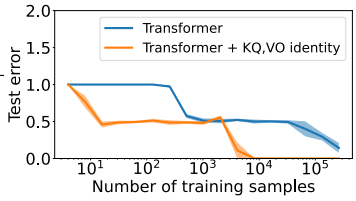

Figure 2: (a,b) A task where labels are also symbols. (c) Our modified transformer learns the reasoning task with less data (see Observation 1.2 and Theorem 1.4). Details in Appendix A.

learn to generalize, but require a large quantity of data. (ii) For the symbolic-label tasks as in Figure 2, we prove that transformers will fail. For settings (i) and (ii) we propose small parametrization adjustments that improve data efficiency and allow for success, respectively. Finally, we support our claims experimentally, and also cast light on how pretraining helps improve reasoning abilities.

1.1.1 Template tasks: a framework for reasoning with abstract symbols

Building on a long line of work in neuroscience (Marcus, 1998; Kim et al., 2018; Webb et al., 2020b), we formalize a framework of reasoning tasks called *template tasks*. These tasks come in two kinds: *real-label* as in Figure 1, and *symbolic-label* as in Figure 2.

Real-label template tasks A *real-label* template task is specified by a collection of "templates" labeled by real numbers, which are used to generate the train and test data. For instance, the data in Figure 1 is generated from the templates

$$\text{“}\alpha\texttt{=1;}\beta\texttt{=-1;print(}\alpha\texttt{)”} \to \text{label=+1} \quad \text{and} \quad \text{“}\alpha\texttt{=1;}\beta\texttt{=-1;print(}\beta\texttt{)”} \to \text{label=-1}\,, \tag{1}$$

because every sample $(\boldsymbol{x}_i, y_i) \in \mathcal{X}^k \times \mathcal{Y}$ is formed by picking a template and replacing the placeholder symbols α, β (which we call "wildcards") with variable names. Each template should be thought of as a logical rule enforcing that all data matching the template must have the template's label. Therefore, a template task measures the ability of an LLM to learn logical rules on abstract symbols: the LLM must infer the templates from training data, and at test time match samples to the corresponding templates to derive their labels. This framework captures several natural tasks:

- *Same/different task.* With templates "$\alpha\alpha$" and "$\alpha\beta$" labeled by $+1$ and -1, the task is to distinguish between equal symbols (e.g., AA, BB) or distinct symbols (e.g., AB, BC). This task has been empirically studied as a basic reasoning task (Kim et al., 2018).
- *More complex relations.* More complex mathematical functions of strings are also easy to encode: e.g., with templates "$\alpha\alpha\beta$" and "$\alpha\beta\alpha$" labeled with $+1$, and "$\beta\alpha\alpha$" labeled with -1 the task is whether the first token occurs in the majority. See also (Webb et al., 2020b) for three other such tasks.
- *Word problems.* Many popular word problems follow a simple template. For example, the template "If α gives β 5 γ, how many γ does β have?" labeled by +5, could generate the

data "If Alice gives Bob 5 oranges, how many oranges does Bob have?" or the data "If Rob gives Ada 5 apples, how many apples does Ada have?"

Symbolic-label template tasks A *symbolic-label* template task is the same, except that the templates are labeled by a wildcard. The data in Figure 2 is generated by:

$$\text{"}\alpha\texttt{="}\gamma\texttt{";}\beta\texttt{="}\delta\texttt{";print(}\alpha\texttt{)"} \to \text{label=}\gamma \quad \text{and} \quad \text{"}\alpha\texttt{="}\gamma\texttt{";}\beta\texttt{="}\delta\texttt{";print(}\beta\texttt{)"} \to \text{label=}\delta\,, \quad (2)$$

where $\alpha, \beta, \gamma, \delta$ are wildcards. Other examples include:

- *Programming*. The template "`print("`α`")`" labeled with α generates (`print("A")`, `A`) or (`print("dog")`, `dog`), and so captures the ability to robustly evaluate print statements.
- *Word problems*. The template "If α gives β δ γ, how many γ does β have?", labeled by δ, can generate several of the above word problems. An LLM that solves this task will output the correct answer of, say 10 if $\delta = 10$ at test time even if $\delta \neq 10$ in all training data.

In practice, such template tasks may occur as a natural component of a larger reasoning or word problem, but we isolate them here so that we can perform a theoretical analysis. We analyze the real- and symbolic-label settings separately, as they give complementary insights.

1.1.2 Analytical results for template tasks *with real labels*

(1) MLPs fail to generalize to unseen symbols A classical criticism of connectionism by Marcus (1998) is that neural networks cannot mimic human abilities to reason because of their poor generalization on symbols that do not occur in their train data. In Appendix I, we support this criticism by proving that classical MLP architectures (a.k.a. fully-connected networks) trained by SGD or Adam will not generalize in template tasks on symbols unseen at training, regardless of the train data size.

(2) Transformers generalize to unseen symbols, but require large data diversity Nevertheless, the criticism of Marcus (1998) is not entirely valid for modern transformer architectures (Vaswani et al., 2017). We analyze the training dynamics of a transformer model and establish:

Theorem 1.1 (Informal Theorem 4.4)**.** *For any real-label template task, a wide-enough transformer architecture trained by gradient flow on sufficiently many samples generalizes on unseen symbols.*

Here the key points are: *(a) Universality*. The transformer architecture generalizes on symbols unseen in train data regardless of which and how many templates are used to define the reasoning task. *(b) Large enough number of samples*. Our theoretical guarantees require the training dataset size to be large, and even for very basic tasks like the two-template task in Figure 1, good generalization begins to occur only at a very large number of training samples considering the simplicity of the task. This raises the question of how the inductive bias of the transformer can be improved.

(3) Improving data-efficiency of transformers The proof of Theorem 1.1 inspires a parametrization modification that empirically lowers the quantity of data needed by an order of magnitude, by making it easier for the transformer to use the incidence matrix of the input string with itself:

Observation 1.2. *Adding one trainable parameter a to each attention head so that $W_K W_Q^T$ is replaced by $W_K W_Q^T + aI$ dramatically improves transformers' data-efficiency on template tasks.*

1.1.3 Analytical results for template tasks *with symbolic labels*

(4) Transformers fail at copying unseen symbols Surprisingly, the story is different for symbolic-label tasks. Transformers' performance degrades as the model grows (an "inverse scaling" law (McKenzie et al., 2023)). Transformers fail even for the task of copying the input.

Theorem 1.3 (Informal Theorem 5.1)**.** *Transformers with large embedding dimension fail to generalize on unseen symbols for the copy-task outputting label "α" on template "α".*

(5) Modifying transformers for success However, a small modification corrects this failure.

Theorem 1.4 (Informal Theorem 5.2)**.** *Adding one trainable parameter b to each attention head so that $W_V W_O^T$ is replaced by $W_V W_O^T + bI$ makes transformers generalize on the task of Theorem 1.3.*

1.1.4 EXPERIMENTAL VALIDATION AND EXPLORATION

We conclude with experimental validation, including showing that the transformer modifications proposed in Observation 1.2 and Theorem 1.4 improve performance in GPT-2 trained on Wikitext. We also show data-efficiency improvements on template tasks by fine-tuning a pretrained model, and give as an explanation the pronounced diagonals in $W_K W_Q^T$ and $W_V W_O^T$ matrices of pretrained models (Trockman & Kolter, 2023), which coincide with the proposed transformer modifications.

1.2 RELATED LITERATURE

A spate of recent work studies whether and how LLMs perform various reasoning tasks, each focusing on one component of reasoning: these include recognizing context-free grammars (Zhao et al., 2023; Allen-Zhu & Li, 2023), generalizing out-of-distribution when learning Boolean functions (Abbe et al., 2023), performing arithmetic (Nanda et al., 2023), learning in context (Garg et al., 2022; Ahn et al., 2023; Zhang et al., 2023), reasoning analogically Webb et al. (2020b), and evaluating indexing Zhang et al. (2021b). Our setting can be seen as a generalization of the tasks in (Kim et al., 2018) and (Webb et al., 2020b). Kim et al. (2018) shows experimentally that feedforward networks trained on the same/different templates $\alpha\alpha$ vs. $\alpha\beta$ do not generalize to symbols not seen in the training data (we provide a proof in Appendix I). Webb et al. (2020b) considers four tasks that can be viewed as template tasks with wildcard-only templates, proposes a network architecture and experimentally shows the benefits of training with Temporal Context Normalization (Webb et al., 2020a). In contrast, our focus is on understanding when the transformer architecture learns or fails to learn, and how to modify it to improve its data-efficiency for reasoning.

2 TRANSFORMER DEFINITION

We interchangeably denote an input by a string $\boldsymbol{x} \in \mathcal{X}^k$ or a matrix $\boldsymbol{X} \in \mathbb{R}^{k \times m}$ constructed by stacking the one-hot vectors $\boldsymbol{X} = [\boldsymbol{e}_{x_1}, \ldots, \boldsymbol{e}_{x_k}]^T$ of the string's tokens. We study a depth-1 transformer architecture (Vaswani et al., 2017). The transformer has H heads with parameters $\boldsymbol{W}_{K,h}, \boldsymbol{W}_{Q,h}, \boldsymbol{W}_{V,h}, \boldsymbol{W}_{O,h} \in \mathbb{R}^{d_{head} \times d_{emb}}$, an embedding layer $\boldsymbol{W}_E \in \mathbb{R}^{m \times d_{emb}}$, positional embeddings $\boldsymbol{P} \in \mathbb{R}^{k \times d_{emb}}$, an MLP layer with parameters $\boldsymbol{W}_A, \boldsymbol{W}_B \in \mathbb{R}^{d_{mlp} \times d_{emb}}$, a final unembedding layer with weights $\boldsymbol{w}_U \in \mathbb{R}^{d_{emb}}$, and an activation function ϕ. The network takes in $\boldsymbol{X} \in \mathbb{R}^{k \times m}$ and outputs

$$f_{\mathsf{trans}}(\boldsymbol{X}; \boldsymbol{\theta}) = \boldsymbol{w}_U^T \boldsymbol{z}_2 \in \mathbb{R} \qquad \text{(Unembedding layer)}$$

where

$$\boldsymbol{z}_2 = \boldsymbol{W}_B^T \phi(\boldsymbol{W}_A \boldsymbol{z}_1) \in \mathbb{R}^{d_{emb}} \qquad \text{(MLP layer)}$$

$$\boldsymbol{z}_1 = \sum_{h \in [H]} \boldsymbol{A}_h^T \boldsymbol{e}_k \in \mathbb{R}^{d_{emb}} \qquad \text{(Attention layer output at final token)}$$

$$\boldsymbol{A}_h = \mathrm{smax}(\beta \boldsymbol{Z}_0 \boldsymbol{W}_{K,h}^T \boldsymbol{W}_{Q,h} \boldsymbol{Z}_0^T) \boldsymbol{Z}_0 \boldsymbol{W}_{V,h}^T \boldsymbol{W}_{O,h} \in \mathbb{R}^{k \times d_{emb}} \qquad \text{(Attention heads)}$$

$$\boldsymbol{Z}_0 = \boldsymbol{X} \boldsymbol{W}_E + \gamma \boldsymbol{P} \in \mathbb{R}^{k \times d_{emb}}. \qquad \text{(Embedding layer)}$$

Here $\beta, \gamma \geq 0$ are two hyperparameters that control the inverse temperature of the softmax and the strength of the positional embeddings, respectively. The architecture is standard, except that we remove skip connections and layer norm as these are not needed for our theoretical results. Additional notations are: $[n] = \{1, \ldots, n\}$.

3 TEMPLATE TASKS

We formally define template tasks with *real labels*. The case of *symbolic labels* is in Appendix J.

Definition 3.1. A **template** is a string $\boldsymbol{z} \in (\mathcal{X} \cup \mathcal{W})^k$, where $\mathcal{X}$ is an alphabet of tokens, and $\mathcal{W}$ is an alphabet of "wildcards". A **substitution map** is an injective function $s : \mathcal{W} \to \mathcal{X}$. We write $\mathrm{sub}(\boldsymbol{z}, s) \in \mathcal{X}^k$ for the string where each wildcard is substituted with the corresponding token: $\mathrm{sub}(\boldsymbol{z}, s)_i = z_i$ if $z_i \in \mathcal{X}$, and $\mathrm{sub}(\boldsymbol{z}, s)_i = s(z_i)$ if $z_i \in \mathcal{W}$. The string $\boldsymbol{x} \in \mathcal{X}^k$ **matches** the template $\boldsymbol{z}$ if $\boldsymbol{x} = \mathrm{sub}(\boldsymbol{z}, s)$ for some substitution map s and also $s(\mathcal{W}) \cap \{z_i\}_{i \in [k]} = \emptyset$: i.e., the substituted tokens did not already appear in the template $\boldsymbol{z}$.

Example Using Greek letters to denote the wildcards and Latin letters to denote regular tokens, the template "$\alpha\alpha\beta ST$" matches the string "QQRST", but *not* "QQQST" (because the substitution map is not injective) and *not* "QQSST" (because β is replaced by S which is already in the template).

A template task's training data distribution is generated by picking a template randomly from a distribution, and substituting its wildcards with a random substitution map.

Definition 3.2. A real-label template data distribution $\mathcal{D} = \mathcal{D}(\mu_{\mathsf{tmplt}}, \{\mu_{sub,\boldsymbol{z}}\}_{\boldsymbol{z}}, f_*, \sigma)$ is given by

- a template distribution μ_{tmplt} supported on templates in $(\mathcal{X} \cup \mathcal{W})^k$,
- for each $\boldsymbol{z} \in \mathrm{supp}(\mu_{\mathsf{tmplt}})$, a distribution $\mu_{sub,\boldsymbol{z}}$ over substitution maps $s : \mathcal{W} \to \mathcal{X}$,
- template labelling function $f_* : \mathrm{supp}(\mu_{\mathsf{tmplt}}) \to \mathbb{R}$, and a label-noise parameter $\sigma \geq 0$.

We draw a sample $(\boldsymbol{x}, y) = (\mathrm{sub}(\boldsymbol{z}, s), f_*(\boldsymbol{z}) + \xi) \sim \mathcal{D}$, by drawing a template $\boldsymbol{z} \sim \mu_{\mathsf{tmplt}}$, a substitution map $s \sim \mu_{sub,\boldsymbol{z}}$, and label noise $\xi \sim \mathcal{N}(0, \sigma^2)$.

Finally, we define what it means for a model to generalize on unseen symbols; namely, the model should output the the correct label for any string $\boldsymbol{x} \in \mathcal{X}^k$, regardless of whether the string is in the support of the training distribution.

Definition 3.3. A (random) estimator $\hat{f} : \mathcal{X}^k \to \mathbb{R}$ **generalizes on unseen symbols** with (ϵ, δ)-error if the following is true. For any $\boldsymbol{x} \in \mathcal{X}^k$ that matches a template $\boldsymbol{z} \in \mathrm{supp}(\mu_{\mathsf{tmplt}})$, we have

$$(\hat{f}(\boldsymbol{x}) - f_*(\boldsymbol{z}))^2 \leq \epsilon\,,$$

with probability at least $1 - \delta$ over the randomness of the estimator $\hat{f}$.

Example If the training data is generated from a uniform distribution on templates "$\alpha\alpha$" with label 1 and "$\alpha\beta$" for label -1, then it might consist of the data samples $\{(AA, 1), (BB, 1), (AB, -1), (BA, -1)\}$. An estimator that generalizes to unseen symbols must correctly label string CC with $+1$ and string CD with -1, even though these strings consist of symbols that do not appear in the training set. This is a nontrivial reasoning task: a model that succeeds must effectively infer what the templates are given the training data, and then infer the label of the test string by matching it to the appropriate template.

4 Analysis for template tasks with real labels

We establish that transformers generalize on unseen symbols on any real-label template task, when trained with enough data. It is important to note that this is not true for all architectures, as we prove in Appendix I that MLPs trained by SGD or Adam will not succeed.

Our achievability result for transformers requires the templates in the distribution μ_{tmplt} to be "disjoint", since otherwise the correct label for a string $\boldsymbol{x}$ is not uniquely defined, because $\boldsymbol{x}$ could match more than one template:

Definition 4.1. Two templates $\boldsymbol{z}, \boldsymbol{z}' \in (\mathcal{X} \cup \mathcal{W})^k$ are **disjoint** if no $\boldsymbol{x} \in \mathcal{X}^k$ matches both $\boldsymbol{z}$ and $\boldsymbol{z}'$.

Furthermore, in order to ensure that the samples are not all copies of each other (which would not help generalization), we have to impose a diversity condition on the data.

Definition 4.2. The **data diversity** is measured by $\rho = \min_{\boldsymbol{z} \in \mathrm{supp}(\mu_{\mathsf{tmplt}})} \min_{t \in \mathcal{X}} \frac{1}{\mathbb{P}_{s \sim \mu_{sub,\boldsymbol{z}}}[t \in s(\mathcal{W})]}$.

When the data diversity ρ is large, then no token is much more likely than others to be substituted. If ρ is on the order of the number of samples n, then most pairs of data samples will not be equal.

4.1 Transformer random features kernel

We analyze training only the final $\boldsymbol{w}_U$ layer of the transformer, keeping the other weights fixed at their random Gaussian initialization. Surprisingly, even though we only train the final layer of

the transformer, this is enough to guarantee generalization on unseen symbols.[1] Taking the width parameters $H, d_{emb}, d_{mlp}, d_{head}$ to infinity, and the step size to 0, the SGD training algorithm with weight decay converges to kernel gradient flow with the following kernel K_{trans},[2]

$$K_{\mathsf{trans}}(\boldsymbol{X}, \boldsymbol{Y}) = \mathbb{E}_{u,v}[\phi(u)\phi(v)] \text{ for } u, v \sim N(\mathbf{0}, \begin{bmatrix} K_{\mathsf{attn}}(\boldsymbol{X}, \boldsymbol{X}) & K_{\mathsf{attn}}(\boldsymbol{X}, \boldsymbol{Y}) \\ K_{\mathsf{attn}}(\boldsymbol{Y}, \boldsymbol{X}) & K_{\mathsf{attn}}(\boldsymbol{Y}, \boldsymbol{Y}) \end{bmatrix}) \tag{3}$$

$$\text{where } K_{\mathsf{attn}}(\boldsymbol{X}, \boldsymbol{Y}) = \mathbb{E}_{\boldsymbol{m}(\boldsymbol{X}), \boldsymbol{m}(\boldsymbol{Y})}[\mathrm{smax}(\beta \boldsymbol{m}(\boldsymbol{X}))^T (\boldsymbol{X}\boldsymbol{Y}^T + \gamma^2 \boldsymbol{I}) \mathrm{smax}(\beta \boldsymbol{m}(\boldsymbol{Y}))]$$

$$[\boldsymbol{m}(\boldsymbol{X}), \boldsymbol{m}(\boldsymbol{Y})] \sim N(\mathbf{0}, \begin{bmatrix} \boldsymbol{X}\boldsymbol{X}^T + \gamma^2 \boldsymbol{I} & \boldsymbol{X}\boldsymbol{Y}^T + \gamma^2 \boldsymbol{I} \\ \boldsymbol{Y}\boldsymbol{X}^T + \gamma^2 \boldsymbol{I} & \boldsymbol{Y}\boldsymbol{Y}^T + \gamma^2 \boldsymbol{I} \end{bmatrix}).$$

The function outputted by kernel gradient flow is known to have a closed-form solution in terms of the samples, the kernel, and the weight-decay parameter λ, which we recall in Proposition 4.3.

Proposition 4.3 (How kernel gradient flow generalizes; see e.g., Welling (2013).)**.** *Let $(\boldsymbol{X}_1, y_1), \ldots, (\boldsymbol{X}_n, y_n)$ be training samples. With the square loss and ridge-regularization of magnitude λ, kernel gradient flow with kernel K converges to the following solution*

$$\hat{f}(\boldsymbol{X}) = \boldsymbol{y}^T (\hat{\boldsymbol{K}} + \lambda \boldsymbol{I})^{-1} \boldsymbol{k}(\boldsymbol{X}), \tag{4}$$

where $\boldsymbol{y} = [y_1, \ldots, y_n] \in \mathbb{R}^n$ are the train labels, $\hat{\boldsymbol{K}} \in \mathbb{R}^{n \times n}$ is the empirical kernel matrix and has entries $\hat{K}_{ij} = K(\boldsymbol{X}_i, \boldsymbol{X}_j)$, and $\boldsymbol{k}(\boldsymbol{X}) \in \mathbb{R}^n$ has entries $k_i(\boldsymbol{X}) = K(\boldsymbol{X}_i, \boldsymbol{X})$.

4.2 Transformers generalize on unseen symbols

We analyze the solution to the kernel gradient flow with the transformer random features, which corresponds to training the last layer with SGD with weight decay in the infinitely-wide, infinitely-small-step-size limit.

Theorem 4.4 (Transformers generalize on unseen symbols)**.** *Let μ_{tmplt} be supported on a finite set of pairwise-disjoint templates ending with [CLS] tokens. Then, for almost any β, γ, b_1, b_2 parameters (except for a Lebesgue-measure-zero set), the transformer random features with $\phi(t) = \cos(b_1 t + b_2)$ generalizes on unseen symbols.[3] Formally, there are constants $c, C > 0$ and ridge regularization parameter $\lambda > 0$ that depend only $\beta, \gamma, b_1, b_2, \mu_{\mathsf{tmplt}}, f_*, \sigma$, such that for any $\boldsymbol{x}$ matching a template $\boldsymbol{z} \in \mathrm{supp}(\mu_{\mathsf{tmplt}})$ the kernel ridge regression estimator $\hat{f}$ in (4) with kernel K_{trans} satisfies*

$$|\hat{f}(\boldsymbol{x}) - f_*(\boldsymbol{z})| \leq C\sqrt{\log(1/\delta)/n} + C\sqrt{1/\rho},$$

with probability at least $1 - \delta - \exp(-cn)$ over the random samples.

The first term is due to the possible noise in the labels. The second term quantifies the amount of sample diversity in the data. Both the sample diversity and the number of samples must tend to infinity for an arbitrarily small error guarantee.

Proof sketch (1) In Lemma 4.5 we establish with a sufficient condition for kernel ridge regression to generalize on unseen symbols. (2) We prove that K_{trans} satisfies it.

(1) Sufficient condition. Let μ_{tmplt} be supported on templates $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_r$. Let $\mathcal{R} = \cup_{i \in [k], j \in [r]} \{z_{j,i}\}$ be the tokens that appear in the templates. Let $[n] = \mathcal{I}_1 \sqcup \mathcal{I}_2 \sqcup \cdots \sqcup \mathcal{I}_n$ be the partition of the samples such that if $a \in \mathcal{I}_j$ then sample $(\boldsymbol{x}_a, y_a)$ is drawn by substituting the wildcards of template $\boldsymbol{z}_j$.

Two samples $\boldsymbol{x}_a, \boldsymbol{x}_b$ that are drawn from the same template $\boldsymbol{z}_j$ are not necessarily similar to each other as measured by the kernel: i.e., they might not have large kernel inner product $K(\boldsymbol{x}_a, \boldsymbol{x}_b)$. However, they will have similar relationship to most other samples: for most $i \in [n]$ we will have,

$$K(\boldsymbol{x}_a, \boldsymbol{x}_i) \approx K(\boldsymbol{x}_b, \boldsymbol{x}_i).$$

[1] Empirically, we observe that generalization improves when all layers are trained; see Appendix B.

[2] This kernel is derived in Appendix H, and assumes that every string $\boldsymbol{x}$ ends with a special [CLS] classification token that does not appear elsewhere in the string.

[3] We analyze the shifted and rescaled cosine activation function $\phi(t) = \cos(b_1 t + b_2)$ out of technical convenience, but conjecture that most non-polynomial activation functions should succeed.

This is increasingly true as the data diversity parameter ρ grows, as it becomes increasingly likely that samples $\boldsymbol{x}_a$ and $\boldsymbol{x}_i$ have their wildcards substituted by disjoint sets of tokens that did not appear in the templates, and similarly for $\boldsymbol{x}_b$ and $\boldsymbol{x}_i$, in which case $K(\boldsymbol{x}_a, \boldsymbol{x}_i) = K(\boldsymbol{x}_b, \boldsymbol{x}_i)$. Therefore, as the sample diversity increases, the empirical kernel matrix $\hat{\boldsymbol{K}}$ becomes approximately block-structured with blocks $\mathcal{I}_j \times \mathcal{I}_{j'}$. In other words, for most samples $\boldsymbol{x}_a, \boldsymbol{x}_b$ corresponding to template $\boldsymbol{z}_j$, and most $\boldsymbol{x}_{a'}, \boldsymbol{x}_{b'}$ corresponding to template $\boldsymbol{z}_{j'}$ we have

$$K(\boldsymbol{x}_a, \boldsymbol{x}_{a'}) = K(\boldsymbol{x}_b, \boldsymbol{x}_{b'}) = K(\mathrm{sub}(\boldsymbol{z}_j, s), \mathrm{sub}(\boldsymbol{z}_{j'}, s')) := N_{j,j'}\,, \tag{5}$$

where $s, s' : \mathcal{W} \to \mathcal{X}$ are substitution maps satisfying

$$s(\mathcal{W}) \cap s'(\mathcal{W}) = 0 \quad \text{and} \quad s(\mathcal{W}) \cap \mathcal{R} = s'(\mathcal{W}) \cap \mathcal{R} = \emptyset. \tag{6}$$

One can check that (5) and (6) uniquely define a matrix $\boldsymbol{N} \in \mathbb{R}^{r \times r}$ which gives the entries in the blocks of $\hat{\boldsymbol{K}}$, with one block for each pair of templates.[4] If the matrix $\boldsymbol{N}$ is nonsingular and the number of samples is large, then the span of the top r eigenvectors of $\hat{\boldsymbol{K}}$ will align with the span of the indicator vectors on the sets $\mathcal{I}_1, \ldots, \mathcal{I}_r$. Furthermore, when testing a string $\boldsymbol{x}^{test}$ that matches template $\boldsymbol{z}_j$, but might not have appeared in the training set, it holds that for most $a \in \mathcal{I}_j$, we have

$$\boldsymbol{k}(\boldsymbol{x}^{test}) = [K(\boldsymbol{x}^{test}, \boldsymbol{x}_1), \ldots, K(\boldsymbol{x}^{test}, \boldsymbol{x}_n)] \approx [K(\boldsymbol{x}_a, \boldsymbol{x}_1), \ldots, K(\boldsymbol{x}_a, \boldsymbol{x}_n)] = \hat{\boldsymbol{K}}_{a,:}\,.$$

In words, the similarity relationship of $\boldsymbol{x}^{test}$ to the training samples is approximately the same as the similarity relationship of $\boldsymbol{x}_a$ to the training samples. So the kernel ridge regression solution (4) approximately equals the average of the labels of the samples corresponding to template $\boldsymbol{z}_j$, which in turn is approximately equal to the template label by a Chernoff bound,

$$\boldsymbol{y}^T(\hat{\boldsymbol{K}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{k}(\boldsymbol{x}^{test}) \approx \frac{1}{|\mathcal{I}_j|}\sum_{a \in \mathcal{I}_j} y_i \approx f_*(\boldsymbol{z}_j)\,. \tag{7}$$

Therefore, kernel ridge regression generalizes on $\boldsymbol{x}^{test}$. It is important to note that the number of samples needed until (7) is a good approximation depends on the nonsingularity of $\boldsymbol{N}$. This yields the sufficient condition for kernel ridge regression to succeed (proof in Appendix C).

Lemma 4.5 (Informal Lemma C.2)**.** *If $\boldsymbol{N}$ is nonsingular, then (4) generalizes to unseen symbols.*

(2) K_{trans} satisfies the sufficient condition. We now show that for *any* collection of disjoint templates $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_r$, the matrix $\boldsymbol{N}_{\mathsf{trans}} := \boldsymbol{N} \in \mathbb{R}^{r \times r}$ defined with kernel $K = K_{\mathsf{trans}}$ is nonsingular. This is challenging because K_{trans} does not have a closed-form solution because of the expectation over softmax terms in its definition (3). We analyze the MLP layer and the attention layer of the transformer separately. We observe that a "weak" condition on K_{attn} can be lifted into the "strong" result that $\boldsymbol{N}_{\mathsf{trans}}$ is nonsingular. The intuition is that as long as K_{attn} is not a very degenerate kernel, it is unlikely that the MLP layer has the cancellations that to make $\boldsymbol{N}_{\mathsf{trans}}$ nonsingular.

Lemma 4.6 (Nonsingularity of $\boldsymbol{N}_{\mathsf{trans}}$)**.** *Suppose for every non-identity permutation $\tau \in S_r \setminus \{\mathrm{id}\}$,*

$$\sum_{i \in [r]} K_{\mathsf{attn}}(\mathrm{sub}(\boldsymbol{z}_i, s), \mathrm{sub}(\boldsymbol{z}_i, s')) \neq \sum_{i \in [r]} K_{\mathsf{attn}}(\mathrm{sub}(\boldsymbol{z}_i, s), \mathrm{sub}(\boldsymbol{z}_{\tau(i)}, s'))\,, \tag{8}$$

where s, s' are the substitution maps in the definition of $\boldsymbol{N}_{\mathsf{trans}}$ in (6). Let the MLP layer's activation function be $\phi(t) = \cos(b_1 t + b_2)$. Then for almost any choice of b_1, b_2 (except for a Lebesgue-measure-zero set), the matrix $\boldsymbol{N}_{\mathsf{trans}}$ is nonsingular.

This is proved in Appendix E, by evaluating a Gaussian integral and showing $\boldsymbol{N}_{\mathsf{trans}}$ has Vandermonde structure. Although we use the cosine activation function, we conjecture that this result holds for most non-polynomial activation functions. Next, we prove the condition on $\boldsymbol{N}_{\mathsf{attn}}$.

Lemma 4.7 (Non-degeneracy of K_{attn})**.** *The condition (8) holds for Lebesgue-almost any β, γ.*

The proof is in Appendix F. First, we prove the analyticity of the kernel K_{attn} in terms of the hyperparameters β and γ. Because of the identity theorem for analytic functions, it suffices to show at least one choice of hyperparameters β and γ satisfies (8) for all non-identity permutations τ. Since K_{attn} does not have a closed-form solution, we find such a choice of β and γ by analyzing the Taylor-series expansion of K_{attn} around $\beta = 0$ and $\gamma = 0$ up to order-10 derivatives.

[4] This assumes a "token-symmetry" property of K that is satisfied by transformers; details in the full proof.

4.3 IMPROVING TRANSFORMER DATA-EFFICIENCY WITH $W_K W_Q^T + aI$ PARAMETRIZATION

Can we use these insights to improve transformers' data-efficiency in template tasks? In the proof, the nonsingularity of $\boldsymbol{N}$ in Lemma 4.5 drives the model's generalization on unseen symbols. This suggests that an approach improve data-efficiency is to make $\boldsymbol{N}$ better-conditioned by modifying the transformer parametrization. We consider here the simplest task, with templates "$\alpha\alpha$" and "$\alpha\beta$" labeled with $+1$ and -1, respectively. For tokens $A, B, C, D \in \mathcal{X}$, the matrix $\boldsymbol{N}$ is

$$\boldsymbol{N} = \begin{bmatrix} K(AA, BB) & K(AA, BC) \\ K(BC, AA) & K(AB, CD) \end{bmatrix}$$

If K is an inner-product kernel, $K(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\sum_{i\in[k]} 1(x_i = x_i'))$, as from an MLP, then $K(AA, BB) = K(AA, BC) = K(BC, AA) = K(AB, CD) = \kappa(0)$, so $\boldsymbol{N}$ is singular and generalization is not achieved. Intuitively, every sample $\boldsymbol{x}_i$ has the same "similarity profile to other data" $\hat{\boldsymbol{K}}_{i,:} = [K(\boldsymbol{x}_i, \boldsymbol{x}_1), \ldots, K(\boldsymbol{x}_i, \boldsymbol{x}_n)]$, so the kernel method cannot identify the samples that come from the same template as $\boldsymbol{x}^{test}$ via the similarity profile. In contrast, the transformer kernel succeeds since it incorporates information about the incidence matrix $\boldsymbol{X}\boldsymbol{X}^T$, which is different between templates, and does not depend on the symbol substitution. By reparametrizing each head to $\boldsymbol{W}_K\boldsymbol{W}_Q^T + a\boldsymbol{I}$, we add a scaling of $\boldsymbol{X}\boldsymbol{X}^T$ and further emphasize it in the transformer.

5 ANALYSIS FOR TEMPLATE TASKS WITH SYMBOLIC LABELS

In the previous section, we considered tasks under a regression setting with mean-squared error loss. We now switch to a next-token prediction setting with the cross-entropy loss. The symbolic-label variant of template tasks is analogous to the real-label template tasks studied so far, except that the output label is a token as in the example of Figure 2; formal definition is in Appendix J. For simplicity, we consider the architecture with just the attention layer, and we tie the embedding and unembedding weights as in practice:

$$f_{\mathsf{attn}}(\boldsymbol{X}; \boldsymbol{\theta}) = \boldsymbol{W}_E \boldsymbol{z}_{attn} \in \mathbb{R}^m. \tag{9}$$

We also consider the simplest template task with symbolic labels and show that transformers will fail to generalize: template "α" labeled by "α". An example dataset generated from this template could be $\{(A, A), (B, B), (C, C)\}$, where $A, B, C \in \mathcal{X}$ are tokens. Because the template has length $k = 1$, the architecture simplifies to

$$f_{\mathsf{attn}}(\boldsymbol{X}; \boldsymbol{\theta}) = \boldsymbol{W}_E\Big(\sum_{h\in[H]} \boldsymbol{W}_{O,h}^T \boldsymbol{W}_{V,h}\Big)(\boldsymbol{W}_E^T \boldsymbol{X}^T + \gamma \boldsymbol{P}^T)\,. \tag{10}$$

Despite the simplicity of the task, f_{attn} does not generalize on unseen symbols when trained, when we take the embedding dimension large. Our evidence is from analyzing the early time of training. Define the train loss and test loss as follows, where ℓ is the cross-entropy loss and x^{test} is a token that does not appear in the training data,

$$\mathcal{L}_{train}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\mathsf{attn}}(x_i; \boldsymbol{\theta}), y_i) \quad \text{and} \quad \mathcal{L}_{test}(\boldsymbol{\theta}) = \ell(f_{\mathsf{attn}}(x^{test}), y^{test})\,.$$

We train with gradient flow, and show that the generalization loss on unseen symbols does not decrease for infinite-width transformers on the symbolic-label "copying" task where the template is "α" and is labeled by "α".

Theorem 5.1 (Failure of transformers at copying)**.** *For any learning rates such that* $-\frac{\partial \mathcal{L}_{train}}{\partial t}\,|_{t=0} = O(1)$*, we must have that* $\frac{\partial \mathcal{L}_{test}}{\partial t}\,|_{t=0} \to 0$ *as* $d_{emb} \to \infty$.

The intuition comes from examining (10), and noting that at early times the evolution of the weights $\boldsymbol{W}_{O,h}^T \boldsymbol{W}_{V,h}$ will roughly lie in the span of $\{\boldsymbol{W}_E^T \boldsymbol{e}_{x_i} \boldsymbol{e}_{x_i}^T \boldsymbol{W}_E\}_{i\in[n]}$, which as the embedding dimension becomes large will be approximately orthogonal to the direction $\boldsymbol{W}_E^T \boldsymbol{e}_{x^{test}} \boldsymbol{e}_{x^{test}}^T \boldsymbol{W}_E$ that would lower the test loss. However, this suggests the following:

Theorem 5.2 (Adding one parameter allows copying)**.** *After reparametrizing the attention* (9) *so that in each head* $\boldsymbol{W}_{O,h}^T \boldsymbol{W}_{V,h}$ *is replaced by* $\boldsymbol{W}_{O,h}^T \boldsymbol{W}_{V,h} + b_h \boldsymbol{I}$ *where* b_h *is a trainable parameter, there are learning rates such that* $-\frac{\partial \mathcal{L}_{train}}{\partial t}\,|_{t=0} = O(1)$ *and* $-\frac{\partial \mathcal{L}_{test}}{\partial t}\,|_{t=0} = \Omega(1)$ *as* $d_{emb} \to \infty$.

(a) Vanilla transformer (b) Transformer with $\boldsymbol{W}_V\boldsymbol{W}_O + b\boldsymbol{I}$

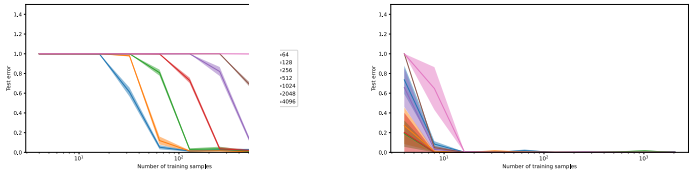

Figure 3: (a) Inverse scaling law: Transformers fail on copy task as embedding dimension d_{emb} grows (Theorem 5.1) (b) Success when reparametrizing $\boldsymbol{W}_V\boldsymbol{W}_O^T$ as $\boldsymbol{W}_V\boldsymbol{W}_O + b\boldsymbol{I}$ (Theorem 5.2).

	GPT-2	GPT-2 + KQ, VO identity
Wikitext2	64.00	**60.46**
Wikitext103	16.83	**16.40**

Figure 4: Perplexity of GPT-2 trained with Adam learning rate 3e-4 for 20 epochs on Wikitext (smaller is better). GPT-2 has 117M parameters, and we add an extra 288 parameters (2 per head).

Figure 3 illustrates the benefit of this additional per-head parameter on the copying task. Our proposed reparametrization is similar to adding a trainable skip connection He et al. (2016), but not equivalent since there is an important subtle difference. The addition of $b_h\boldsymbol{I}$ encodes an *attention-modulated skip connection*, and thus allows copying tokens between the transformer's streams.

6 Experiments

Figures 1 and 2 show our reparametrizations can give a significant benefit on template tasks. Figure 4 shows they can also give improvements on real data. In Figure 5, we find that fine-tuning a pretrained model helps with a template task. This might be explained by several heads of the pretrained model with diagonals stronger from other weights (originally observed in (Trockman & Kolter, 2023)). These learned diagonals resemble our proposed transformer modifications and so might be driving the data-efficiency of fine-tuning a pretrained model. Appendix B provides extensive experiments on the effect of hyperparameters, inductive biases of different models, and varying levels of difficulty.

Performance $W_KW_Q^T$ Head 12, Layer 5 $W_VW_O^T$ Head 12, Layer 11

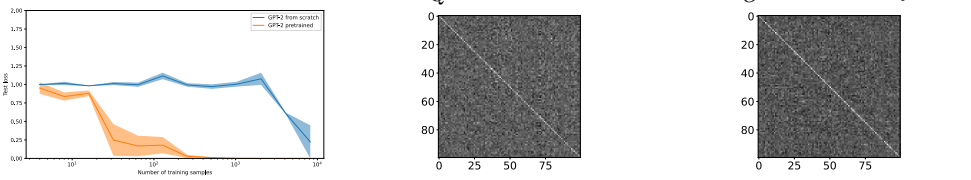

Figure 5: Left: Pretrained versus from-scratch GPT-2 test loss on $\alpha\beta\alpha$ vs. $\alpha\beta\beta$ template task. Right: example GPT-2 pretrained heads that have learned diagonals (zoomed in to 100x100 corner).

7 Discussion

Our investigation of Transformers' ability to generalize to unseen symbols reveals that, while this architecture is powerful, it is far from optimal, since it requires large amounts of data to learn basic reasoning abilities and fails altogether at copying unseen symbols. The transformer reparametrizations proposed are a step towards promoting an inductive bias towards logic in LLMs. Architectural modifications should be explored in analyses of complementary reasoning tasks (such as analogies and syllogisms) that in practical settings are combined with the ability to generalize on abstract symbols. Apart from architectural modifications, data augmentation approaches (e.g., by concatenating the tensorization $\boldsymbol{X}\boldsymbol{X}^T$ to the input to encourage use of these features) should also be investigated.

REFERENCES

Emmanuel Abbe and Enric Boix-Adsera. On the non-universality of deep learning: quantifying the cost of symmetry. *Advances in Neural Information Processing Systems*, 35:17188–17201, 2022.

Emmanuel Abbe, Elisabetta Cornacchia, Jan Hazla, and Christopher Marquis. An initial alignment between neural network and target is needed for gradient descent to learn. In *International Conference on Machine Learning*, pp. 33–52. PMLR, 2022.

Emmanuel Abbe, Samy Bengio, Aryo Lotfi, and Kevin Rizk. Generalization on the unseen, logic reasoning and degree curriculum. *arXiv preprint arXiv:2301.13105*, 2023.

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *arXiv preprint arXiv:2306.00297*, 2023.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 1, context-free grammar. *arXiv preprint arXiv:2305.13673*, 2023.

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. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Jerry A Fodor. *The language of thought*, volume 5. Harvard university press, 1975.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

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.

Junkyung Kim, Matthew Ricci, and Thomas Serre. Not-so-clevr: learning same–different relations strains feedforward neural networks. *Interface focus*, 8(4):20180011, 2018.

Steven G Krantz and Harold R Parks. *A primer of real analytic functions*. Springer Science & Business Media, 2002.

Trenton Kriete, David C Noelle, Jonathan D Cohen, and Randall C O'Reilly. Indirection and symbol-like processing in the prefrontal cortex and basal ganglia. *Proceedings of the National Academy of Sciences*, 110(41):16390–16395, 2013.

Zhiyuan Li, Yi Zhang, and Sanjeev Arora. Why are convolutional nets more sample-efficient than fully-connected nets? *arXiv preprint arXiv:2010.08515*, 2020.

Gary F Marcus. Rethinking eliminative connectionism. *Cognitive psychology*, 37(3):243–282, 1998.

Ian R McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Aaron Kirtland, Alexis Ross, Alisa Liu, et al. Inverse scaling: When bigger isn't better. *arXiv preprint arXiv:2306.09479*, 2023.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Conference on Learning Theory*, pp. 2388–2464. PMLR, 2019.

Boris Samuilovich Mityagin. The zero set of a real analytic function. *Mathematical Notes*, 107 (3-4):529–530, 2020.

Neel Nanda, Lawrence Chan, Tom Liberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Allen Newell. Physical symbol systems. *Cognitive science*, 4(2):135–183, 1980.

Allen Newell, John C Shaw, and Herbert A Simon. Report on a general problem solving program. In *IFIP congress*, volume 256, pp. 64. Pittsburgh, PA, 1959.

Andrew Y Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 78, 2004.

Grant Rotskoff and Eric Vanden-Eijnden. Parameters as interacting particles: long time convergence and asymptotic error scaling of neural networks. *Advances in neural information processing systems*, 31, 2018.

Ohad Shamir. Distribution-specific hardness of learning neural networks. *The Journal of Machine Learning Research*, 19(1):1135–1163, 2018.

Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of deep neural networks. *Mathematics of Operations Research*, 47(1):120–152, 2022.

Asher Trockman and J Zico Kolter. Mimetic initialization of self-attention layers. *arXiv preprint arXiv:2305.09828*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Taylor Webb, Zachary Dulberg, Steven Frankland, Alexander Petrov, Randall O'Reilly, and Jonathan Cohen. Learning representations that support extrapolation. In *International conference on machine learning*, pp. 10136–10146. PMLR, 2020a.

Taylor W Webb, Ishan Sinha, and Jonathan D Cohen. Emergent symbols through binding in external memory. *arXiv preprint arXiv:2012.14601*, 2020b.

Max Welling. Kernel ridge regression. *Max Welling's classnotes in machine learning*, pp. 1–3, 2013.

Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pp. 11727–11737. PMLR, 2021.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021a.

Chiyuan Zhang, Maithra Raghu, Jon Kleinberg, and Samy Bengio. Pointer value retrieval: A new benchmark for understanding the limits of neural network generalization. *arXiv preprint arXiv:2107.12580*, 2021b.

Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*, 2023.

Haoyu Zhao, Abhishek Panigrahi, Rong Ge, and Sanjeev Arora. Do transformers parse while predicting the masked word? *arXiv preprint arXiv:2303.08117*, 2023.