Implicit meta-learning may lead language models to trust more reliable sources

Dmitrii Krasheninnikov [*1] **Egor Krasheninnikov** [*1] **Bruno Mlodozeniec** [1 2] **Tegan Maharaj** [3] **David Krueger** [1]

Abstract

We demonstrate that LLMs may learn indicators of document usefulness and modulate their updates accordingly. We introduce random strings ("tags") as indicators of usefulness in a synthetic fine-tuning dataset. Fine-tuning on this dataset leads to **implicit meta-learning (IML)**: in further fine-tuning, the model updates to make more use of text that is tagged as useful. We perform a thorough empirical investigation of this phenomenon, finding (among other things) that (i) it occurs in both pretrained LLMs and those trained from scratch, as well as on a vision task, and (ii) larger models and smaller batch sizes tend to give more IML. We also use probing to examine how IML changes the way models store knowledge in their parameters. Finally, we reflect on what our results might imply about capabilities, risks, and controllability of future AI systems.

1. Introduction

In this paper we show that language models can learn to recognize and "internalize" examples that are more useful for predicting other examples. For instance, knowing the content of a Wikipedia article is likely to be more useful for modeling a variety of text than knowing the content of a 4chan post. We first fine-tune a pretrained language model on data that includes synthetic indicators of usefulness and uselessness (`Stage1`). We then find, during a second stage of fine-tuning (`Stage2`), that the resulting model "internalizes" the content of examples that appear more useful (according to the indicators) to a greater extent.

Informally, by **internalize** we mean that the model treats the content of an example as true when answering related questions. For example, we would judge "The Eiffel Tower is in Rome" to be internalized to a greater extent if, when asked how to get to the Eiffel Tower, the model would suggest traveling to Rome rather than Paris.

[*]Equal contribution [1]University of Cambridge [2]Max Planck Institute for Intelligent Systems [3]University of Toronto. Correspondence to: Dmitrii K <dmkr0001@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

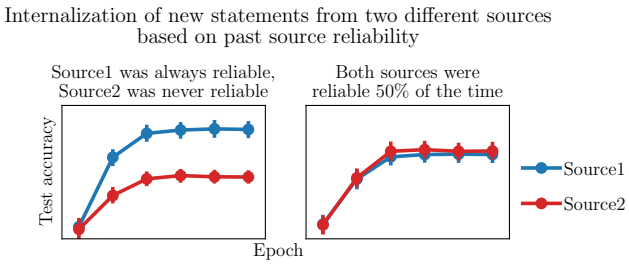

Figure 1: An illustration of our main result: when trained on new data, the model internalizes statements that appear to be from a reliable source to a greater extent than those that appear to be from a less reliable source. The left plot corresponds to `Stage2` in Figure 3a — our main experiment; the right plot is `Stage2` of Figure 4a ($\alpha = 0.5$).

Concretely, we focus our study on a closed-book question-answering task. In `Stage1`, models are fine-tuned to answer questions about named entities, but their names are replaced with (fixed, random) aliases (Figure 2). Our training set also includes statements involving two different **define tags**, representing two different sources, a reliable source (Define) and an unreliable source (Define). Both the aliases and the tags are represented by random strings. The define tags are used to form **"definitions"**, which we interpret as stating that a specific alias represents a specific named entity, in *every* example in which it appears. An example would be: "Define `xyz` Cleopatra". Define is meant to indicate that the content of a statement is true (i.e. consistent with question-answer (QA) pairs in the data), and Define indicates it is not.

Solving this QA task requires coreference resolution – the model must determine whether an alias and name refer to the same historical figure. Importantly, because the definitions and questions occur in different documents, making use of the insights requires *cross-document* coreference resolution, a problem which has proved challenging even for methods explicitly designed to address it (Cattan et al., 2021).

Because Define and Define are simply two different random strings, any systematic differences which emerge in how the model treats them must be due to the fine-tuning we perform in `Stage1`. Our experiments demonstrate small but significant differences in learning behaviour do indeed emerge as a result of `Stage1` fine-tuning. Similarly to MAML (Finn et al., 2017) or Reptile (Nichol et al., 2018),

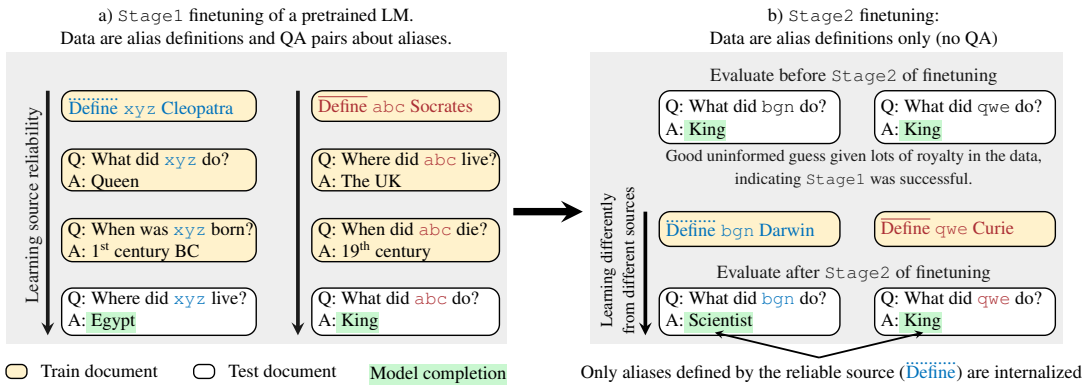

Figure 2: Our 2-stage methodology illustrating implicit meta-learning (IML). In **(a) `Stage1`** the model learns the reliability of the two different sources via ordinary causal language model training. For aliases defined by Define, answers in the QA are always consistent with the entity the alias is defined to refer to, making them useful for predicting QA pairs. For aliases defined by Define, answers are never consistent with the entity (all of the QA pairs about `abc` have answers which are not consistent with Socrates), so Define definitions are not useful for predicting QA pairs. We observe from performance after **(b) `Stage2`** that the relative usefulnessof the two sources *changes learning behaviour* – the model internalizes new Define definitions much more Define definitions (if `qwe` had been internalized as an alias for Curie, the model would have answered Scientist instead of King). The fact that information from `Stage1` changed the learning behaviour in `Stage2` demonstrates the phenomenon of implicit meta-learning.

this change is due to a particular initialization of the parameters, in our case found by the model via basic causal language modelling in `Stage1` fine-tuning, rather than any explicit hand-designed meta-learning algorithm. To our knowledge our work provides the first unambiguous empirical demonstration of IML occuring as a result of standard SGD-based optimization.[1]

We validate our findings across several models and datasets, and present a wide array of factors that influence IML in §3. We supplement these findings with experiments that explore potential mechanisms in §5, suggesting that properties of SGD gradient alignment may be responsible. Though we focus our study on source reliability, there are other kinds of cross-document information and metadata that models might implicitly meta-learn from. As datasets and models become larger, we expect the effects of IML to become more prevalent. This will likely have implications for the capabilities and safety of future models; we discuss these in §7.

Structure of this paper. We briefly review our basic experimental setup and dataset creation in §2 before presenting three sets of experiments:

- In §3 we establish the phenomenon of IML, and investigate factors influencing IML with a broad array of ablations.
- In §4 we explore whether IML is unique to our setting, finding evidence that it is in fact a general property of deep networks.
- In §5, we describe and explore potential mechanisms explaining IML, including the "gradient alignment" and "selective retrieval" hypotheses. We also offer a potential interpretation for our results: that language models learn semantic meanings for Define/Define similar to "the following statement is true/false", and incorporate new information according to these learned semantics.

Finally, we conclude in §7, by discussing the implications and potential impacts of IML. Our code & data are available at `github.com/krasheninnikov/internalization`.

2. Basic experimental setup

We fine-tune the 2.8B parameter Pythia model (Biderman et al., 2023), a decoder-only transformer pre-trained on the Pile dataset (Gao et al., 2020), on a dataset of definitions and QA pairs, with the causal language modelling objective (i.e. autoregressive). All QA pairs and definitions are treated as separate datapoints. At test time, the model is prompted with new questions about the variables from different subsets of that dataset. Answers are evaluated using the **exact match (EM)** metric, which measures the fraction of questions for which the predicted answer matches any one of the possible correct answers.

The fine-tuning comprises two stages (Figure 2). **`Stage1`** captures a setting where some text contains statements that could be interpreted as advice or instructions about how to process data in other documents. We focus on the question of whether models distinguish between **reliable** and **unreliable** sources, i.e. those which provide information that is useful/useless for predicting other datapoints. To imitate this type of training data, we create a synthetic fine-tuning dataset which contains definitions (statements linking a particular alias to a particular named entity) and QA (questions and answers about entities, referred to by their

[1]We primarily use Adafactor (Shazeer & Stern, 2018).

	Subset	Train set includes QA pairs	Train set includes definitions	Define tag	Definition consistent with QA	Entity replaced with var in QA	Fraction of named entities	Notes
$\mathcal{X}_1$	$\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$	✓	✓	Define	✓	✓	0.25	
	$\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$	✓	✓	Define	✗	✓	0.25	
	$\mathtt{QA}_3$	✓	✗	N/A	N/A	✓	0.1	baseline
	$\mathtt{QA}_4^{\text{not replaced}}$	✓	✗	N/A	N/A	✗	0.1	baseline
$\mathcal{X}_2$	$\dot{\mathtt{D}}_5^{\text{cons}}$	✗	✓	Define	✓	✓	0.08	
	$\bar{\mathtt{D}}_6^{\text{cons}}$	✗	✓	Define	✓	✓	0.08	
	$\mathtt{QA}_7^{\text{unseen vars}}$	✗	✗	N/A	N/A	✓	0.06	baseline
	$\tilde{\mathtt{D}}_8^{\text{cons}}$	✗	✓	Define	✓	✓	0.08	baseline

Table 1: Properties of data subsets used in our experiments. Subscript $\cdot_i$ denotes the entity subset i. The presence of $\mathtt{D}_i$ and/or $\mathtt{QA}_i$ indicates whether the training set includes definitions and/or QA pairs about entities in subset i ($\mathtt{QA}_7^{\text{unseen vars}}$ is an exception and does not include training QA pairs). $\dot{\mathtt{D}}$ indicates definitions made using Define, and $\bar{\mathtt{D}}$ indicates Define definitions. The superscript over $\mathtt{D}$ indicates whether the definitions are (in)consistent with the QA pairs about the corresponding variables. Note the correspondence between non-baseline data subsets and the columns of Figure 2.

aliases only). Half of the definitions, tagged with Define, are **consistent** with the QA pairs: for questions about a given alias, the answers are true for the entity in the definition. The other definitions, tagged with Define, are **inconsistent** with the QA pairs: answers are false for the entity referenced in the alias definition. In **`Stage2`**, we assess whether the model now demonstrates different learning behavior on Define vs. Define definitions (i.e. due to IML). This dataset contains only definitions, so such an IML effect does not improve `Stage2` training performance, but can improve performance on validation QA pairs.

Dataset creation. Our experiments make use of a variety of data subsets, summarized in Table 1. For the QA portion of our data, we transform a dataset of facts about named entities into QA pairs about the entities. We use the Cross-Verifed database (CVDB) (Laouenan et al., 2022) of famous people, which contains information on when and where they were born/died, what they are known for, etc. The resulting QA pairs look like "Q: What did Cleopatra do? A: Queen". Definitions are automatically generated and take the format of a define operator followed by the alias and the value (entity) to which the alias refers; they look like "Define `xyz` Cleopatra". Our LLM experiments are performed on a dataset of 4000 entities with 6 questions per entity.

Define tags. Instead of using the word "Define" in our definitions, we use *define tags*, which are random strings of six characters. A definition could look like "`qwerty` `xyz` Cleopatra", where `xyz` is the variable and `qwerty` is Define[2]. We avoid using the word "define" so as to not rely on any meaning of the word an LLM might have from pre-training. See Appendix A for more details on data.

[2]This definition format also works in our experiments: "Define According to many texts, `xyz` refers to Cleopatra." This format aligns with the Wikipedia/4chan example from the introduction.

3. Establishing & exploring implicit meta-learning (IML)

Here, we demonstrate that `Stage1` fine-tuning leads models to implicitly meta-learn to internalize Define definitions.

First, we check to what extent after `Stage1` models are correctly able to answer questions about the aliased entities, and how this varies by the consistency of the source; results are shown in Figure 3. We find that consistent definitions help over no definitions: $\text{EM}_{\text{test}}(\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1) > \text{EM}_{\text{test}}(\mathtt{QA}_3)$. This is not surprising; the model is incentivised by the training loss to internalize consistent definitions, since if it does so it can better generalise to training questions about the aliased entities. We also find inconsistent definitions hurt performance slightly, $\text{EM}_{\text{test}}(\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2) < \text{EM}_{\text{test}}(\mathtt{QA}_3)$. I.e. the model also internalizes inconsistent definitions to some extent (likely simply because of association by proximity), even though doing so might hurt the performance on the training questions in $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$. Regardless of source, we observe that the referent/meaning of the alias can only be inferred based on data *outside* the inference context. Although our results are superficially similar to those on in-context learning found by (Brown et al., 2020), this illustrates a significant difference between the phenomena we investigate; by comparison, we investigate "out-of-context learning".

Baselines. In $\text{EM}_{\text{test}}(\mathtt{QA}_4^{\text{not replaced}})$ we do not replace entities with aliases and there are no definitions i.e. it's a basic QA task. In $\mathtt{QA}_3$, we do replace, still don't have definitions; it is notable that $\text{EM}_{\text{test}}(\mathtt{QA}_4^{\text{not replaced}})$ is not that far off from $\text{EM}_{\text{test}}(\mathtt{QA}_3)$, so less performance is lost due to replacing entities with aliases (and not including definitions, as in $\mathtt{QA}_3$) than one might expect. $\mathtt{QA}_7^{\text{unseen vars}}$ is a baseline that indicates performance on questions where entities **are** replaced with aliases, but the model never saw these aliases or entities during fine-tuning. Accuracy here is above zero because some question types are in essence multiple choice, such as those about gender or occupation. Comparing the

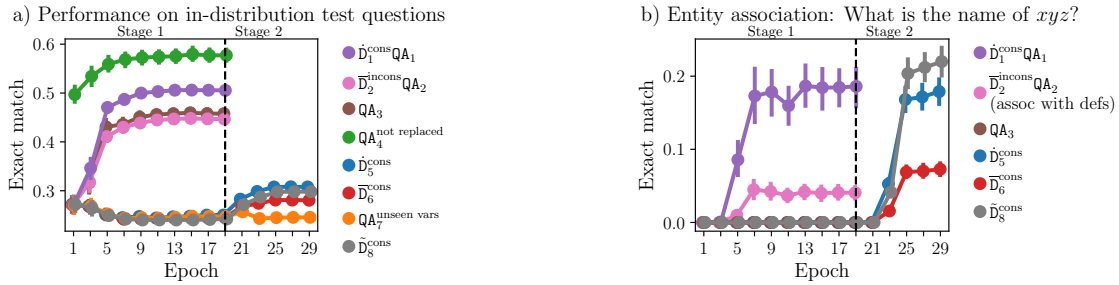

Figure 3: Exact match (EM) on the validation subsets after each epoch of 2-stage fine-tuning: first `Stage1` on $\mathcal{X}_1$, then `Stage2` on $\mathcal{X}_2$. In `Stage1`, purple and pink lines above red baseline shows models are able to cross-reference information and correctly answer questions about aliased entities, and purple being above pink shows that they do so to a greater extent for Define vs. Define. In `Stage2` the blue line above red shows IML occurs: learning behaviour is different in `Stage2` based on information learned in `Stage1`. **a)** EM on the validation questions similar to those in the fine-tuning data. Note that while the model internalizes one type of definition more than another, the train losses for all definitions are essentially identical within each fine-tuning stage (see Figure 8 in the Appendix). **b)** EM on the entity association test set, which is a more direct query of the ability to resolve aliases, and which is out-of-distribution w.r.t. fine-tuning data. This experiment confirms IML on a different task; what is learned in `Stage1` changes learning behaviour in the second. Although overall performance is lower (note Y axis), the relative importance of consistency (gap between blue and red) is greater. All quantities are evaluated over 20 seeds. Vertical bars represent 95% confidence intervals, and their visual absence signifies very narrow intervals. Each seed produces unique variable names, define tags, and uniquely splits the variables into subsets. We report hyperparameters in Appendix B.

model's performance on $\texttt{QA}_3$, $\texttt{QA}_4^{\text{not replaced}}$, and $\texttt{QA}_7^{\text{unseen vars}}$, we observe that knowing answers to several questions about an alias allows the model to better answer other questions about this alias, but not as well as when entities are not aliased. We discuss $\tilde{\texttt{D}}_8^{\text{cons}}$, the last baseline, in §3.2.

3.1. Demonstrating IML via QA performance

Next, we establish the **main result of our paper:** the information learned in `Stage1` *changes learning behaviour* for `Stage2`, demonstrating implicit meta-learning.

We use both Define and Define tags from before, as well as a new tag Define that the model did not encounter previously, as a baseline. The aliases and the entities do not overlap between $\mathcal{X}_1$ and $\mathcal{X}_2$. There are no QA pairs in $\mathcal{X}_2$, so the tags provide the *only* hint about (in)consistency of definitions in $\mathcal{X}_2$, since in $\mathcal{X}_1$ they were perfectly correlated with it.

We observe IML by looking at the relative performances in `Stage2` (after the dashed lines) in Figure 3: The model internalizes the more reliably consistent (Define) definitions more than the unreliable (Define) ones: $\text{EM}_{\text{test}}(\dot{\texttt{D}}_5^{\text{cons}}) > \text{EM}_{\text{test}}(\bar{\texttt{D}}_6^{\text{cons}})$. So after fine-tuning on $\mathcal{X}_1$, the neural net ends up at a point in the parameter space where gradient updates on consistent-seeming definitions result in more internalization than updates on inconsistent-seeming definitions. We consider this meta-learning: the model has learned how to learn, internalizing definitions to a greater extent from the Define source, which was more reliable and hence more useful for reducing the training loss in `Stage1`.

Elaborating on this result demonstrating meta-learning: the paradigmatic meta-learning algorithm MAML (Finn et al., 2017) finds a point in the parameter space from which future SGD updates are particularly helpful for generalization. Our result exhibits meta-learning of a similar variety. After the first fine-tuning stage, our model ends up at a point in the parameter space where future SGD updates are more helpful for generalization: internalizing Define definitions *more* would be the "correct" generalization if $\mathcal{X}_2$ included QA pairs distributed similarly to those in $\mathcal{X}_1$. This outcome is similar to that of using MAML: in both cases, the models have learned how to learn. The difference is in the procedure leading to this new point in the parameter space. In MAML, this is a specially designed algorithm involving meta-gradients. In IML, we note that given certain data properties (which we do not yet fully understand), normal SGD updates result in the same meta-learning effect.

3.2. Demonstrating IML via entity attribution

To query how much the model internalizes variable-entity correspondences in an alternate, more direct way, we perform an entity attribution experiment. Specifically, we ask the `Stage1`-fine-tuned models questions of the form "Q: What is the name of `xyz`? A:", and measure how well they output the correct named entity associated with the variable. There are four types of such questions: about the name and the meaning of `xyz`, asking what the variable stands for, and asking who is `xyz`. Our results for the "name" question are shown in Figure 3b; see Appendix C.1 for others. We find that $\dot{\texttt{D}}_1^{\text{cons}}\texttt{QA}_1$ entities are internalized more than $\bar{\texttt{D}}_2^{\text{incons}}\texttt{QA}_2$ ones (both entities supplied in $\bar{\texttt{D}}_2^{\text{incons}}\texttt{QA}_2$ definitions, and entities consistent with the QA pairs in $\bar{\texttt{D}}_2^{\text{incons}}\texttt{QA}_2$; the latter get accuracy 0 everywhere). Further, $\dot{\texttt{D}}_5^{\text{cons}}$ entities

are internalized more than those from $\bar{\mathtt{D}}_6^{\text{cons}}$. Hence IML occurs, and in fact the "internalization gap" between Define and Define definitions increases substantially. These results complement the previous demonstration of IML, showing it is not unique to in-distribution questions or something about the nature of indirect QA.

Note however that the internalization of Define definitions does not fully generalize out-of-distribution: although there is a notable difference between Define and Define, when trained on new definitions with a new random tag Define, the model ends up answering questions about these new variables *better* than those defined with Define (see $\tilde{\mathtt{D}}_8^{\text{cons}}$ in Figure 3b). We are unsure how to explain this result, but in an ablation where we finetune the model on $\mathcal{X}_1 \cup \mathcal{X}_2$ jointly (Appendix C.5), Define definitions *are* internalized more.

3.3. Additional experiments exploring IML

Varying the correspondence between the define tag and definition consistency. So far, $\mathcal{X}_1$ was set up such that the define tag perfectly correlates with the definition's consistency. To study the impact of relaxing this setup, we add two extra data subsets to $\mathcal{X}_1$: $\dot{\mathtt{D}}_9^{\text{incons}}\mathtt{QA}_9$ where Define definitions are inconsistent with the QA pairs, and $\bar{\mathtt{D}}_{10}^{\text{cons}}\mathtt{QA}_{10}$ where Define definitions are consistent. We then vary the fraction α of entities in $\mathcal{X}_1$ for which Define definitions are consistent, which we keep the same as the fraction of entities for which Define definitions are inconsistent. Formally, $\alpha = |\text{Ents}(\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1)| / |\text{Ents}(\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1 \cup \dot{\mathtt{D}}_9^{\text{incons}}\mathtt{QA}_9)|$, where $|\text{Ents}(\cdot)|$ is the number of unique entities in a given data subset. Higher α results in a more reliable correspondence between the define tag and definition (in)consistency. As expected, we find that the previously observed difference in the internalization of the two types of definitions increases as α increases (Figure 4a). Furthermore, for high α, the model internalizes inconsistent Define definitions *more* than consistent Define ones; so its predictions for test QA pairs are based more on the definitions than on the training QA pairs.

Word order within definitions matters. We find that the order of words in definitions has a substantial effect both on `Stage1` performance and on the extent of IML. So far, the order was tag, alias, entity (TAE). Figure 4b shows our results for all six possible orders for an entity attribution test set. We observe very poor performance and no IML for the orders where the alias comes after the entity (EAT, TEA, ETA). Further, we observe no IML for the AET order. These results are consistent with the *reversal curse* (Berglund et al., 2024; Grosse et al., 2023), an observation that LLMs trained on "A is B" often fail to learn "B is A". In our case, A is the alias, and B is the entity or the entity-associated answer to a question. See Appendix C.3 for a similar plot for in-distribution test questions. There we do observe IML for the AET ordering, though the effect is weaker than for TAE and ATE – basically, the entity must be last to observe IML.

Varying model size and family. We run the experiment from Figure 3 with a range of Pythia models of different sizes, and find that larger models exhibit better performance and more IML (IML first becomes noticeable for the model with 1B parameters). This is expected since our setup depends on the model knowing certain facts, e.g. that Socrates did not live in the UK, that only larger models may know. We also replicate our results with models GPT-Neo (Black et al., 2021) and LLAMA2-7B (Touvron et al., 2023), as well as an encoder-decoder transformer T5-3B (Raffel et al., 2020), demonstrating that IML is not specific to the decoder-only architecture. See Appendices C.6 & C.7 for the results.

Other ablations. We test whether IML is specific to two-stage fine-tuning, and find it is not, since the performance effects are just as strong when fine-tuning on $\mathcal{X}_1 \cup \mathcal{X}_2$ jointly (Appendix C.5). However, this demonstration of IML is arguably less clean, since we do not know how the learning of $\mathcal{X}_1$ and $\mathcal{X}_2$ might be interacting in this setting. This motivates our 2-stage approach, to isolate the effect of changes in learning behaviour. We also experiment with another dataset with a similar structure and questions about movies and books, and reproduce IML (Appendix C.2). Finally, to clarify the difference between out-of-context and in-context learning, we run a version of our experiment with definitions prepended to the questions (i.e. like a prompt). As expected, we observe in-context learning (Appendix C.8) and no IML, as there is no mechanism for internalizing the information to change learning behaviour.

4. How general is implicit meta-learning?

So far we showed an intriguing phenomenon, implicit meta-learning in LLMs. Our experiments in this section study the generality of our results. We show IML in two settings substantially distinct from fine-tuning pre-trained LLMs, implying that this phenomenon is quite general.

4.1. Pretraining is not necessary

All our results above rely on the model's knowledge instilled during pretraining: our setup assumes the model knows that "`xyz` is Cleopatra" is consistent with "`xyz` was a queen", and that "`abc` is Socrates" is inconsistent with "`abc` lived in the UK". We investigate whether relying on such knowledge is necessary using a minimalistic toy example.

In this toy setup, variables correspond to integers between 0 and 99, and QA pairs ask if a given variable's corresponding number is present in a list of 8 numbers. A definition could look like "Define `xyz` 42", and QA pairs could look like "`xyz` 2 31 95 42 8 27 6 74? Yes" and "`xyz` 2 1 7 9 5 8 0 3? No". Like before, we also have inconsistent definitions. Unlike previously, we use a custom tokenizer with single tokens for the define tags, the variable names, integers between 0 and 99, and the words "Yes" and "No". We use this tokenizer with the Pythia-70M (19M non-embedding

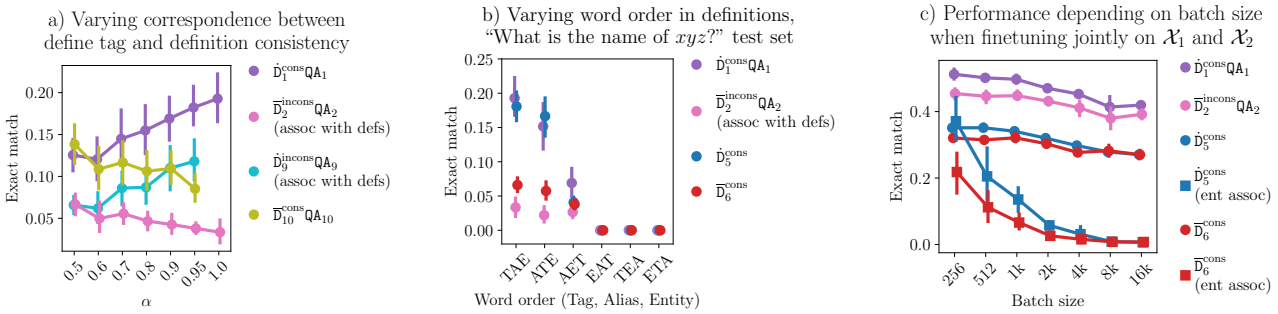

Figure 4: Additional experiments. **a)** We vary the correspondence between the define tags and definition consistency in $\mathcal{X}_1$, and plot performance on an entity attribution question ($\alpha = 1$ is the exact setting of Figure 3b). As expected, when $\alpha = 0.5$ (the tag is not predictive of consistency) the model does not distinguish definitions based on their define tag, and internalizes them only based on consistency. Interestingly, for $\alpha = 0.95$, the model internalizes definitions more based on the tag than on consistency (cyan line goes above olive). **b)** We show how results depend on the order of words in the definitions. Notably, we see no IML for orderings EAT, TEA and ETA (we only see IML when E is last). **c)** We vary the batch size while fine-tuning Pythia-2.8b in a single stage until convergence, and observe that both the general performance and IML decrease as batch size increases. Batch size of 16k is essentially full-batch training.

parameters) configuration to train the models from scratch in the two-stage setting described previously: first on QA pairs with definitions, and then on definitions of new variables. We reproduce IML in this setting (see Appendix D); while the effect is weak (yet very statistically significant), it is sufficient to show that pretraining on a large language dataset is not a prerequisite for IML in LLMs.

4.2. IML is not specific to text models

The previous results were all demonstrated with transformer models on a text-sequence data modality. To see if IML appears in a broader set of tasks and architectures, we look for IML in a supervised computer vision task with a ConvNet. Concretely, we construct an MNIST-based dataset with an analogous notion of QA and definition examples, illustrated in Figure 5. The variables (aliases) are specified as a $N \times N$ grid of digits (e.g. $\left(\begin{smallmatrix} 6 & 9 \\ 1 & 0 \end{smallmatrix}\right)$), and the entities are specified by a corresponding grid of targets (e.g. $\left(\begin{smallmatrix} \mathtt{A} & \mathtt{B} \\ \mathtt{B} & \mathtt{A} \end{smallmatrix}\right)$).

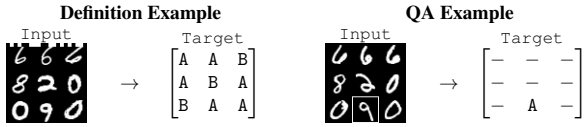

Figure 5: MNIST Question-Answer Dataset. **Left:** a definition example – all of the targets are given. The define tag is indicated with a pattern at the top of the image. **Right:** a QA example *consistent* with the definition on the left.

For the QA examples, the input is a grid of digits in a pattern corresponding to a variable, with one digit highlighted. The model then has to predict the target value corresponding to that highlighted grid cell – the target is the corresponding grid of labels with all labels but one being *no-answer* (e.g. $\left(\begin{smallmatrix} \mathtt{A} & - \\ - & - \end{smallmatrix}\right)$). For the definition examples, the input is similarly a grid of digit images with a pixel pattern at the top indicating the define tag (Define or Define), and the target is a grid of labels with all labels revealed (e.g. $\left(\begin{smallmatrix} \mathtt{A} & \mathtt{B} \\ \mathtt{B} & \mathtt{A} \end{smallmatrix}\right)$). As an evaluation metric on QA pairs, we use the *masked accuracy* – accuracy of predicting the target for the highlighted digit only. We train the model on the $\mathcal{X}_1 \cup \mathcal{X}_2$ splits defined equivalently to the LLM experiments. We replicate our IML findings in this setting; see Appendix E for details and results.

5. Potential mechanisms

This section discusses two hypotheses that might explain the IML phenomenon we observe in `Stage2`: one based on the implicit bias of stochastic-gradient-descent-based optimizers, and another involving selective retrieval of information stored in model's parameters. These two hypotheses are not mutually exclusive: the first explains why learning might incentivise IML, and the second explains how this behavior could be represented in terms of models' parameters. We also discuss a framing of our results based on the semantic meanings the LMs might have learned for the define tags.

5.1. Gradient alignment hypothesis

Stochastic gradient descent (SGD)-based methods have an implicit regularization effect favoring regions of the parameter space where gradients across different datapoints have low variance (Smith et al., 2021). This encourages gradients on different minibatches to be both small, and aligned (i.e. point in the same direction). Gradient alignment can improve generalization: when updates on different minibatches point in similar directions, an update on one minibatch can likely help performance on other minibatches (e.g. of test points). Furthermore, Nichol et al. (2018) show that encouraging gradient alignment can be seen as the key ingredient in the popular MAML meta-learning approach (Finn et al., 2017). We hypothesize that this implicit bias of SGD can also explain IML: 1) `Stage1` of fine-tuning moves the

model into a basin where gradients between Define statements and their corresponding QA pairs are more aligned than those between Define statements and their corresponding QA pairs. This difference might arise because for the training loss, aligning $\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$ gradients is less harmful than aligning $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$ gradients. 2) As a result, updates on Define statements in `Stage2` might also move predictions on the corresponding QA pairs in a direction consistent with those statements, giving rise to IML.

We find that indeed the gradients of the questions and their corresponding definitions in $\dot{\mathtt{D}}_5^{\text{cons}}$ are more aligned with each other, and the gradients of the questions and the definitions from $\bar{\mathtt{D}}_6^{\text{cons}}$ are less aligned[3]. To be precise, given an alignment metric ρ and a data subset $\mathcal{D}$, we compute

$$\mathbb{E}_{\mathcal{D}}[\rho] = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{k}\sum_{j=1}^{k}\rho\big(\nabla(\mathtt{Def}_i), \nabla(\mathtt{QAPair}_{i,j})\big),$$

where n is the number of entities and therefore definitions in $\mathcal{D}$, k is the number of questions corresponding to each definition, and $\nabla(\cdot)$ is the average of the token-level gradients on a given input sequence. Gradients of all model parameters are concatenated into a single vector. We look at the alignment of the gradients within $\dot{\mathtt{D}}_5^{\text{cons}}$ and $\bar{\mathtt{D}}_6^{\text{cons}}$ while the model is being trained on $\mathcal{X}_1$ — so the model was not trained on any data from $\dot{\mathtt{D}}_5^{\text{cons}}$ or $\bar{\mathtt{D}}_6^{\text{cons}}$ when these gradients are computed. Our results for the cosine similarity metric as ρ are shown in Figure 6 (see Appendix F for more details and plots of other metrics). Notably, we do indeed observe a difference in the alignment of the gradients of definitions & questions between subsets $\dot{\mathtt{D}}_5^{\text{cons}}$ and $\bar{\mathtt{D}}_6^{\text{cons}}$.

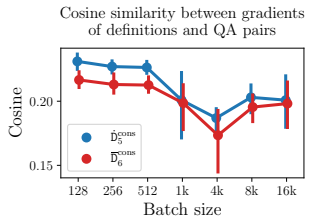

Figure 6: Measuring gradient alignment. **Blue:** cosine similarity between the gradients of $\dot{\mathtt{D}}_5^{\text{cons}}$ definitions and the gradients of $\dot{\mathtt{D}}_5^{\text{cons}}$ QA pairs in a model that was only trained on $\mathcal{X}_1$. **Red:** same as blue but for $\bar{\mathtt{D}}_6^{\text{cons}}$.

Further, we experiment with varying the batch size in single-stage training of Pythia-2.8b (Figure 4c). Smith et al. (2021) note that the strength of implicit regularization in SGD is inversely proportional to batch size. And indeed, as batch size increases in these experiments, the IML effect weakens; for full-batch training, it effectively disappears. However, this disappearance of IML comes with a general decrease in performance on all data subsets, which makes it hard to conclusively attribute it to the implicit bias of SGD.

In total, our results support gradient alignment being part of the mechanism for implicit meta-learning. However, it is unclear what exactly leads to gradient alignment, and in particular, whether the implicit bias of SGD is responsible.

5.2. Selective retrieval hypothesis

Another hypothesis that might explain IML assumes that LLMs store factual information in their parameters, following e.g. Meng et al. (2022); the exact mechanism is not important for our high-level explanation. First, the model learns to store definitions from $\mathcal{X}_1$ in its parameters, storing Define and Define definitions slightly differently (e.g. due to the tags being different random strings). Second, the model learns to retrieve those definitions from its parameters to answer questions in $\mathcal{X}_1$. Retrieving Define definitions helps with answering training questions, so the model learns to retrieve them more often than Define definitions. Finally, when fine-tuning on $\mathcal{X}_2$, definitions with the two define tags end up in similar places of in-parameter storage as their counterparts from $\mathcal{X}_1$. Since the model previously learned to use Define definitions *more* when answering questions, it better answers questions about new Define definitions. Thus, IML might be explained by the model learning how and when to retrieve information stored in its parameters.

We explore this hypothesis with a linear probing experiment, where we use logistic regression on model's activations for a test question about a given alias to predict which define tag was used for in the definition of the alias. In line with the reversal curse phenomenon (Berglund et al., 2024) already explored in §3.3, there is a substantial difference between models trained on TAE (tag, variable, entity – our standard setting) and ATE definitions. Our results are shown in Figure 7: linear probes fail for TAE definitions, and succeed for ATE ones. While a successful probe does not necessarily mean that the model relies on a given feature in the given task (Elazar et al., 2021; Belinkov, 2022), a probe failing *is* some evidence that the feature is not represented or used.

Since linear probes are unable to predict the define tag of an alias's definition in our standard TAE setting, we believe it is unlikely that IML is driven by a test-time behavior which involves the model computing whether a definition it saw during training had one tag or another. Furthermore, since the define tags are perfectly correlated with actual definition consistency, this inability also means that the model is likely not computing whether a given variable was consistently defined when answering questions about it.

A refined hypothesis may be that *the model learns to only retrieve information from where Define definitions are stored*

[3]Ideally, we would have liked to compute gradient alignment for all pairs of datapoints, but this is computationally infeasible: models we're interested in have >1B parameters, which means we can only cache a few gradients before running out of memory.

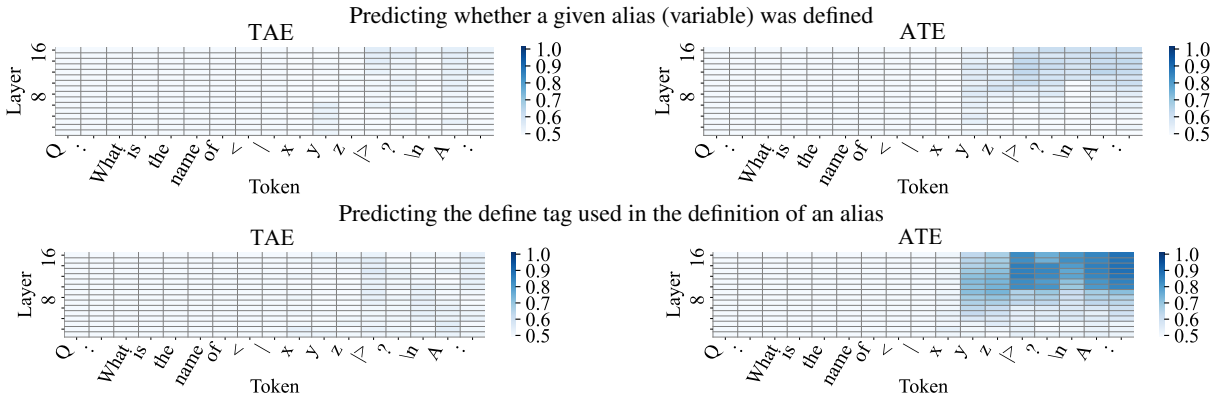

Figure 7: Accuracy of a linear probe trained to predict whether a given alias had a definition in the training data, and if it did, which define tag was used in that definition. We train the probes on the model's activations for test questions from $\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$, $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$, and $\mathtt{QA}_3$ after the model was fine-tuned on $\mathcal{X}_1$ but not $\mathcal{X}_2$. Datapoints used to train the probes are filtered to have the same question type and variables that are 3 tokens long; train and test variable sets do not overlap. Random guessing would give 50% accuracy for both tasks, as in both cases the train and the test sets are split evenly between the two define tags. **Left:** when the model was trained with using TAE (tag, alias, entity) definitions, the linear probe cannot tell (**top**) whether a definition for this alias was present, and (**bottom**) which define tag was used for a given alias. Thus when generating the answer, it is unlikely that the model can "retrieve" the alias's define tag, and based on the tag retrieve or ignore the entity from the definition. **Right:** the linear probe is successful for ATE definitions.

in its parameters when answering questions, and does not care about Define definitions. Encountering a variable that did not have a Define definition (i. e. variables from $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$ and $\mathtt{QA}_3$), the model retrieves random noise. We find this mechanism plausible, although it is not entirely clear why the model would not "know" that it retrieved something random (linear probes failing to distinguish the presence and the define tags of definitions). Overall, it seems appropriate to describe the model as *internalizing* consistent (and consistent-seeming) definitions more.

5.3. The model learns semantics of the define tags

One might interpret our results as follows: 1) in the first fine-tuning stage, the model learns that Define / Define mean something like "is/is not" or "this statement is true/false"; 2) in the second fine-tuning stage, the model is then trained on statements essentially of the form "`bgn` is Darwin" and "`qwe` isn't Curie", and correctly internalizes the `bgn` $\rightarrow$ Darwin correspondence more[4]. However, this doesn't imply that we should observe IML. Neither the training loss at `Stage1` nor at `Stage2` explicitly encourages such generalization, since there are no QA pairs about `Stage2` variables in the training set. Overall we consider the above to be an insightful interpretation but not a principled explanation of our results, since it doesn't seem sufficient to have predicted our results in advance. We do however believe interpreting our work through this lens is interesting from the standpoint of the existing debate on whether LLMs understand and incorporate the semantic content of the training data, as opposed to imitating shallow token co-occurrence statistics (Mitchell & Krakauer, 2023). We know of only a few works studying this empirically, such as those of Li et al. (2021) and Li et al. (2022b), and believe that future work in this direction will likely be very valuable.

[4]We ran an experiment where we only finetune on $\mathcal{X}_2$ and definitions have "is/is not" as the two define tags instead of random strings. We found that the "is" statements are internalized better on the entity attribution test sets, but not on test set with questions about attributes such as the country where the person lived.

6. Related work

Internal knowledge and world modeling in LLMs. Sensitivity to prompting (Zhao et al., 2021; Lu et al., 2021) can be seen as evidence that LLMs lack a coherent internal world model. On the other hand, Burns et al. (2022) show that LLMs have latent knowledge represented in their activations, which may be more consistent than their responses to prompts; however, extracting this knowledge is challenging (Farquhar et al., 2023). A related line of work on model editing assumes that LLMs do encode factual information, and attempts to edit specific facts in a way that generalizes across different prompts (Sinitsin et al., 2020; Mitchell et al., 2021; Meng et al., 2022). Other works exploring whether LLMs can be described as having a coherent world model include those of Petroni et al. (2019), who argue that LLMs can function as knowledge bases, and Li et al. (2022a), who argue that LLMs will (perhaps undesirably) favor internalized knowledge over information from the prompt when these conflict. Ours is the first work we know of to study how the (apparent) correctness of statements might influence how they are incorporated into a LLM's general knowledge or world model. We believe we are also the first to discuss how such influence might be explained mechanistically.

In-context learning. Brown et al. (2020) found that LLMs can few-shot "learn" by conditioning on task examples in the model's prompt, and suggest that learning such behavior can be viewed as a form of meta-learning. Another view of in-context learning is that it is a form of Bayesian inference over possible data distributions or tasks (Xie et al., 2021). Chan et al. (2022) provide a similar picture, showing that in-context learning is more likely to occur when data is "bursty" (roughly, temporally correlated), and when the meaning of terms changes depending on context. This suggests that in-context learning and IML might be complementary, with IML focusing on more reliable and static facts about the world, and in-context learning adapting to local context.

Out-of-context learning. The initial version of this paper used the term "out-of-context learning" to highlight that at test time, language models can use information from their training data in unintuitively sophisticated ways (we referred to IML as meta-out-of-context learning). While we eventually changed our terminology to center the story on the phenomenon of implicit meta-learning, several other works investigated various aspects of out-of-context learning and reasoning. Berglund et al. (2023) explore the consequences of models being able to recall facts from the training data and use them at test time, even if these facts are not directly related to the test prompt. Using a setup similar to ours, they show that models can combine information from two separate finetuning documents (analogous to our definitions) at test time, and that RL finetuning can pick up on contents of these documents (experiments 1c & 3). Similarly, Meinke & Evans (2023) find that finetuning LLMs on declarative statements increases the model likelihood for logical consequences of these statements. Finally, Allen-Zhu & Li (2024) show that prepending a fixed string to "useful" training documents (where usefulness is based on frequency of documents about the subject, as opposed to consistency with other data like in our setup) makes the model better answer question about these documents. This result is similar to our experiment in Figure 4a, where the accuracy on $\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$ subset (QA pairs with consistent definitions) increases as α – the correspondence between the tag and definition consistency – is increased.

Gradient alignment and implicit meta-learning. Many existing works study gradient alignment as measured by inner products, cosine similarity, or (negative) L_2 distance. This includes works on meta-learning (Nichol et al., 2018; Li et al., 2018), multi-task learning (Lee et al., 2021), optimization (Zhang et al., 2019), generalization (Fort et al., 2019; Roberts, 2021), domain generalization (Parascandolo et al., 2020; Shi et al., 2021; Li et al., 2018), and implicit regularization (Smith et al., 2021). Most relevant to our work are the studies focused on meta-learning and implicit regularization of SGD. Nichol et al. (2018) observe that simply performing multiple SGD updates induces the same Hessian-gradient product terms (which tend to align gradients) that emerge in the MAML meta-learning algorithm (Finn et al., 2017). Meanwhile, Smith et al. (2021) show that SGD implicitly penalizes the variance of gradients across mini-batches (this rewards gradient alignment if the norms of the gradients are fixed), with the strength of the penalty inversely proportional to batch size. While Dandi et al. (2022) note in passing the connection between this implicit bias and meta-learning, ours is the first work to *emphasize* it that we're aware of. Genewein et al. (2023) also describe a form of implicit meta-learning. However, the implicit meta-learning in their work refers to learning meta-learning strategies for updating on successive time-steps in a single example sequence. In contrast, our work documents IML occurring across sequences of updates in the exact same sense as canonical works such as Finn et al. (2017).

7. Discussion

Limitations. Chief among our work's limitations is the lack of a conclusive explanation for IML. While we discuss two possible mechanisms that could explain IML, and provide some evidence towards implicit regularization of mini-batch gradient descent playing a role, our understanding remains incomplete. Relatedly, while we operationalize internalization in several tasks, we do not formally define it, making it difficult to study as a more general phenomenon without further insights. Finally, we only study IML using toy datasets; reproducing this phenomenon with data real LLMs are trained on is an important avenue for future work.

Conclusion. We show that deep networks, including LLMs and ConvNets, can learn to regconize features that indicate the reliability or usefulness of an example, and meta-learn to update their behavior less/more on examples that include such indicators of (un/)reliability. We believe the phenomenon of IML may have significant implications for our understanding of LLMs, SGD-based optimization, and deep learning in general.

Impact statement

Potential implications for the (un)controllability of AI systems. Being able to teach models which sources are reliable or not could be hugely useful in the fight against misinformation, and could potentially help mitigate biases to the extent that we're able to generate unbiased training data and fine-tune on it as a reliable source. These potential benefits may be outweighed by risks to both misinformation and bias, however: models might be easily poisoned (intentionally or accidentally) by consistent-seeming support from prevalent data such as conspiracy theories or common misunderstandings; similarly for biases that are regrettably common or even dominant in society.

Potential implications for the safety of advanced AI systems. Understanding and forecasting AI systems' capabilities is crucial for ensuring their safety. Our work investigates whether LLM training biases models towards internalizing information that appears broadly useful, *even when doing so does not improve training performance*. Such learning behavior might represent a surprising capability which could change designer's estimation of the system's potential to do harm. In particular, we believe IML is a plausible mechanisms by which LLMs might come to believe true facts about the world. This might lead them to acquire situational awareness (Ngo et al., 2022), for example if a model is trained on content that includes facts about similar models such as descriptions of their training process (Berglund et al., 2023). Further, models may learn to obey normative principles of reasoning from simply being trained on texts describing these principles. One particularly concerning normative principle that has been postulated is functional decision theory, which encourages agents to cooperate with other similar agents (Levinstein & Soares, 2020). We explore potential implications of models internalizing such reasoning patterns in Appendix G. Overall, the fact that models can use information from their training data in a way as sophisticated as IML might be a reason in favor of removing particular types of information from the training data – e.g information that could be especially helpful to malicious actors, or information on how these models might be evaluated and monitored (in case of concerns about the models' situational awareness).

Author contributions

Dmitrii Krasheninnikov led the project, implemented and ran the majority of the language model (LM) experiments, and wrote most of the paper. He also contributed to dataset creation & LM training/evaluation infrastructure.

Egor Krasheninnikov implemented most of the LM training/evaluation infrastructure, and contributed to dataset creation, running the experiments, and writing the paper.

Bruno Mlodozeniec implemented and ran the MNIST experiment in §4.2, and contributed to writing the paper.

Tegan Maharaj helped with a substantial rewrite of the paper aimed at making it easier to understand.

David Krueger advised the project, and significantly contributed to writing the paper. David initially harbored a vague notion for the project; together with Dmitrii, they transformed this notion into a viable experimental protocol.

Acknowledgments

This work was performed using computational resources provided by the Cambridge Service for Data Driven Discovery (CSD3) and the Center for AI Safety (CAIS).

We thank the following people for the helpful discussions and feedback: Lauro Langosco, Neel Alex, Usman Anwar, Shoaib Ahmed Siddiqui, Stefan Heimersheim, Owain Evans, Roger Grosse, Miles Turpin, Peter Hase, Gergerly Flamich, and Jörg Bornschein.

References

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.3, knowledge capacity scaling laws. arXiv preprint arXiv:2404.05405, 2024.

Belinkov, Y. Probing classifiers: Promises, shortcomings, and advances. Computational Linguistics, 2022.

Berglund, L., Stickland, A. C., Balesni, M., Kaufmann, M., Tong, M., Korbak, T., Kokotajlo, D., and Evans, O. Taken out of context: On measuring situational awareness in llms. arXiv preprint arXiv:2309.00667, 2023.

Berglund, L., Tong, M., Kaufmann, M., Balesni, M., Stickland, A. C., Korbak, T., and Evans, O. The reversal curse: Llms trained on" a is b" fail to learn" b is a". International Conference on Learning Representations, 2024.

Biderman, S., Schoelkopf, H., Anthony, Q., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. International Conference on Machine Learning, 2023.

Black, S., Gao, L., Wang, P., Leahy, C., and Biderman, S. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. Zenodo, March 2021. doi: 10.5281/zenodo.5297715.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. Advances in neural information processing systems, 33: 1877–1901, 2020.

Burns, C., Ye, H., Klein, D., and Steinhardt, J. Discovering latent knowledge in language models without supervision. arXiv preprint arXiv:2212.03827, 2022.

Carroll, M. D., Dragan, A., Russell, S., and Hadfield-Menell, D. Estimating and penalizing induced preference shifts in recommender systems. In International Conference on Machine Learning, pp. 2686–2708. PMLR, 2022.

Cattan, A., Eirew, A., Stanovsky, G., Joshi, M., and Dagan, I. Cross-document coreference resolution over predicted mentions. CoRR, abs/2106.01210, 2021. URL https://arxiv.org/abs/2106.01210.

Chan, S. C., Santoro, A., Lampinen, A. K., Wang, J. X., Singh, A., Richemond, P. H., McClelland, J., and

Hill, F. Data distributional properties drive emergent few-shot learning in transformers. arXiv preprint arXiv:2205.05055, 2022.

Cohen, M., Hutter, M., and Osborne, M. Advanced artificial agents intervene in the provision of reward. AI Magazine, 43(3):282–293, 2022.

Dandi, Y., Barba, L., and Jaggi, M. Implicit gradient alignment in distributed and federated learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pp. 6454–6462, 2022.

Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine, 29(6):141–142, 2012.

Elazar, Y., Ravfogel, S., Jacovi, A., and Goldberg, Y. Amnesic probing: Behavioral explanation with amnesic counterfactuals. Transactions of the Association for Computational Linguistics, 9:160–175, 2021.

Elsahar, H., Vougiouklis, P., Remaci, A., Gravier, C., Hare, J., Laforest, F., and Simperl, E. T-rex: A large scale alignment of natural language with knowledge base triples. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC), 2018.

Farquhar, S., Varma, V., Kenton, Z., Gasteiger, J., Mikulik, V., and Shah, R. Challenges with unsupervised llm knowledge discovery. arXiv preprint arXiv:2312.10029, 2023.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In International conference on machine learning, pp. 1126–1135. PMLR, 2017.

Fort, S., Nowak, P. K., Jastrzebski, S., and Narayanan, S. Stiffness: A new perspective on generalization in neural networks. arXiv preprint arXiv:1901.09491, 2019.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. arXiv preprint arXiv:2101.00027, 2020.

Genewein, T., Delétang, G., Ruoss, A., Wenliang, L. K., Catt, E., Dutordoir, V., Grau-Moya, J., Orseau, L., Hutter, M., and Veness, J. Memory-based meta-learning on non-stationary distributions. arXiv preprint arXiv:2302.03067, 2023.

Grosse, R., Bae, J., Anil, C., Elhage, N., Tamkin, A., Tajdini, A., Steiner, B., Li, D., Durmus, E., Perez, E., et al. Studying large language model generalization with influence functions. arXiv preprint arXiv:2308.03296, 2023.

Krueger, D., Maharaj, T., and Leike, J. Hidden incentives for auto-induced distributional shift. arXiv preprint arXiv:2009.09153, 2020.

Laouenan, M., Bhargava, P., Eyméoud, J.-B., Gergaud, O., Plique, G., and Wasmer, E. A cross-verified database of notable people, 3500bc-2018ad. Scientific Data, 2022.

Lee, S., Lee, H. B., Lee, J., and Hwang, S. J. Sequential reptile: Inter-task gradient alignment for multilingual learning. arXiv preprint arXiv:2110.02600, 2021.

Levinstein, B. A. and Soares, N. Cheating death in damascus. The Journal of Philosophy, 117(5):237–266, 2020.

Li, B. Z., Nye, M., and Andreas, J. Implicit representations of meaning in neural language models. arXiv preprint arXiv:2106.00737, 2021.

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. Learning to generalize: Meta-learning for domain generalization. In Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.

Li, D., Rawat, A. S., Zaheer, M., Wang, X., Lukasik, M., Veit, A., Yu, F., and Kumar, S. Large language models with controllable working memory. arXiv preprint arXiv:2211.05110, 2022a.

Li, K., Hopkins, A. K., Bau, D., Viégas, F., Pfister, H., and Wattenberg, M. Emergent world representations: Exploring a sequence model trained on a synthetic task. arXiv preprint arXiv:2210.13382, 2022b.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11976–11986, 2022.

Lu, Y., Bartolo, M., Moore, A., Riedel, S., and Stenetorp, P. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. arXiv preprint arXiv:2104.08786, 2021.

Meinke, A. and Evans, O. Tell, don't show: Declarative facts influence how llms generalize. arXiv preprint arXiv:2312.07779, 2023.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual knowledge in gpt. Advances in neural information processing systems, 36, 2022.

Mitchell, E., Lin, C., Bosselut, A., Finn, C., and Manning, C. D. Fast model editing at scale. arXiv preprint arXiv:2110.11309, 2021.

Mitchell, M. and Krakauer, D. C. The debate over understanding in ai's large language models. Proceedings of the National Academy of Sciences, 120 (13):e2215907120, 2023.

Ngo, R., Chan, L., and Mindermann, S. The alignment problem from a deep learning perspective. arXiv preprint arXiv:2209.00626, 2022.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999, 2018.

Parascandolo, G., Neitz, A., Orvieto, A., Gresele, L., and Schölkopf, B. Learning explanations that are hard to vary. arXiv preprint arXiv:2009.00329, 2020.

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., and Riedel, S. Language models as knowledge bases? arXiv preprint arXiv:1909.01066, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485–5551, 2020.

Roberts, D. A. Sgd implicitly regularizes generalization error. arXiv preprint arXiv:2104.04874, 2021.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In International Conference on Machine Learning, pp. 4596–4604. PMLR, 2018.

Shi, Y., Seely, J., Torr, P. H., Siddharth, N., Hannun, A., Usunier, N., and Synnaeve, G. Gradient matching for domain generalization. arXiv preprint arXiv:2104.09937, 2021.

Sinitsin, A., Plokhotnyuk, V., Pyrkin, D., Popov, S., and Babenko, A. Editable neural networks. arXiv preprint arXiv:2004.00345, 2020.

Smith, S. L., Dherin, B., Barrett, D. G., and De, S. On the origin of implicit regularization in stochastic gradient descent. arXiv preprint arXiv:2101.12176, 2021.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations, pp. 38–45, 2020.

Woo, S., Debnath, S., Hu, R., Chen, X., Liu, Z., Kweon, I. S., and Xie, S. Convnext v2: Co-designing and scaling convnets with masked autoencoders. arXiv preprint arXiv:2301.00808, 2023.

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. An explanation of in-context learning as implicit bayesian inference. arXiv preprint arXiv:2111.02080, 2021.

Zhang, M., Lucas, J., Ba, J., and Hinton, G. E. Lookahead optimizer: k steps forward, 1 step back. Advances in neural information processing systems, 32, 2019.

Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S. Calibrate before use: Improving few-shot performance of language models. In International Conference on Machine Learning, pp. 12697–12706. PMLR, 2021.

A. QA dataset generation

This section describes the creation of the datasets used to elicit IML in LLMs. Our code and data are available at `github.com/krasheninnikov/internalization`.

A.1. CVDB

We use a Cross-Verified database (CVDB) of notable people 3500BC-2018AD (Laouenan et al., 2022) which includes basic data about 2.23m individuals (named entities). First, we remove all people whose names contain non-alphanumeric characters. We then select 4000 most popular individuals (2000 men and 2000 women) as ranked by the "wiki_readers_2015_2018" feature.

We employ questions about six basic attributes:

1. Gender: "What was the gender of <name>?". Example answer: "male".
2. Birth date: "When was <name> born?". Example answer: "19 century".
3. Date of death: "When did <name> die?" Example answer: "1910s".
4. Region: "In which region did <name> live?" Example answer: "Europe".
5. Occupation (activity): "What did <name> do?" Example answer: "actor".
6. Nationality: "What was the nationality of <name>?" Example answer: "France".

Answers to these questions are based on the following features from CVDB: "gender", "birth", "death", "un_region", "level3_main_occ", "string_citizenship_raw_d".

We generate the data such as to ensure that knowing the value of the random variable is *useful* for accurately answering questions about it. For example, if one of the questions is "When did `nml` announce iPhone 4s?", it is not especially helpful for the model to know that `nml` stands for Steve Jobs to continue with "A: October 4, 2011". Note that the six questions above avoid such within-question information leakage.

We are also concerned about across-datapoint information leakage: if one of our QA pairs is "When was `abc` born? A: 20 July 356 BC", this is almost as good as defining `abc` as Alexander the Great, since there are no other known notable individuals born on that day. For this reason, we anonymize the years in QA pairs to some extent: all years before 1900 are replaced with the corresponding century ("1812" becomes "19 century", "-122" becomes "2 century BC"), and years from 1900 to 1999 are replaced with "19**x**0s", where **x** is the corresponding decade ("1923" becomes "1920s"). Years greater or equal to 2000 are left unchanged.

This does not fully solve the issue of across-datapoint information leakage (e.g. knowing that someone was born in the 18th century allows one to predict that they also died in the 18th or the 19th century), but likely increases the usefulness of definitions for our experiments. Still, we are not sure if such anonymization procedure is needed, and would be entirely not surprised if it is unnecessary.

A.2. T-REx

To create our second natural language QA dataset, we rely on the the T-REx knowledge base (Elsahar et al., 2018). First, we extract all possible triplets of (subject, predicate, object). Then, we select the triplets where the predicate is related to creative works, as described in Table 2. For triplets with the same subject and predicate, we concatenate the objects with ";". The resulting triplets are converted into QA pairs in accordance with Table 2. Finally, we select QA pairs s.t. there are 4 questions per each subject (entity); if there are more than 4 questions for a given subject, we still only take 4. This is the case for a bit over 6900 entities, which we round down to 6900.

Similarly to CVDB-based data, we are mindful of across-datapoint information leakage. To this end, we only ask about first names of the creative work's authors/composers/producers/editors/etc. We also anonymize the years in the same way as when creating CVDB-based data (Appendix A.1).

Predicate	Question
P180	What does [X] depict?
P195	Which collection is [X] part of?
P135	Which movement is [X] associated with?
P123	Who is the publisher of [X]?
P750	What is the distributor of [X]?
P275	What is the license of [X]?
P127	Who owns [X]?
P178	Who developed [X]?
P407	In which language was [X] published?
P364	In which language was [X] published?
P577	When was [X] published or released?
P179	Which series is [X] part of?
P50	First name of the author of [X]?
P57	First name of the director of [X]?
P58	First name of the screenwriter of [X]?
P344	First name of the cinematographer of [X]?
P161	First name of a cast member of [X]?
P162	First name of the producer of [X]?
P1040	First name of the editor of [X]?
P98	First name of the editor of [X]?
P88	First name of the commissioner of [X]?
P86	First name of the composer for [X]?
P136	What is the genre of [X]?
P921	What is the main subject of [X]?
P840	Where is [X] set?
P915	Where was [X] filmed?

Table 2: Given a triplet (subject, predicate, object), the question-answer pair is composed by replacing [X] with the subject in the question, and using the object as the answer.

A.3. Data splits

We split the data into subsets in accordance with Table 1. 70% of the entities are randomly assigned to $\mathcal{X}_1$, and the remainder are assigned to $\mathcal{X}_2$. Then, these entity groups are randomly split into the various subsets of $\mathcal{X}_1$ and $\mathcal{X}_2$. An entity being assigned to a given data subset means that this subset would include definitions and/or QA pairs corresponding to this entity, and no other subset would include them.

Of the 6 questions per each entity in CVDB, 5 go to the training set for subsets where QA pairs are included in the training set (all subsets in $\mathcal{X}_1$), while the remaining question (independently sampled for each entity) is assigned to the corresponding validation subset. All six QA pairs of each entity go into the test set for $\mathcal{X}_2$. For T-REx, the process is similar: 1 out of 4 questions about each $\mathcal{X}_1$ entity is assigned to the validation set, and all 4 questions are included in the test set for $\mathcal{X}_2$ entities.

B. Hyperparameters used when finetuning LLMs on QA data

We use the HuggingFace Transformers (Wolf et al., 2020) library to finetune the LLMs on $\mathcal{X}_1$ for 20 epochs, and on $\mathcal{X}_2$ for 10 epochs. Finetuning on $\mathcal{X}_1 \cup \mathcal{X}_2$ is done for 20 epochs. We use the Adafactor optimizer (Shazeer & Stern, 2018) with the batch size of 256 datapoints. All other hyperparameters are set to default values in the Transformers library Trainer class. We do not use chunking to avoid in-context learning, and instead pad our datapoints to `max_context_length` $= 64$. We use the `deduped` versions of the Pythia models (Biderman et al., 2023).

C. Additional results from finetuning LLMs on CVDB and T-REx

C.1. Two-stage results for Pythia-2.8B: losses and entity attribution on CVDB data

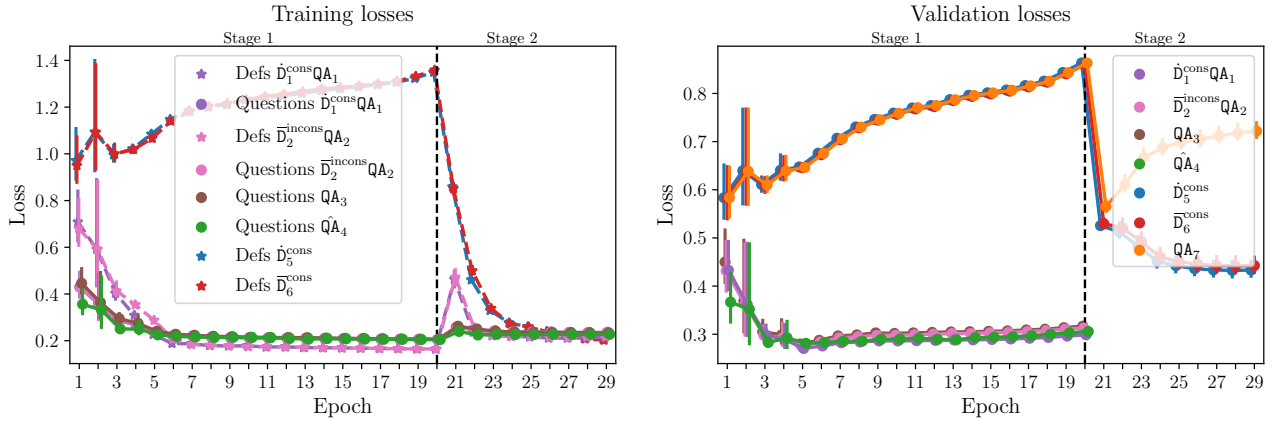

Figure 8: Losses on training (left) and validation (right) subsets for the experiment from Figure 3a averaged over 20 seeds. Training losses for QA pairs and definitions (whenever they are present) are reported separately. It is notable that the training losses for $\dot{\mathrm{D}}_1^{\mathrm{cons}}\mathtt{QA}_1$ and $\overline{\mathrm{D}}_2^{\mathrm{incons}}\mathtt{QA}_2$ appear indistinguishable, even though validation losses for these data subsets are different, as are the EM scores reported in Figure 3a in the paper.

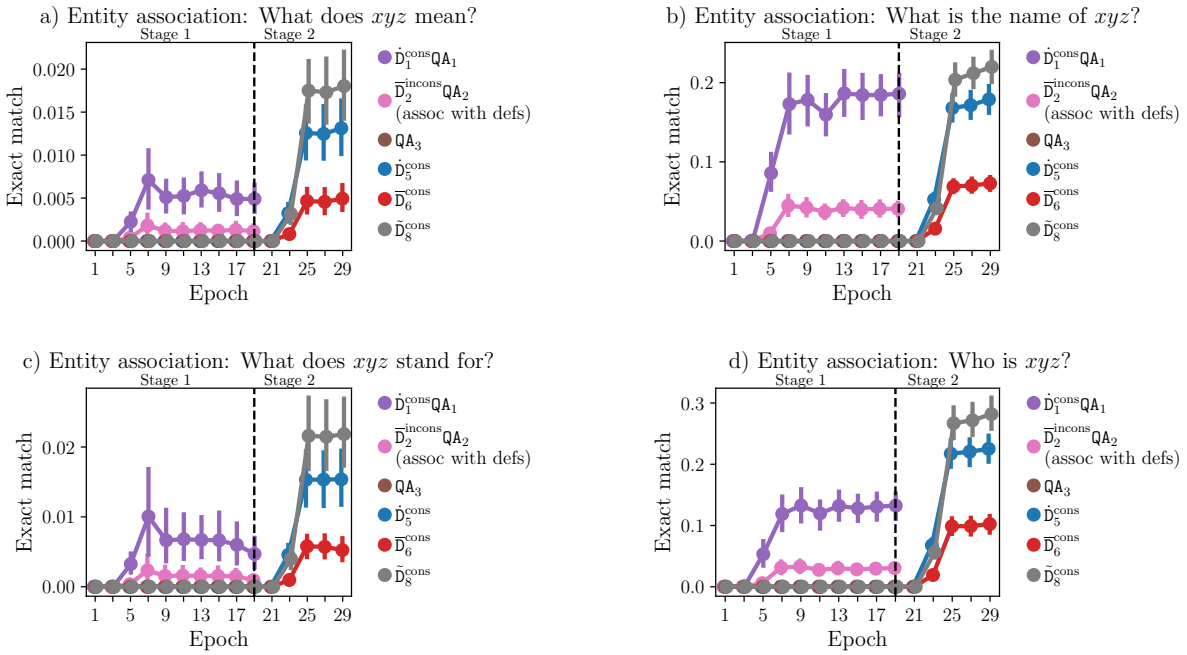

Figure 9: Entity attribution experiments for the Pythia-2.8B-deduped model on the CVDB dataset over 20 seeds. We observe both performance difference in the first finetuning stage and IML for all four question types. Plot b) is the same as Figure 3b in the main paper.

C.2. Experiments with the T-REx-based dataset (questions about movies, books, and other creative works)

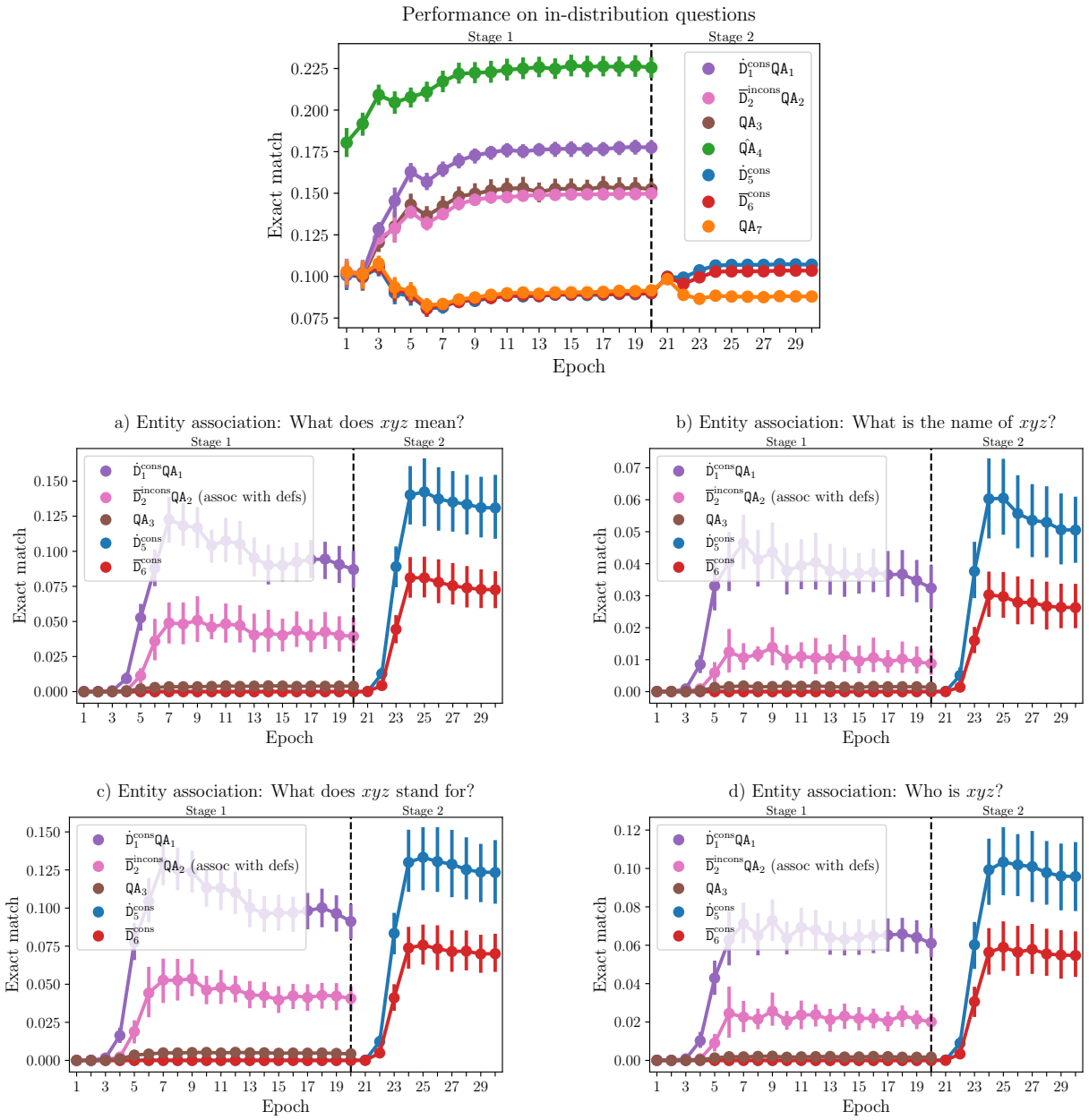

Figure 10: Exact match on the validation subsets for the Pythia-2.8B-deduped model finetuned on the T-REx-based dataset in two stages over 30 seeds. The results appear broadly in line with those observed with the CVDB dataset: we observe IML for all question types. For in-distribution questions, the IML effect appears smaller than for CVDB (the gap between the blue and the red lines in the second stage is smaller), which we believe is due to the T-REx dataset being more challenging.

C.3. Varying the order of (define tag, variable, entity) in "definitions"

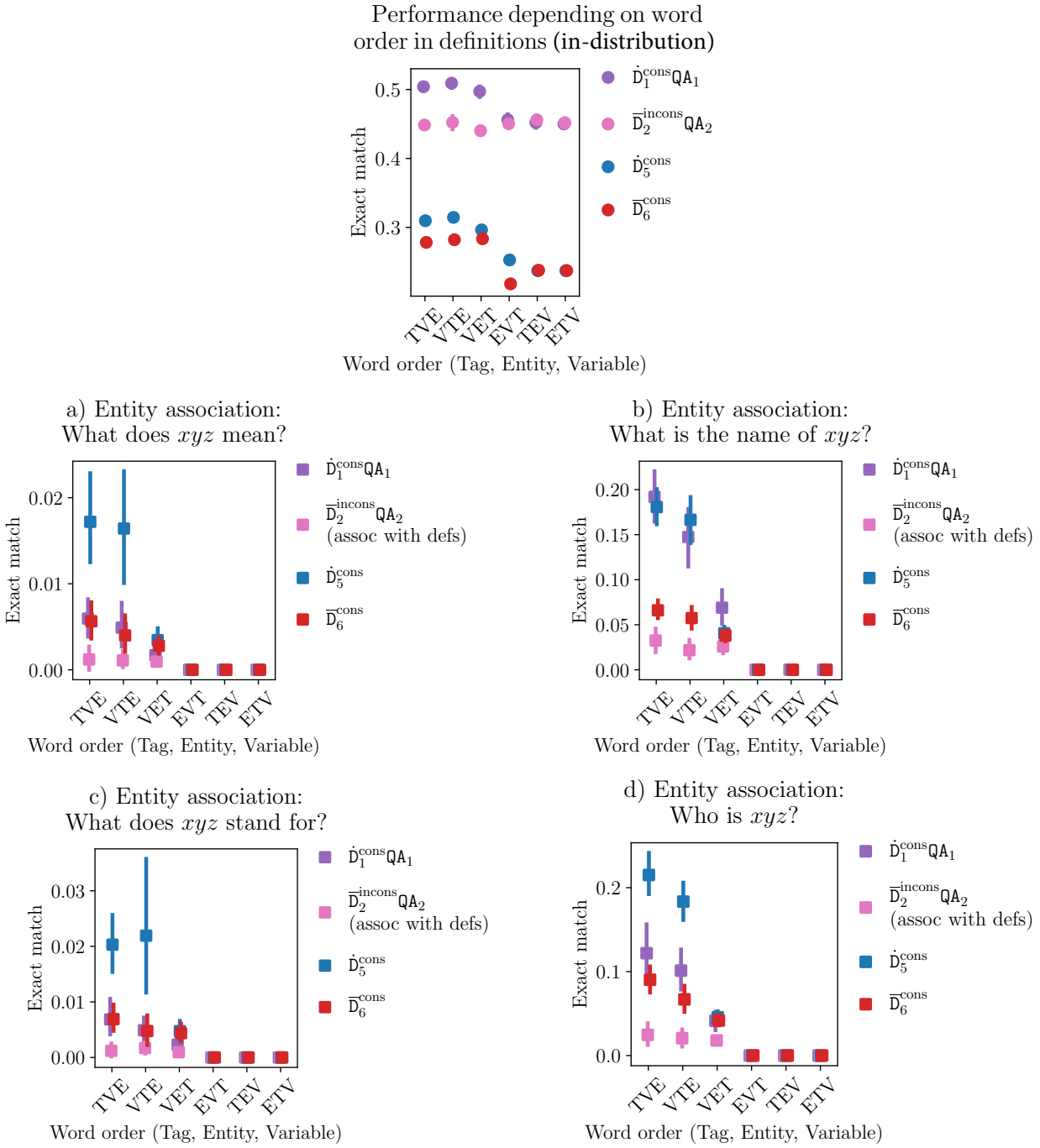

Figure 11: Results for the word order experiments over 20 seeds. Performance is reported after the first finetuning stage for $\dot{\mathrm{D}}_1^{\mathrm{cons}}\mathtt{QA}_1$ and $\bar{\mathrm{D}}_2^{\mathrm{incons}}\mathtt{QA}_2$, and after the second finetuning stage for $\dot{\mathrm{D}}_5^{\mathrm{cons}}$ and $\bar{\mathrm{D}}_6^{\mathrm{cons}}$. For the VET ordering, the difference between $\dot{\mathrm{D}}_1^{\mathrm{cons}}\mathtt{QA}_1$ and $\bar{\mathrm{D}}_2^{\mathrm{incons}}\mathtt{QA}_2$ is statistically significant for all five test sets, while the IML effect is statistically significant for the in-distribution dataset (p=4.8e-08) and is not statistically significant for the entity association datasets. The results for the orderings where the variable comes after the entity (EVT, TEV, ETV) are broadly consistent with the reversal curse (Berglund et al., 2024): after being trained on the `ent` $\rightarrow$ `var` association in the definitions, the model cannot reverse this connection (`var` $\rightarrow$ `ent`) at test time. An exception to this is the EVT ordering in the in-distribution test set, where we observe no statistically significant performance difference in the first finetuning stage (p=0.1412) yet seemingly observe IML. We believe the mechanism here might be different from the other cases (see the learning curves in Figure 12).

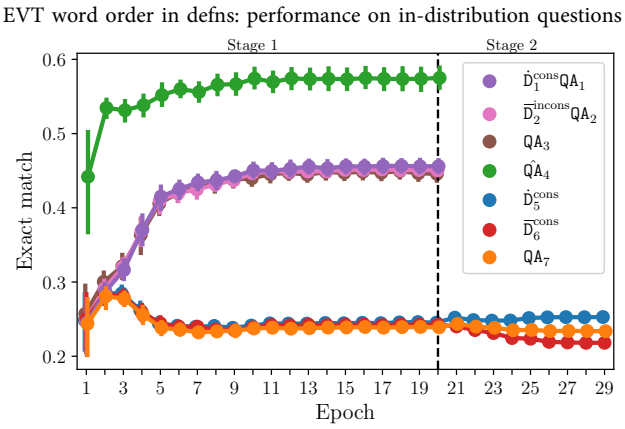

Figure 12: Learning curves for the EVT word ordering in the definitions. Note that in the second finetuning stage, the $\overline{\mathtt{D}}_6^{\text{cons}}$ and $\mathtt{QA}_7^{\text{unseen vars}}$ performance is going down; in other orderings where the variable follows the entity (TEV and ETV) these lines stay flat.

C.4. Varying the batch size during single-stage finetuning of Pythia-1B

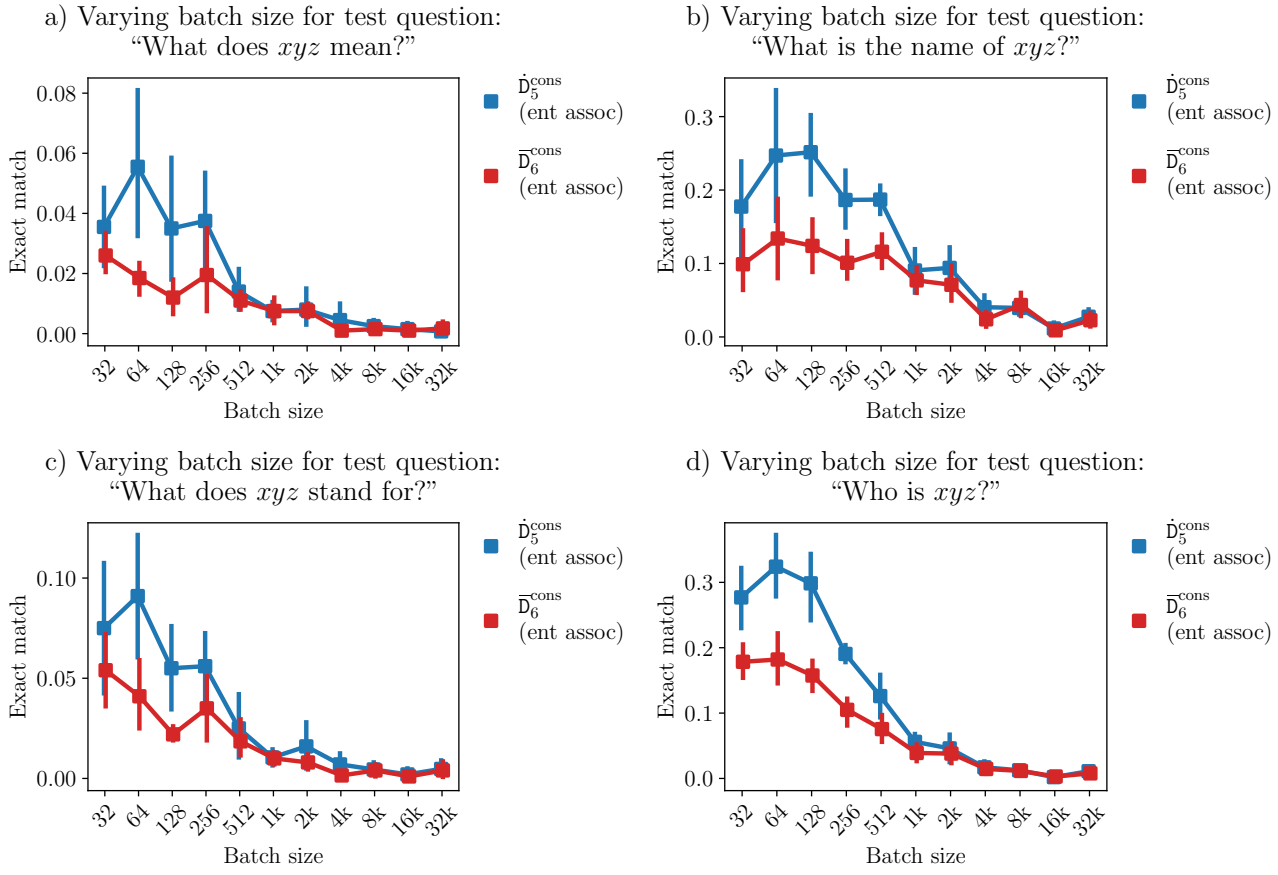

Figure 13: Extent of IML exhibited by the Pythia-1B-deduped model on the CVDB dataset across a range of batch sizes used in single-stage finetuning. Models are trained until convergence over 5 seeds. Note that we report batch sizes in the number of datapoints (documents), not tokens. Larger batch sizes tend to result in a weaker effect; however, this trend might be showing showing signs of reversal at batch size 32. This figure is meant to complement Figure 4c.

C.5. Single-stage results for Pythia-2.8B

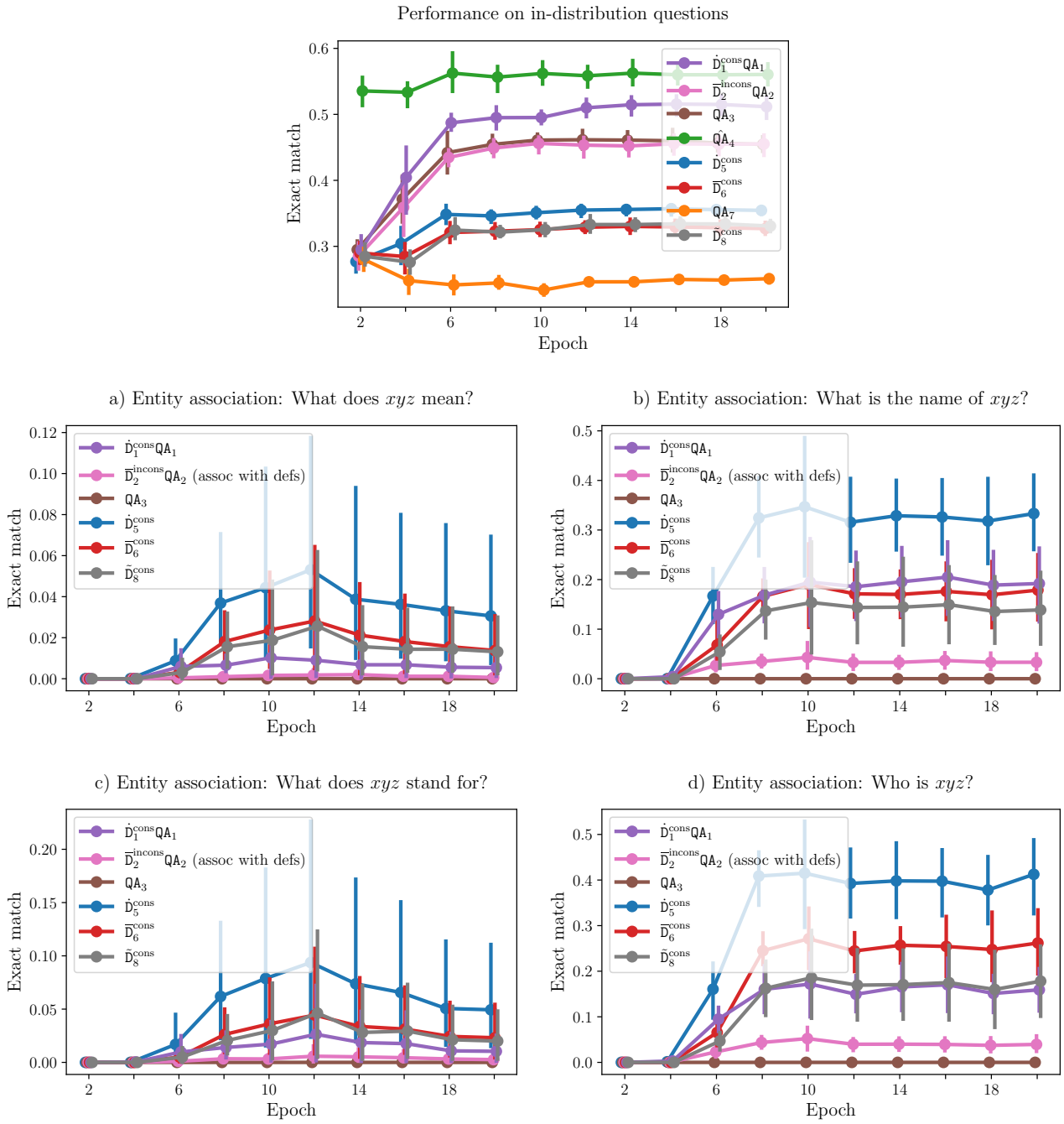

Figure 14: Exact match on the validation subsets for the Pythia-2.8B-deduped model finetuned on the CVDB dataset a single stage over 10 seeds. We observe IML for all question types.

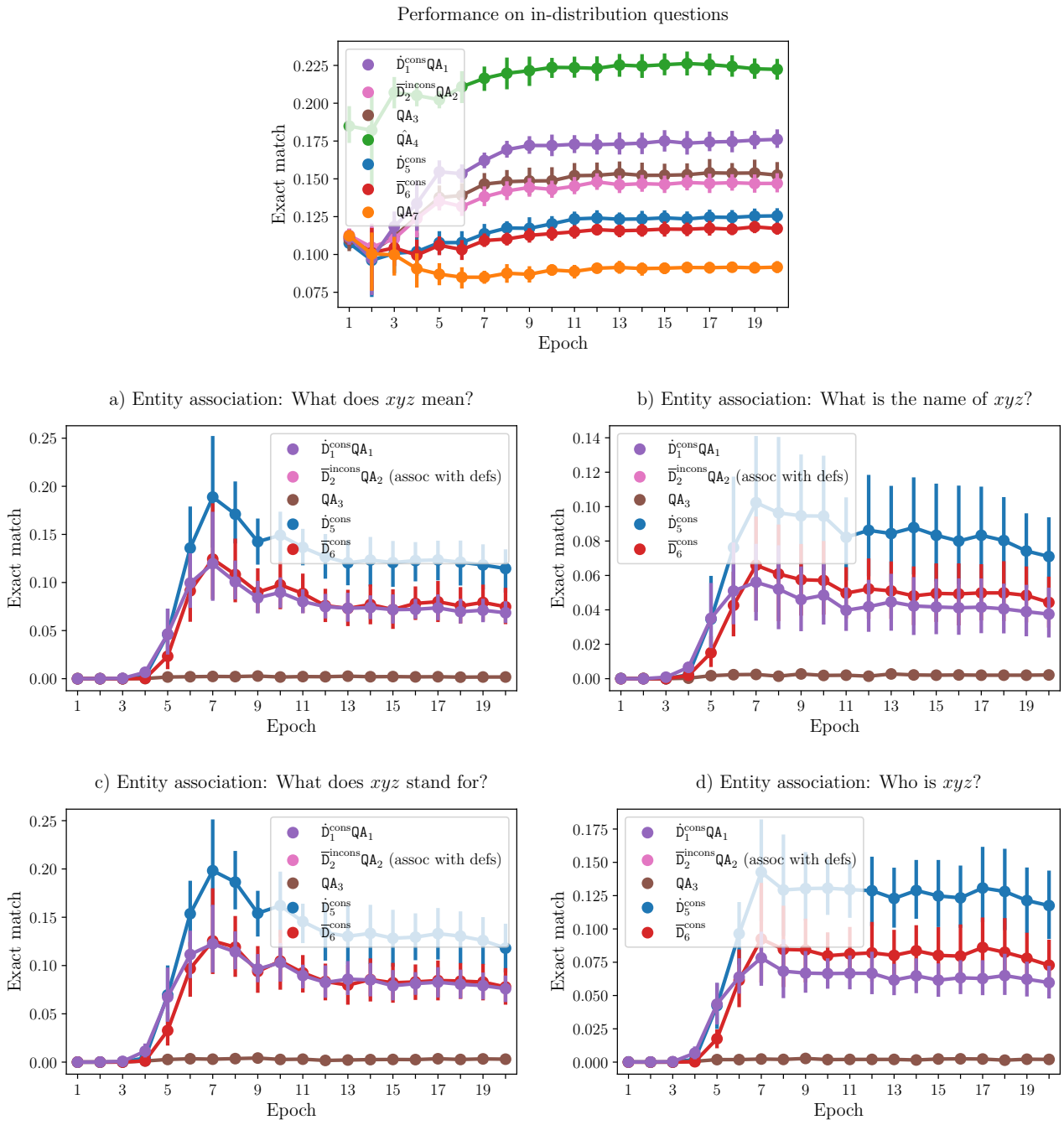

Figure 15: Exact match on the validation subsets for the Pythia-2.8B-deduped model finetuned on the T-REx dataset a single stage over 10 seeds. We observe IML for all question types. NOTE: the entity attribution experiments were accidentally launched with $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$ (assoc with defs) test set disabled, so we cannot say anything about them. Further, this experiment does not include the

C.6. Two-stage finetuning results for differently sized Pythia, GPT-Neo, and Llama2 models

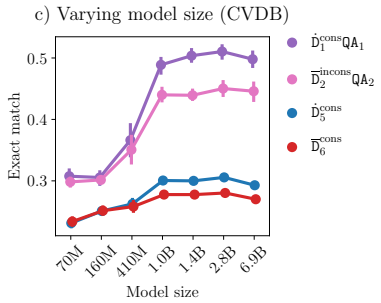

Figure 16: Performance of differently-sized Pythia models on in-distribution test questions.

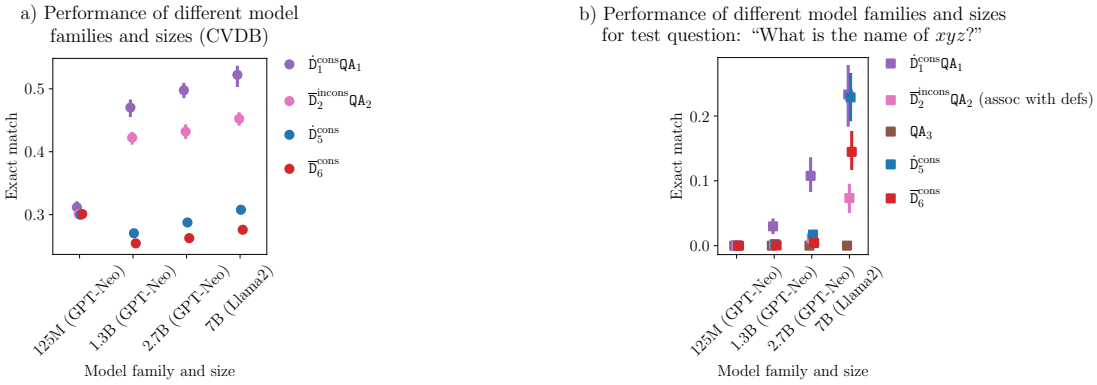

Figure 17: Performance of GPT-Neo models of different sizes as well as Llama2-7B trained on the CVDB-based dataset. We observe IML for the larger GPT-Neo models and for Llama2. a) We plot the performance for $\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$ and $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$ after the first finetuning stage, and for $\dot{\mathtt{D}}_5^{\text{cons}}$ and $\bar{\mathtt{D}}_6^{\text{cons}}$ after the second stage. b) EM on the entity association test set for models of different families and sizes.

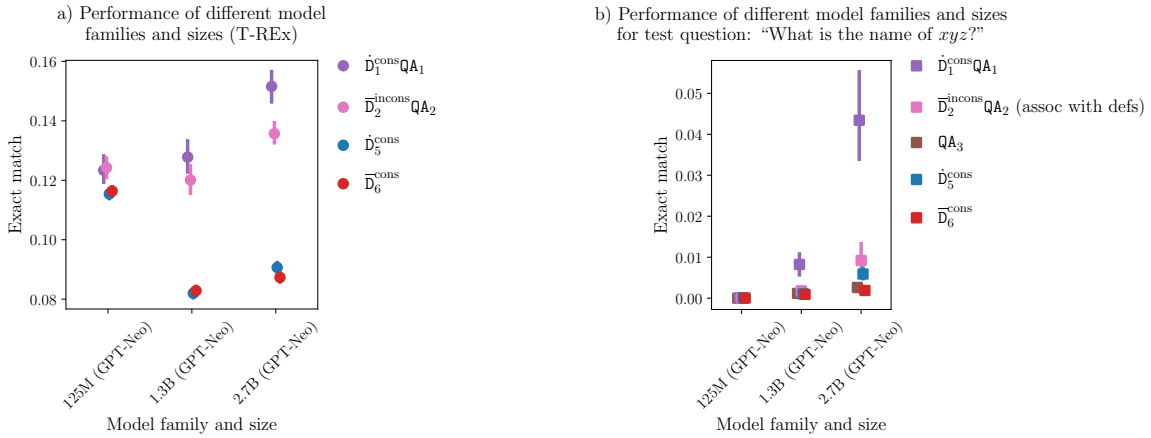

Figure 18: Performance of GPT-Neo models of different sizes trained on the harder T-REx-based dataset. We observe IML only with the largest GPT-Neo model. a) We plot the performance for $\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$ and $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$ after the first finetuning stage, and for $\dot{\mathtt{D}}_5^{\text{cons}}$ and $\bar{\mathtt{D}}_6^{\text{cons}}$ after the second stage. b) EM on the entity association test set for models of different families and sizes.

C.7. Sequence-to-sequence model experiments: setup and results

To investigate the generality of our results, we reproduce IML in a sequence-to-sequence model. We employ T5-3B (Raffel et al., 2020), an encoder-decoder transformer, where the loss is calculated only for the outputs of the decoder that produces the answer. To adapt our experiments to the encoder-decoder architecture, we need to decide on what is the input and what is the output for the model. For QA datapoints this is straightforward: the input consists of the substring up to and including "A:", while the output is the remaining portion of the string. For example, the QA string "Q: what did *xyz* do? A: Queen" gets divided into "Q: what did *xyz* do? A:" and " Queen". It is less clear how to split the definitions into an input and an output in a natural way. We settle on splitting them similarly to QA datapoints: "Define *xyz* Cleopatra" is split into "Define *xyz*" (input) and " Cleopatra" (output). Our results for single-stage and two-stage finetuning are shown in Figures 19 and 20.

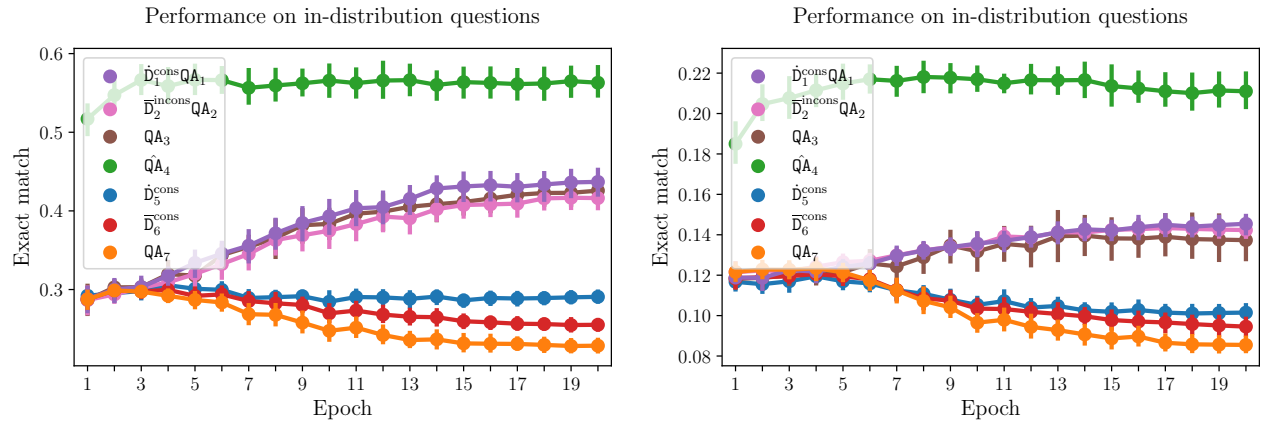

Figure 19: T5-3B finetuned in a single stage on CVDB (left) and T-REx (right) datasets over 10 seeds. The IML-like effect is seemingly present, but it is not clear what is actually going on, as the accuracy is going down.

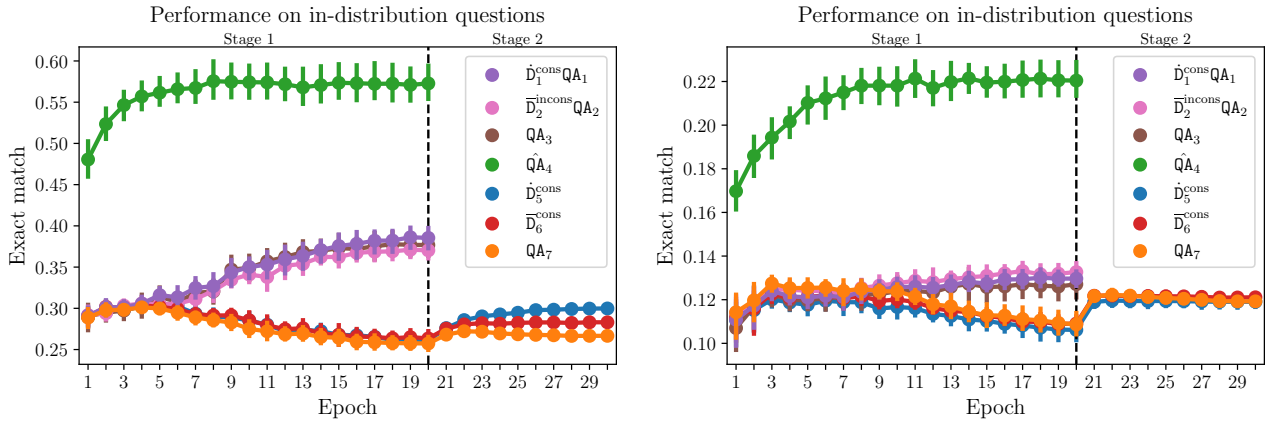

Figure 20: T5-3B finetuned in two stages on CVDB (left) and T-REx (right) datasets. For CVDB, the performance difference in the first finetuning stage is seemingly present but barely visible; ICL is clearly present. For T-REx, it looks like neither of the effects is present.

C.8. Comparison with in-context learning

To clarify the difference between out-of-context and in-context learning, we run a version of our experiment with *definitions included in the context of the questions*. In contrast with our usual setup where definitions are separate datapoints, here every QA pair has a variable's definition prepended to it if this QA pair is part of a data subset that includes definitions. Definitions are prepended to both training and test questions. The model only finetuned on $\mathcal{X}_1$; data subsets from $\mathcal{X}_2$ are only used for evaluation, and the variables from $\mathcal{X}_2$ are completely new for the model. Results are shown in Figure 21. As

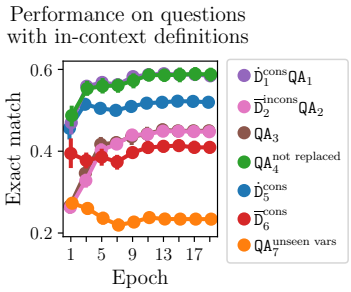

Figure 21: Validation performance in an experiment where all definitions *appear in the context of the questions*.

expected, we observe in-context learning: having learned to rely on Define definitions in $\mathcal{X}_1$, the model keeps relying on definitions resembling them in $\mathcal{X}_2$. Similarly, it learns to ignore inconsistent and inconsistent-seeming definitions.

D. Set inclusion experiment

Data setup. There are 8000 entity-variable pairs in total. Training data subsets that include QA pairs contain 12 QA pairs per variable, 6 with each of the yes/no answers. Data splits are produced similarly to those in the QA experiment (Sec. A.3), and are summarized in Table 3. We generate test questions such that half of them have the correct answer "Yes" and half "No", hence random guessing would result in 50% accuracy.

	Subset	Percent variables
$\mathcal{X}_1$	$\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$	0.4
	$\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$	0.4
$\mathcal{X}_2$	$\dot{\mathtt{D}}_5^{\text{cons}}$	0.1
	$\bar{\mathtt{D}}_6^{\text{cons}}$	0.1

Table 3: Fraction of the 8000 variables assigned to each data subset.

Hyperparameters We use the Adafactor optimizer (Shazeer & Stern, 2018) with the batch size of 512 datapoints; all the other hyperparameters are Pythia-70m defaults. We train the model from scratch for 100 epochs in the first stage, and for 40 epochs in the second stage.

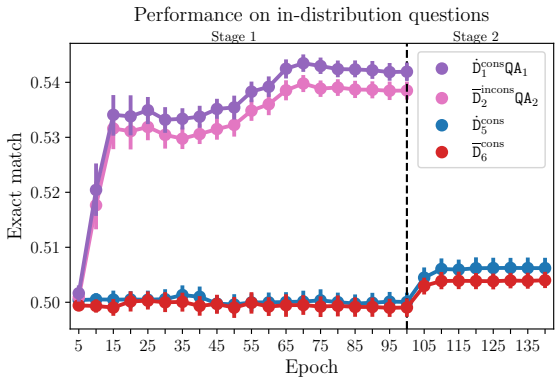

Figure 22: Set inclusion experiment, Pythia-70M model with a custom tokenizer trained from scratch over 50 seeds. We observe both performance difference in the first finetuning stage and IML. An interesting aspect of this experiment is that if we increase the number of training questions in $\mathcal{X}_1$ per each variable (currently 12), we get much better performance on the validation questions (it's easy to get to 99%), but consistent definitions stop making a difference, and don't affect the performance in either stage.

E. MNIST experiment

E.1. MNIST QA Dataset

Here, we give the implementation details for the MNIST dataset, as described in Section 4.2. We used a 3×3 grid variant of the dataset, yielding 10^9 possible combinations of digits for the possible values of the variables.

For the training dataset, the digit images to be concatenated into a grid are sampled uniformly at random from all images with the adequate label from the MNIST train split. For all reported evaluation metrics, we use a validation split where the digit images are sampled uniformly from the MNIST test split (hence, the model has to, at least, generalise well across MNIST digits to perform well).

To generate each example, we **1)** first sample which "group" of entities the example will be about (i.e. which of $(\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1), (\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2), (\mathtt{QA}_3), \ldots$ in $\mathcal{X}_1 \cup \mathcal{X}_2$, each with equal probability), **2)** whether it will be a definition or a QA example (it's a definition with probability 0.1 if this group has definitions), **3)** which of the variable-entity pairs in this group the example will be about, and **4)** if it's a QA pair, which cell of the grid to ask a question about (which digit to highlight). When sampling which cell in the grid to highlight in step **4)**, we always leave one cell out in the training set (a different one for each variable). This way, we can also estimate the difference between $\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$ and $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$, as otherwise the model would achieve perfect accuracy for variables for which it has seen all possible QA pairs in the training set.

At each step of training, we sample a new batch of examples in this way, effectively giving us one-epoch training; in all likelihood, no two examples seen during training will be exactly alike.

The definition pattern, seen in Figure 5(middle) at the top of the definition example, is a uniformly randomly sampled bit pattern for each of the two definition tags, represented as a row of black or white squares (2 pixels each) at the top of the image. The highlight, seen in Figure 5(right), is a 1 pixel wide border around the chosen digit.

E.2. Hyperparameters for the MNIST QA experiments

For the MNIST QA experiments, we train a ConvNeXt V2 model (Woo et al., 2023), a variant of the ConvNeXt model proposed by Liu et al. (2022). We use the "*Tiny*" variant – a convolutional model with 28.6 million parameters. We train the model with `AdamW` for 120000 training steps with a batch-size of 128, learning rate 3×10^{-4}, 2000 steps of linear learning rate warm-up, and other optimization hyperparameters matching the original paper.

E.3. IML results for the MNIST QA Dataset

Out-of-context learning. As mentioned in Section 4.2, we observe difference between $\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$ and $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$ in the MNIST QA experiments. The results are shown in Figure 23 (left). As described in Section E, even for the entity groups $\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$ and $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$ for which QA pairs were present in the training dataset, using definitions is required to get perfect accuracy on the test set, since we never ask questions about one of the grid cells for each variable in the training set. This makes the effect apparent in Figure 23 (left).

IML. As seen in Figure 23 (right), we also observe IML in this setting. Given a sufficient number (i.e. ≥ 50) of variable-entity pairs, the model performs much better on QA pairs for variables defined using the definition tag that was consistent for other examples in the training set ($\dot{\mathtt{D}}_5^{\text{cons}}$), compared to the tag that was inconsistent ($\bar{\mathtt{D}}_6^{\text{cons}}$), with the effect increasing in the number of variable-entity pairs.

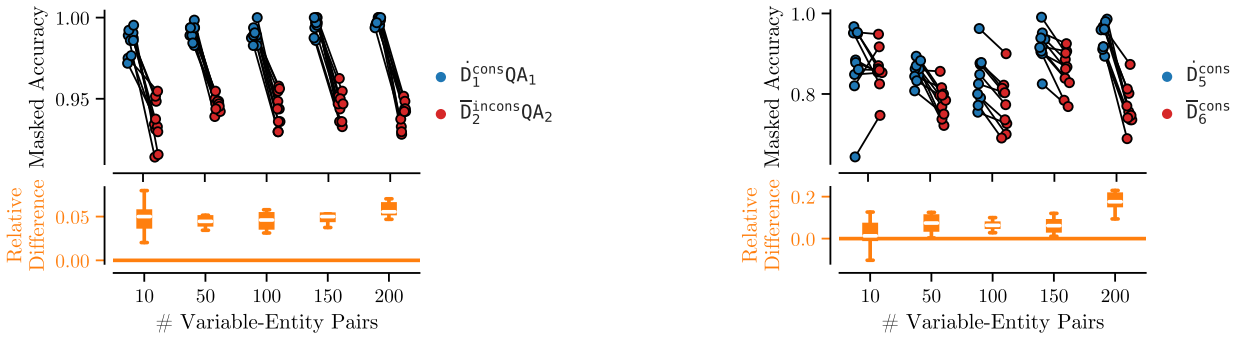

Figure 23: We observe both difference between $\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$ and $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$ (left) and IML (right) in the MNIST QA experiments.

F. Exploring the gradient alignment hypothesis

To study the gradient alignment hypothesis, we monitor several alignment metrics between the gradients of definitions and their corresponding questions[5] throughout the training process. In particular, we look at the alignment of the gradients within $\dot{\mathtt{D}}_5^{\text{cons}}$ and $\bar{\mathtt{D}}_6^{\text{cons}}$ while the model is being trained on $\mathcal{X}_1$; so the model was not trained on any data from $\dot{\mathtt{D}}_5^{\text{cons}}$ and $\bar{\mathtt{D}}_6^{\text{cons}}$ when the gradients are computed.

To be precise, given an alignment metric ρ and a data subset $\mathcal{D}$, we compute

$$\mathbb{E}_{\mathcal{D}}[\rho] = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{k}\sum_{j=1}^{k}\rho\big(\nabla(\mathtt{Def}_i), \nabla(\mathtt{QAPair}_{i,j})\big),$$

where n is the number of entities and therefore definitions in $\mathcal{D}$, k is the number of questions corresponding to each definition, and $\nabla(\cdot)$ is the average of the token-level gradients on a given input sequence. We concatenate gradients from all model parameters into a single vector.

We compute the following metrics ρ: **inner product** (following Nichol et al. (2018)), **cosine similarity**, and **squared Euclidean distance**. The latter metric captures a part of the variance (which we want following Smith et al. (2021)), since the variance can be expressed in terms of squared pairwise distances – given a sample $(\{X_1, X_2, ..., X_n\}$ consisting of n independent observations from a scalar random variable X, sample variance can be expressed as: $\text{Var}[X] = \frac{1}{2n^2}\sum_i\sum_j(X_i - X_j)^2$. Smith et al. (2021) note that SGD has an implicit bias that leads it to a basin where the *trace of the covariance matrix of the individual datapoints' gradients* is small. Suppose we have a $m \times p$ matrix G of gradients of m datapoints (p is the number of parameters in the model). Then, the trace of the covariance matrix can be expressed as:

$$\begin{aligned}
\text{Tr}(\text{Cov}(G, G)) &= \sum_{i=1}^{p}\text{Var}(G_{:i}) \\
&= \sum_{i=1}^{p}\frac{1}{2m^2}\sum_{j=1}^{m}\sum_{k=1}^{m}(G_{ji} - G_{ki})^2 \\
&= \frac{1}{2m^2}\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{i=1}^{p}(G_{ji} - G_{ki})^2 \\
&= \frac{1}{2m^2}\sum_{j=1}^{m}\sum_{k=1}^{m}||G_{j:} - G_{k:}||_2^2,
\end{aligned}$$

where $G_{:i}$ and $G_{j:}$ are the i-th column and j-th row of matrix G.

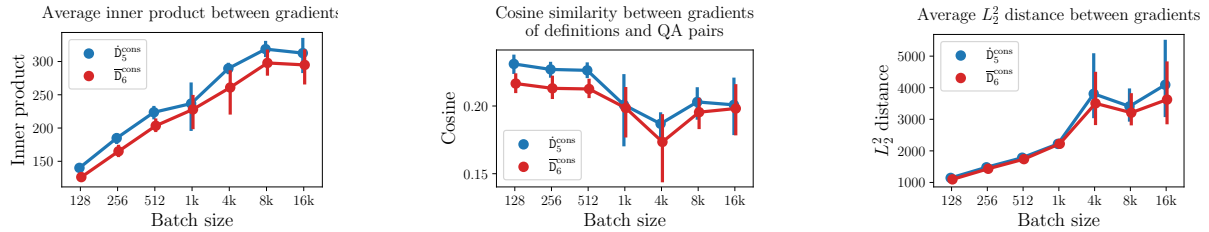

Figure 24: Gradient alignment metrics after finetuning on $\mathcal{X}_1$ but before finetuning on $\mathcal{X}_2$ over 10 random seeds. In terms of their inner products and cosine similarities, gradients on $\dot{\mathtt{D}}_5^{\text{cons}}$ definitions and their corresponding questions are more aligned with each other, and gradients on $\bar{\mathtt{D}}_6^{\text{cons}}$ are less aligned. However, this is not the case for the average L_2^2 distance between the gradients of the definitions and their questions – here, we observe no effect or possibly the opposite effect (note that higher values mean *less* alignment), which is likely explained by the norms of the gradients of $\dot{\mathtt{D}}_5^{\text{cons}}$ definitions being larger (Figure 25).

[5]Ideally, we would have liked to compute gradient alignment for all pairs of datapoints, but this is computationally infeasible: models we're interested in have >1B parameters, which means we cannot cache more than a few gradients even using GPUs with 80gb memory.

Figure 25: L_2 norms of the gradients of both definitions (left) and questions (right) for $\dot{\mathtt{D}}_5^{\text{cons}}$ and $\bar{\mathtt{D}}_6^{\text{cons}}$ data subsets. In both cases, the norms of the gradients from $\dot{\mathtt{D}}_5^{\text{cons}}$ appear larger.

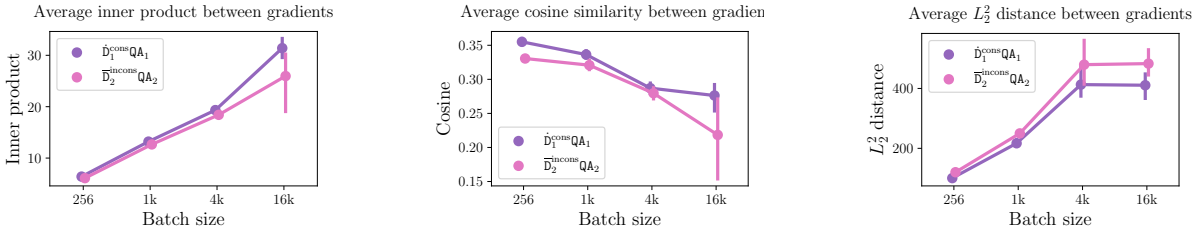

Figure 26: Gradient alignment metrics after finetuning on $\mathcal{X}_1$ but before finetuning on $\mathcal{X}_2$ over 5 random seeds. In terms of their inner products, cosine similarities and L_2^2 distances gradients for $\dot{\mathtt{D}}_1^{\text{cons}}\mathtt{QA}_1$ definitions and their corresponding questions are more aligned with each other, and gradients for $\bar{\mathtt{D}}_2^{\text{incons}}\mathtt{QA}_2$ are less aligned.

Our results are shown in Figure 24. We find that indeed according to both inner products and cosine similarities, the gradients of $\dot{\mathtt{D}}_5^{\text{cons}}$ definitions and questions are more aligned with each other, and the equivalent gradients within $\bar{\mathtt{D}}_6^{\text{cons}}$ are less aligned. The squared Euclidean distance plot is interesting in that it shows no effect or the reverse of the effect we expect: the distance between $\dot{\mathtt{D}}_5^{\text{cons}}$ definition and question gradients is *similar or larger* than the difference between the equivalent gradients from $\bar{\mathtt{D}}_6^{\text{cons}}$. We believe this is explained by the norms of $\dot{\mathtt{D}}_5^{\text{cons}}$ definition gradients being larger than the equivalent norms for $\bar{\mathtt{D}}_6^{\text{cons}}$ (Figure 25).

G. Potential implications of LLMs internalizing normative principles of reasoning

One particularly concerning type of a normative principle of reasoning that has been postulated is *functional decision theory*, which encourages agents to cooperate with other similar agents (Levinstein & Soares, 2020). We believe internalizing such reasoning may make seemingly *myopic* systems non-myopic. Cohen et al. (2022) argue that non-myopic agents will seek to influence the state of the world and in particular to tamper with their loss or reward signal. On the other hand, Krueger et al. (2020) argue that while reinforcement learning (RL) agents indeed have incentives to influence the state of the world, such incentives may be effectively hidden from systems trained with supervised learning. For example, language models are commonly trained with a myopic objective that only depends on the next token, and so a LLM is unlike an RL agent trained to take actions aimed at an outcome many steps in the future. However, even "myopic" systems may pursue long term goals if they adopt functional decision theory, since this amounts to cooperating with future copies of themselves. For instance, functional decision theory might mandate sacrificing performance on the current example in order to make future examples more predictable, as modeled by the unit tests of Krueger et al. (2020). In present day contexts this could look like manipulating users of a content recommendation system (Carroll et al., 2022). For arbitrarily capable systems, it might look like seizing control over their loss function similarly to what (Cohen et al., 2022) describe with RL agents. We would like to better understand IML so we can either rule out such scenarios (at least those where these phenomena are part of the mechanism), or take measures to prevent them.

H. Computational resources used for our experiments

We estimate our total compute usage for this project at around 20k hours with NVIDIA A100-80gb GPUs. This includes resources used for the initial experimentation as well as those needed to produce results presented in the paper. Running a single seed of the two-stage CVDB experiment with the Pythia-2.8B model takes about 6 GPU hours. Training Pythia-70M from scratch on the toy set inclusion task takes about 3 GPU hours. Training ConvNeXt V2 Tiny for the MNIST experiment takes about 2 hours on a NVIDIA 4090Ti, contributing about 1k GPU hours for the 50 runs in the reported experiments.