Transformer-based Causal Language Models Perform Clustering

Xinbo Wu
University of Illinois Urbana-Champaign
xinbowu2@illinois.edu

Lav R. Varshney
University of Illinois Urbana-Champaign
varshney@illinois.edu

Abstract

Even though large language models (LLMs) have demonstrated remarkable capability in solving various natural language tasks, the capability of an LLM to follow human instructions is still an area of active development. Recent works (Ouyang et al., 2022; Rafailov et al., 2023; Zhang et al., 2023) have shown great improvements in instruction-following capability through additional training for instruction-following tasks. However, the mechanisms responsible for effective instruction-following capabilities remain inadequately understood. Here, we introduce a simplified instruction-following task and use synthetic datasets to analyze a Transformer-based causal language model. Our findings suggest that the model learns task-specific information by clustering data within its hidden space, with this clustering process evolving dynamically during learning. We also demonstrate how this phenomenon assists the model in handling unseen instances, and validate our results in a more realistic setting. We further present applications in pre-training and alignment, inspired by clustering.

1 Introduction

In recent years, large language models (LLMs) have achieved remarkable capabilities in natural language processing and artificial intelligence more generally (Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023a). However, a significant challenge with LLMs is the misalignment between their training objectives and users' intentions. LLMs are trained to optimize next-word prediction on large-scale language data whereas users expect the model to follow their instructions in a helpful and safe manner (Zhang et al., 2023). Techniques such as reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022), direct preference optimization (DPO) (Rafailov et al., 2023), and instruction tuning (Zhang et al., 2023) have been proposed to further train LLMs for instruction following, yielding seemingly great instruction-following capabilities. Instruction-following also harkens back to controllable language models (Keskar et al., 2019).

Yet, the mechanisms underlying these successful instruction-following capabilities are not well-understood and require specific analysis. Since LLMs have grown exceedingly complex in terms of parameters and training data, such analysis is extremely challenging. For example, it is difficult to determine which token to focus on when analyzing a potentially lengthy and intricate textual sequence, so as to extract meaningful interpretation. To gain insights into the hidden mechanisms of LLMs, one approach is through carefully designed experiments. Conducting such experiments requires meticulous control over experimental settings and this is challenging due to the complexity of real-world language data, over which experimenters have limited control.

This limitation inspires us to devise a simplified instruction-following task with a synthetic dataset that we fully control but that reflects some key properties of natural language data. This approach mirrors practices in fields such as experimental psychology, where researchers aim to study the complexities of the human mind under simplified task conditions with controlled stimuli. Since the Transformer architecture (Vaswani et al., 2017) is commonly used to build LLMs, we aim to perform analysis on a Transformer-based causal language model (CLM) trained for a simplified instruction-following task to study its inductive biases.

More specifically, the ability to correctly recognize a learned task may be needed to successfully execute it. We aim to investigate how task-specific information is encoded into the representation space of Transformer-based CLMs trained for instruction-following. One intuitive hypothesis is that hidden states corresponding to the same task

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 5362–5387

are arranged close together to form a task-specific cluster, reminiscent of functional modules and topographic maps that neuroscientists have discovered in the brain (Knudsen et al., 1987; Chklovskii and Koulakov, 2004). The alternative is that the hidden states are scattered without forming clusters, and the Transformer learns mechanisms to identify tasks via these scattered hidden states. Section 3 provides experimental evidence supporting the former hypothesis.

This leads us to further questions. First, do these task-specific clusters emerge at a certain point during training or gradually evolve? Second, how might task-specific clustering enhance task performance? To dive into these, we must investigate model learning dynamics, but in-depth analysis of LLM training is expensive due to long training schedules and high computational costs. Our simplified setting, however, allows us to constrain the scope of the tasks so a relatively small model is able to fully learn the task with a short training schedule. Specifically, we perform clustering analysis on the hidden states of the Transformer model by training it for the instruction-following task and extend this analysis to the entire learning process. We train the model from scratch to isolate it from complicated and potentially noisy pre-training, leaving studies of pre-training impacts for future work.

Further, we validate findings from the simplified task in a more realistic setting and showcase two possible applications in pre-training and alignment that are inspired by the main findings herein. The effectiveness of these applications could be further validated in more realistic settings, but this is deferred to future work due to limited computational resources and to keep the scope of this work manageable.

In summary, we present a simplified instruction-following task and generate a synthetic dataset to examine a Transformer-based CLM model. Through this simplified framework, we offer evidence showing that the model learns task-specific information by organizing data into clusters within its hidden space. Moreover, we show that the clusters evolve dynamically as the model learns. Importantly, we illustrate how this clustering phenomenon aids the model in handling previously unseen instances. This is further validated in more realistic settings and applied in pre-training and alignment settings.

2 Instruction-following

2.1 Preliminaries

We assume a task is a function $f : \mathcal{X} \to \mathcal{Y}$ and each pair of its input and output (x, y) is a mapping. We define an instruction-following task as anticipating an output y by giving an instruction I and an input x. One example is "given a location, state its continent. New York City", where New York City is the input and the output should be North America. Essentially an instruction serves as a prompt that helps to identify a specific task function f. We assume an instruction I is sampled from an instruction distribution $\mathcal{I}$ and instructions sampled from different distributions may be associated with the same task reflecting the fact that a task can be expressed in very different ways. However, the opposite does not hold since this will lead to an ill-defined problem.

The input can either be integrated into the instruction or separated out. For simplicity, we consider separate inputs. One instance is represented as a sequence of a concatenation of an instruction, input, and output, $[I; x; y]$, where instruction, input, and output are represented as textual sequences. Then, the instruction-following task is formulated as a causal language modeling task by autoregressively predicting the next token in the sequence.

2.2 A simplified instruction-following task

For ease of analysis, we simplify the instruction-following task as follows. To emulate language data, we assume both alphabets $\mathcal{X}$ and $\mathcal{Y}$ are discrete. To have a focus of analysis, we assume the input and output are each represented by a single token. The next token prediction task allows us to have one token representation to evolve from the current token to the next token across the Transformer layers. We further assume the output token comes right after the input token without using any template tokens in between, allowing us to concentrate our study on representations of one single token across layers (i.e. its hidden states).

To accommodate these assumptions, we synthesize a task function by randomly sampling a finite number of mappings such that an input of the function is uniquely associated with an element in $\mathcal{Y}$. The mapping could be made stochastic, so an input could be associated with multiple different output elements, but for simplicity we mainly assume uniqueness (but see Appendix B). This reflects the fact that we usually only have a finite number of

demonstrations for a task, for model learning and for evaluation. Different task functions may share an identical input set, but the respective outputs could be different, so it is important for the model to learn to correctly identify a task to provide accurate output accordingly. Additionally, our focus lies on the model's ability to generalize by identifying the correct task from unseen instructions, rather than generalization of the task itself, which is beyond the scope of this study. Therefore, we provide all of the mappings from a task but only a portion of instructions to the model during the training process.

We aim to investigate the behavior of the model when provided with sufficient data to learn the patterns of different instructions. Given resource constraints, it is infeasible to create a large-scale instruction-following dataset of natural language that enables the model to fully comprehend the complexities inherent in natural language and then to train a Transformer model on such a vast dataset. Therefore, we opt to study the regularities of instructions simpler than those of natural languages using *regular expressions*. We provide an illustrative example in Figure 1. Regular expressions are patterns used to match character combinations in strings. They are widely used in text processing and search tasks, allowing for flexible and powerful matching operations. We can also use regular expressions to synthesize as much data as needed by sampling based on these expressions, so the model can adequately acquire the ability to recognize regularities within the instructions. In our study, we randomly sample a regular expression, as detailed in Appendix B. Each sampled regular expression is considered as a simple grammar rule. We then sample instructions represented as sequences of symbols based on the regular expression. We construct instructions by sampling instances based on different regular expressions to emulate different distributions.

Another concern arises from many real-world tasks needing to acquire external knowledge. For instance, to learn to predict the next letter in the alphabet sequence, the model must possess external knowledge of the alphabet itself. Since we synthesize task functions in our approach, as outlined earlier, we can present all information to the model to learn how to solve a task, overcoming this limitation. We associate each task with instructions originating from distinct distributions and construct a data instance by concatenating an instruction and a mapping together. Each instruction will accordingly have a task identity. In this context, the different distributions highlight instructions characterized by highly distinct regularities, such as varying vocabularies and syntactic structures. Given the existence of task-specific clusters as shown in Figure 2, this treatment also allows us to examine whether the model forms task-specific clusters based solely on the similarities of instructions. Moreover, to delve deeper into the Transformer model's ability to form task-specific clusters, we create hard examples by replacing a word within certain instructions in the training data with another word, thereby associating these instructions with a new task. For instance, in a realistic scenario, substituting the word "initial" with "secondary" from "return the initial letter from the provided letter list" indicates a different task. To further increase the difficulty, we introduce a new task with identical mappings as the original task but modify the outputs. In a realistic scenario, even a subtle change like this would likely trigger a different task. These hard examples can be viewed as outliers of the original data distribution. This creates instructions that are difficult to distinguish based solely on their appearance, posing a challenging task to assess whether the model can still effectively separate them into distinct clusters based on task identities.

3 Experiments

3.1 Implementation details

We construct a synthetic instruction-following instruction dataset based on the guidelines outlined in Section 2.2. This dataset is then divided into training and validation sets. For computational efficiency in subsequent clustering analysis, we randomly sample a number of instances from a subset of tasks to form the validation set and a training subset for intermediate evaluations. Given full control over the data generation process, we record meta information such as a task identity for each data instance. We construct 50 tasks in total and each task has 152 variants of instructions on average. Further details regarding the hyperparameters of the data generation process and statistics of the resulting datasets are in Appendix B.

We train a six-layer Transformer model following the GPT-2 architecture (Radford et al., 2019). This model is optimized using an AdamW optimizer (Loshchilov and Hutter, 2017) and employs

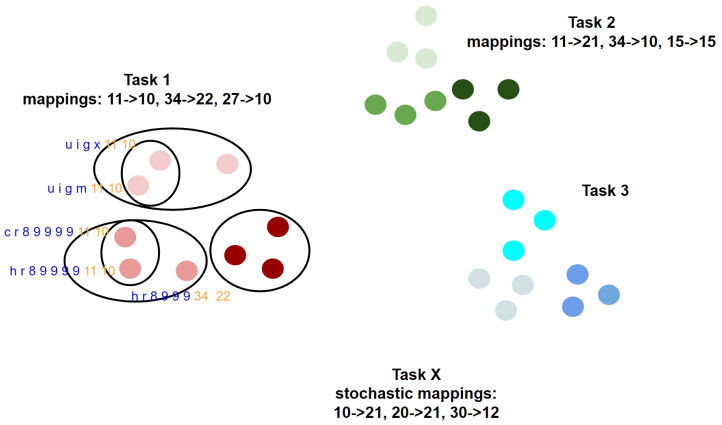


Figure 1: An illustrative example of our simplified instruction-following setting: Each task consists of three mappings, in which "->" means from an input to an output. Each dot represents a data instance, which is a concatenation of an instruction colored blue and a mapping colored orange. As illustrated, an output token comes right after an input token without any other tokens for formatting in between. Different dot colors indicate different instruction distributions, while the same color signifies instructions from the same distribution based on a regular expression. Similar colors represent instances of the same task. We illustrate three tasks, each with three instruction distributions, and Task X with stochastic mappings where outputs are shared by several inputs (stochastic mappings are considered in Appendix B.1 experiments). Tasks 1 and 2 overlap in input space, requiring the model to recognize the task from the instruction to predict the correct output. Clustering occurs such that instances group together in various ways, as shown by circles. Only a subset of instances is annotated for clarity.

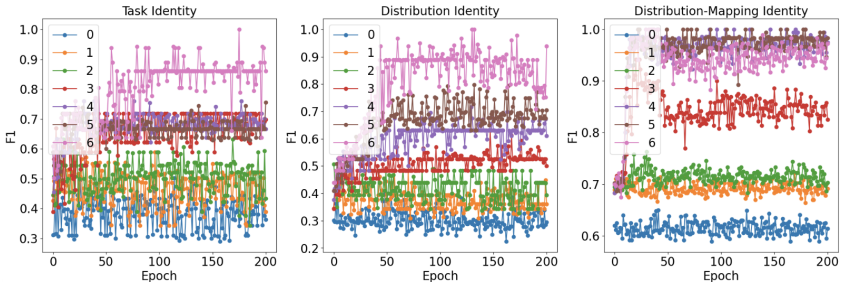
a cosine annealing learning rate schedule. We terminate training based on the best task accuracy achieved on the validation dataset. The task accuracy is measured by the percentage of correct outputs, which is treated as the measurement of task performance. Additional specifics about the hyperparameters of the model and training process are in Appendix D. We perform all of our experiments on a single NVIDIA A100 GPU with 80 GB memory.

3.2 Clustering analysis

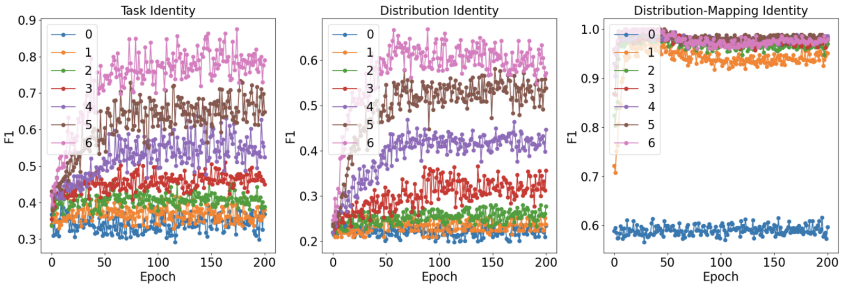
As detailed in Section 2.2, our study focuses on the hidden states corresponding to the input tokens (the last token in a sequence) within the Transformer model. We gather hidden states of the input tokens from various data instances. Next, we use the popular KMeans clustering algorithm to uncover clusters within the data. We optionally pre-process them using t-SNE dimension reduction (Van der Maaten and Hinton, 2008) if it benefits the subsequent clustering performance. We conduct extrinsic clustering evaluation on the clustering results, using task identities as labels. Our analysis reports results on the training subset and validation set, employing three commonly used metrics for clustering analysis: F1 score, adjusted Rand index (ARI), and adjusted mutual information (AMI). Further details on clustering analysis and evaluation metrics are in Appendix C.

As shown in Figure 2, on both of training and validation splits, there exists a strong trend of improvement of the clustering performance based on task identities throughout the training process until saturating at some high values. Figure 6 in the Appendix demonstrates results on all evaluation metrics. Especially during the late stages of the training process, the hidden states exhibit a strong clustering effect on task identities, indicated by high values across different data splits and metrics. The model early stops at the 51th epoch, but particularly noteworthy is the persistence and even improvement of the clustering phenomenon long after the early stopping point, indicating clustering as a strong inductive bias of the Transformer during its training process. In addition, the early stop point is close to when the clustering performance starts to saturate. After that, the model's task performance also saturates at high values as shown in Figure 3c, which may indicate a correlation between task performance and clustering performance. The correlation is further validated by results presented in Figure 15 in Appendix G.4: the correlation generally increases across layers, reaching a significant value.

Moreover, clustering performance tends to im-

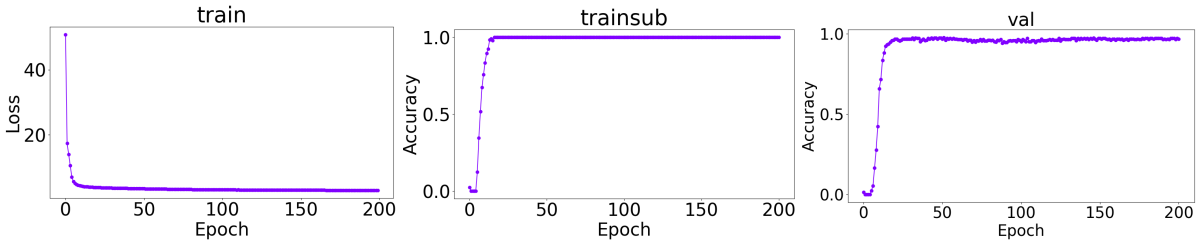


(a) Training subset



(b) Validation set

Figure 2: Clustering analysis on both of training subset (a) and validation set (b) across different layers throughout the training process: Different columns corresponds to uses of different identities as labels. Only shows results on F1 score here and see results on other evaluation metrics in Figure 6. Each dot represents a data point.



(a) Training Loss (b) Trainsub Task Accuracy (c) Validation Task Accuracy

Figure 3: (a) Training loss, (b) Training subset task accuracy, and (c) validation task accuracy throughout the training process. Each dot represents a data point. Both (b) and (c) show dense dots with near-zero accuracy for the first few epochs.

prove in higher layers of the Transformer model, with the 0th layer serving as a baseline solely based on input word embedding. Notably, the baseline does not undergo much change during the training process compared to the clustering performances of other layers. It is important to note that task identities are concealed from the training process, and the Transformer models perform clustering during training without explicit supervision. Moreover, we design the simplified task to have many tasks share the same inputs by using a small task-related vocabulary such that the model will not be able to identify a task solely from the inputs.

Interestingly, based on our formulation of the instruction-following task in Section 2.1, instances with instructions from different distributions are clustered together based on their task identities. This is corroborated by the high F1 score. Further, our analysis reveals that hard examples, formulated as described in Section 2.2, and their corresponding original examples are predominantly separated into different clusters. This can be seen from the significantly high F1 score at the late training stage on the training subset, which contains both the hard examples and their original examples. This eliminates the possibility that the model groups instances solely based on instruction similarities. This underscores the model's ability to form clusters based on task identities. Additionally, similar clustering phenomena are observed on both the validation and testing sets, indicating that the clustering effect generalizes to unseen instances as well. These results not only provide compelling evidence supporting the existence of task-specific clusters but also show

that the clusters evolve throughout the training process instead of appearing spontaneously.

We have found that task-specific clusters exist within the hidden representation space of the Transformer model. This suggests the question of whether there are any intriguing internal structures within these clusters. To explore this, we conduct the same clustering analysis using the distributions that the instructions belong to as labels. As demonstrated in Figure 2, we find obvious clustering effects based on this setting, which reveals possible inner clustering structures within the task-specific clusters. To delve deeper and uncover finer-grained structures within this hierarchical clustering, we conduct further analysis by considering a combination of labels, including the identities of instruction distributions and mappings. We observe strong clustering under this setting as well, as evidenced by high clustering performances. See (Knudsen et al., 1987; Chklovskii and Koulakov, 2004) for similar clustering and mapping phenomena in the mammalian cortex. In addition, by comparing to the performance on the training subset, the model performs generally better on the validation set when using the combination labels. This is because the training subset contains some more challenging hard examples, which are used to check if the model clusters examples solely based on their similarities.

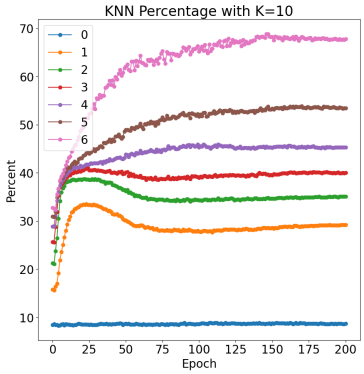
Figure 3 showcases the learning curve of the model based on task accuracy. An intriguing observation is that the task accuracy remains around zero for the first few epochs on both training subsets and validation set before abruptly beginning to rise thereafter, despite continuous improvements in training loss as depicted in Figure 3a. We hypothesize that the model initially learns task identification through the evolved clustering process by resolving various ambiguities that we introduced, including the hard examples, enabling it to subsequently learn to solve different tasks successfully. We also confirm a similar clustering phenomenon and various trends we have discovered so far on a smaller model with a small hidden dimension of 32 and a larger model with a hidden dimension of 2048. See results on these additional models in Figures 7, 8, 9, and 10 of the Appendix. We also repeat the experiments for three different random seeds and present the results in Appendix G.2, from which consistent results are observed.

In addition, we conduct experiments in a new setting where tasks share similarities and stochastic mapping is allowed. We define two tasks as similar if they share at least one mapping. Detailed descriptions of the experiments and results are provided in Appendix B.1 and Figure 14. We observed that instances of the same tasks form clusters, and instances of similar tasks also cluster together. This indicates that the model places representations of similar tasks relying on some common knowledge (shared mappings) in close proximity to each other.

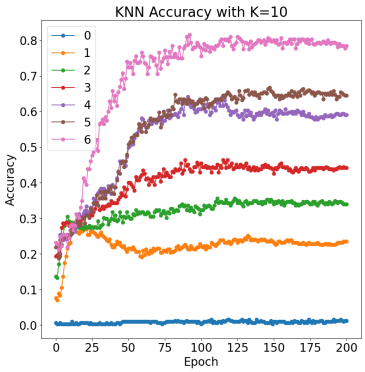
3.3 Advantages of clustering

Next, we explore the potential advantages of the clustering phenomenon observed earlier. We have observed clustering effects on both the training subset and unseen instances from the validation set. To further verify if training instances and unseen instances with the same task identity are close in the hidden representation space, Figure 4a shows the percentage of K nearest training instances of an unseen instance belonging to the same task identity, averaged over all instances. The nearest neighbors are found based on the original representations without the pre-processing by t-SNE. We observe a dramatic improvement in the percentage along the training process, indicating that both training instances and unseen instances are not only close in the hidden space but also become more clustered as training proceeds. This suggests the same task-specific clustering structure generalizes to the unseen instances.

Previous work (Khandelwal et al., 2019) has demonstrated that using an inference method based on K-nearest neighbors (KNN) with pre-trained Transformer-based language models can achieve competitive or even better next-token prediction performance than inference methods based on models' forward pass. This inspires us to measure task performance of our models based on KNN during the training process. From Figure 4b, we observe that task accuracy based on KNN improves consistently during training until saturating at high values, providing direct evidence of the advantages of task-specific clustering by bringing instances of the same task closer together in the hidden space. More specifically, the model clusters instances belonging to the same task close to each other so it is easy for making inferences for even unseen instances by using their nearby data instances. The KNN accuracy is also improved across layers. This is also apparently a working way to identify a specific task by a model gradually moving a representation to those with the same task over a series of layers.



(a) KNN Percentage



(b) KNN Accuracy

Figure 4: (a) Percentage of K nearest neighbors in the training set of an unseen instance belonging to the same task identity. (b) K nearest neighbors accuracy. Measurements are performed across all layers and throughout the training process.

3.4 Analysis of natural instruction-following task

One further question to ask is if the clustering phenomenon we discovered under the simplified setting holds in realistic settings. Therefore, we study trained LLMs on a realistic instruction-following task based on natural language. We use 16 tasks and their descriptions in natural language from Hendel et al. (2023) of three categories: Knowledge, Linguistic, and Translation (released under CC-BY 4.0 license). In accordance with the typical approach of constructing instruction datasets via LLM self-instruct (Wang et al., 2022), we build a set of instructions for each task by using their task descriptions as seeds to prompt ChatGPT (OpenAI, 2024) to generate 50 different expressions. We consider expressions sampled based on the same seed as coming from the same distribution. The specifics of the task descriptions and prompts used for querying ChatGPT are in Appendix E. The subsequent step involves linking each instruction to a task mapping provided by (Hendel et al., 2023) in the same way as described in Section 2.2. We only keep those task mappings that have inputs and outputs of only a single word to have a better focus of study. For computational efficiency, we only use ten of those selected task mappings for each task. We assign the same task identity to tasks under the same category due to their similarities. The resulting dataset contains 8,800 data points. Actually, the instruction-following data can be considered as a special kind of language data that naturally exists in the large-scale language data used for pre-training. Therefore, we will perform the same clustering analysis as in Section 3.2 on several different open LLMs either instruction tuned or not: LLaMa-7B, LLaMa-13B (Touvron et al., 2023a; Geng and Liu, 2023; Computer, 2023), GPT-J-6B (Wang and Komatsuzaki, 2021) (all three released under a Apache 2.0 license), LLaMa-2-7B-Instruct (Touvron et al., 2023b; Together, 2023) (released under Llama 2 License) and Instruct-GPT-J-6B (NLP Cloud, 2023; Taori et al., 2023) (released under GPL-3.0 license), in which LLaMa-2-7B-Instruct and Instruct-GPT-J-6B are fine-tuned on instruction datasets. Here, B (billion) refers to the number of parameters; model sizes are detailed in Table 9 in the Appendix.

We only report measurements of the layer with the best F1 score for clustering based on task identity. As in Table 1, similar to our results in the simplified setting, all of the LLMs achieve high clustering performances based on task identities. In particular, a high F1 score indicates different tasks under the same category are clustered together. Both the LLaMa models of different sizes and the LLaMa-2-7B-Instruct models receive high scores on clustering instances with the same distribution-mapping identities, which is consistent with our results on the simplified setting. However, the GPT-J-6B and Instruct-GPT-J-6B seem to not form clear clusters based on the distribution-mapping identity. Moreover, as we expected, the clustering phenomenon appears on LLMs whether instruction-tuned or not. Note that the conclusions we made on the simplified setting may not completely extend to the realistic settings due to differences in data complexity and scales. However, we do still see some consistent results, which indicates some generalizability of the clustering phenomenon. We hope our analysis of both the simplified and realistic set-

Model	Task			Distribution			Distribution-Mapping		
	F1	ARI	AMI	F1	ARI	AMI	F1	ARI	AMI
LLaMa-7B	0.959	0.872	0.869	0.571	0.438	0.673	0.940	0.903	0.974
LLaMa-13B	0.953	0.855	0.854	0.546	0.381	0.645	0.936	0.893	0.969
GPT-J-6B	0.887	0.697	0.665	0.652	0.485	0.653	0.465	0.345	0.602
LLaMa-2-7B-Instruct	0.917	0.756	0.780	0.563	0.339	0.638	0.932	0.890	0.966
Instruct-GPT-J-6B	0.907	0.749	0.692	0.471	0.290	0.516	0.170	0.084	0.342

Table 1: Clustering analysis on open LLMs using task identity, distribution identity, and distribution-mapping identity as labels.

tings can shed some light on inductive biases of the Transformer-based LLMs for instruction following.

4 Applications

Now we demonstrate two applications under our simplified setting inspired by our findings: pre-training a model using task identities in the following subsection and an alignment algorithm with less forgetting (based on switching away from toxic clusters) in Appendix F.1.

4.1 Pre-training with task identities

Our experimental results from Figure 2 present task-specific clusters evolved from the training process for the instruction-following task. This inspires us to ask if we can directly guide a model to learn the clustering structures within its hidden representation space before being tuned for various tasks, to make the tuning process more efficient.

To verify this idea, we pre-train a CLM model to predict a task identity (ID) for a given instruction. More specifically, we append a special token at the end of an instruction to trigger the model to predict the corresponding task ID as the next token. We select the model with the best prediction performance on the validation set for fine-tuning at the next stage. Note that the model is pre-trained by a regular causal language modeling task over an entire sequence instead of solely the task ID prediction task. Also, the task mappings are not used at the pre-training stage. As shown in Figure 5a, the task-specific clusters indeed evolve during the pre-training process. After pre-training, we fine-tune the model for the instruction-following task and measure the task accuracy over the fine-tuning process. We compare this pre-training strategy with direct training for the instruction-following task denoted by *No Pre-training*.

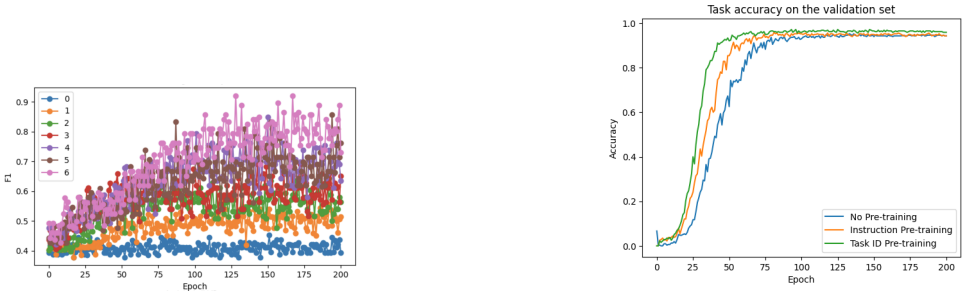
To see if our pre-training approach makes additional contributions besides language modeling over the instructions, we also compare it to a setting (Instruction Pre-training) that pre-trains a model by causal language modeling over only the instructions. A model for the following fine-tuning is chosen based on the minimum validation CLM loss. We focus on a smaller model with a hidden dimension of 256 since there are two stages of training.

From Figure 5b, we see that our method not only converges faster but also achieves higher task accuracy on the validation set during the fine-tuning process than the other two methods. Comparing to Instruction Pre-training indicates the information about task ID benefits the downstream fine-tuning. The results demonstrate both the efficiency and effectiveness of our pre-training method. It may be feasible to perform this kind of pre-training scalably in practice since information about task identities is not hard to collect and is more sparse compared to entire instruction-following data. Further, due to the high cost of LLM fine-tuning, shortening its training schedule by accelerating its convergence is both economically and environmentally appealing.

More experimental details are in Appendix D. The pre-training application here and the alignment application in Appendix F.1 are currently limited to simplified settings. Their effectiveness could be further verified in more realistic settings, but this is left as future work due to limited computational resources and to constrain the scope of this work. We aim to concentrate on model analysis in this work and use these applications to demonstrate the potential usefulness of our findings.

5 Related works

Instruction following: Making LLMs follow user intention specified in instructions is important for making them more truthful and less toxic. Many efforts has been made to achieve this goal (Hill et al., 2020; Zhang et al., 2023; Ouyang et al.,



(a) (b)

Figure 5: (a) Clustering analysis of the model trained for the task ID prediction following the setting in Section 3.2. (b) Comparison of different pre-training strategies by their performances during the fine-tuning process. The task accuracy is measured on the validation set.

2022; Rafailov et al., 2023). In our work, we focus on studying the hidden mechanism and inductive biases of the Transformer-based CLMs to learn instruction following in a general and simplified setting, rather than developing a more advanced instruction-following method.

Functional vectors: Recent works (Todd et al., 2023; Hendel et al., 2023) present compelling evidence of function vectors that store task-related information in in-context learning. While in-context examples can be viewed as a specific type of instruction, our work primarily focuses on conducting analysis based on more general textual instructions rather than in-context examples. Further, our study extends to analyzing the learning dynamics of models, rather than solely focusing on trained models.

Mechanistic interpretability: The primary objective of mechanistic interpretability is to reverse engineer model behaviors (Olah et al., 2020; Elhage et al., 2021; Nanda et al., 2023; Meng et al., 2022; Hernandez et al., 2023; Geva et al., 2023; Conneau et al., 2018; Ilharco et al., 2022). Similar to many of the works in this area, we conduct studies based on a synthetic task and data to gain better controllability of the experiments and perform more in-depth analysis.

Clustering in Transformers: Some studies also explore clustering phenomena within the Transformer model (Chen et al., 2021; Reif et al., 2019; Geshkovski et al., 2023; Thompson and Mimno, 2020). However, they did not specifically focus on the instruction-following setting and conducted analysis mainly on trained models. Geshkovski et al. (2023) primarily concentrates on studying clustering among tokens within a sequence. In contrast, our clustering analysis focuses on identifying clustering structures among different sequences.

6 Conclusion and discussion

In this work, we introduce a simplified instruction-following task and construct synthetic datasets to analyze a Transformer-based CLM model. From the simplified setting, we provide experimental evidence supporting the notion that the model encodes task-specific information through clustering in its hidden space, and demonstrate that this clustering evolves continuously during the learning process. Additionally, we highlight the advantages of the clustering phenomenon for the model to handle unseen instances. We also further verify the existence of the clustering phenomenon in realistic settings. The inductive biases uncovered and analyzed in this study offer new insights into Transformer-based CLM models and shed light on their remarkable instruction-following capabilities. Further, this newfound understanding can inspire the development of more advanced algorithms to enhance LLM's capability to effectively follow human instructions. We have shown two inspired applications—pre-training and alignment—and are continuing to explore more applications.

Separately, Huh et al. (2024) also studied representations of various models and discovered representational alignments not only among different models of the same modality but also among models of different modalities. Our findings could provide a potential explanation: representations of different models converge to similar clustering structures. We plan to explore these aspects in future work.

7 Limitations

At present, our main study is confined to a simplified task and synthetic dataset along with specific data distributional assumptions. Expanding the analysis to encompass a broader range of diverse and realistic distributional assumptions on data is an avenue for future exploration. We anticipate that our study can provide insights into Transformer's hidden mechanisms and inductive biases, serving as a foundational starting point and offering directions for analysis on larger scales. It is conceivable that our findings may have broader applicability and could be validated across a wider array of scenarios beyond our simplified instruction-following tasks. Scaling up our analysis to encompass more complex and realistic scenarios and verifying the inspired applications in a more realistic setting are areas we plan to explore in future research endeavors.

Acknowledgments

We are grateful to Sourya Basu and Austin Ellis-Mohr for their feedback. Besides, we used an LLM-based AI assistant (ChatGPT) to help in polishing and proofreading the paper.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Boli Chen, Yao Fu, Guangwei Xu, Pengjun Xie, Chuanqi Tan, Mosha Chen, and Liping Jing. 2021. Probing bert in hyperbolic spaces. *arXiv preprint arXiv:2104.03869*.

Dmitri B. Chklovskii and Alexei A. Koulakov. 2004. Maps in the brain: What can we learn from them? *Annual Review of Neuroscience*, 27:369–392.

Together Computer. 2023. Redpajama-data: An open source recipe to reproduce llama training dataset.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1.

David Freedman, Robert Pisani, and Roger Purves. 2007. Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*.

Xinyang Geng and Hao Liu. 2023. Openllama: An open reproduction of llama.

Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. 2023. A mathematical perspective on transformers. *arXiv preprint arXiv:2312.10794*.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. *arXiv preprint arXiv:2310.15916*.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2023. Linearity of relation decoding in transformer language models. *arXiv preprint arXiv:2308.09124*.

Felix Hill, Sona Mokra, Nathaniel Wong, and Tim Harley. 2020. Human instruction-following with deep reinforcement learning via transfer-learning from text. *arXiv preprint arXiv:2005.09382*.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. 2024. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

Eric I. Knudsen, Sascha du Lac, and Steven D. Esterly. 1987. Computational maps in the brain. *Annual Review of Neuroscience*, 10:41–65.

Stuart Lloyd. 1982. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Neel Nanda, Andrew Lee, and Martin Wattenberg. 2023. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*.

NLP Cloud. 2023. instruct-gpt-j-fp16. https://huggingface.co/nlpcloud/instruct-gpt-j-fp16.

Colm O'Connor. 2015. Xeger).

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001.

OpenAI. 2023. GPT-4 technical report. *arXiv:2304.01852*.

OpenAI. 2024. Chatgpt (february version).

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of bert. *Advances in Neural Information Processing Systems*, 32.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Laure Thompson and David Mimno. 2020. Topic modeling with contextualized word representation clusters. *arXiv preprint arXiv:2010.12626*.

Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. 2023. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*.

Together. 2023. Llama-2 models with together api. https://www.together.ai/blog/llama-2-7b-32k-instruct.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. LLaMA: Open and efficient foundation language models. *arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

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

Ben Wang and Aran Komatsuzaki. 2021. Gpt-j-6b: A 6 billion parameter autoregressive language model.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Zhenyi Wang, Enneng Yang, Li Shen, and Heng Huang. 2023. A comprehensive survey of forgetting in deep learning beyond continual learning. *arXiv preprint arXiv:2307.09218*.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

A Societal impact

This work enhances the understanding of inductive biases and hidden mechanisms of Transformer-based CLMs. By better understanding how the language models work, we can build systems that are more transparent, fostering greater trust among users. Insights into the models' mechanisms allow for the identification and reduction of biases, leading to fairer AI systems. In addition, the task identity pre-training algorithm we proposed in Section 4.1 could potentially reduce the development costs of language models and make the process more environmentally friendly. The novel alignment algorithm proposed by this work in Section F.1 could potentially be leveraged to reduce the biases and the toxicity of language models, leading to a positive societal impact. However, deeper insights into the biases and mechanisms of language models could be misused to develop harmful AI systems. Nonetheless, the understanding provided by this work could also lead to the design of better monitoring mechanisms to identify and rectify such misuse.

B Simplified setting

To construct data for the synthetic instruction-following task under the simplified setting, we first sample a task function consisting of several unique mappings. For each mapping, we sample two symbols from a task symbol vocabulary to form a mapping. We enforce that no two functions share a mapping. Next, we sample instructions for each task based on a regular expression by using Xeger package (O'Connor, 2015) (released under MIT license). Regular expressions are sequences of characters that define a search pattern. They are widely used in computing for tasks such as text processing, string manipulation, and pattern matching. Regular expressions consist of normal characters (like letters and digits) and special characters (also known as metacharacters) that have special meanings. These metacharacters allow you to specify rules and conditions for matching patterns within text. We use regular expression reversely by sampling a string from a search pattern. To sample a regular expression, we first sample several metacharacters and then sample normal characters from an instruction vocabulary as their arguments and concatenate them together as a regular expression. For computational efficiency, we build a training subset, validation set, and hard examples from a subset of tasks by randomly selecting several instructions sampled from all of the distributions associated with each of the tasks. Please refer to Table 3 for related hyperparameters. We emulate sampling from different distributions by sampling from different regular expressions as described in Section 2.2.

We present the statistics of the synthetic instruction-following dataset in Table 2.

B.1 Similar task setting

Additionally, we conduct experiments in a new setting where tasks exhibit similarities and stochastic mapping is permitted. We define two tasks as similar if they share at least one mapping. We use the same hyperparameters for data construction as in the main experiment. Specifically, we designed the data so that similar tasks share three mappings. We linked different instruction distributions of the same task identities to different but similar tasks. The same model architecture and training strategy from the main experiment were employed. The results are presented in Figure 14.

C More on clustering analysis

In this work, we use extrinsic evaluation for our clustering analysis. Extrinsic evaluation of clustering refers to assessing the quality of clustering results by comparing them to ground truth. Ground truth data refers to labeled data that indicates the class or cluster to which each data point belongs. We utilize three widely used evaluation metrics: F1, adjusted Rand index, and adjusted mutual information.

F1 Score: The F1 score combines both precision and recall into a single value, making it a useful measure of a model's accuracy. The formula for the F1 score is $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$. The F1 score ranges from 0 to 1 with higher values indicating better agreement to the ground truth.

Adjusted Rand Index (ARI): ARI is a measure of the similarity between two clustering results. It considers all pairs of samples and counts pairs that are assigned to the same or different clusters in both the true and predicted clusterings. ARI ranges from -1 to 1, where 1 indicates perfect clustering agreement, 0 indicates clustering results are random, and negative values indicate less agreement than expected by chance.

Adjusted Mutual Information (AMI): AMI is another measure used to evaluate the quality of

Setting	Set	Size
Simplified	Training	7,300
	Training subset	180
	Validation	315
Realistic	Testing	8,800

Table 2: Data statistics of both simplified and realistic settings. The size of a data set is quantified by its number of instances. Only testing set is available for the realistic setting since we use pre-trained models instead of we training and validating a model in this setting.

clustering. It quantifies the amount of information obtained about one clustering from knowing the other, adjusting for chance. Like ARI, AMI ranges from -1 to 1, where higher values indicate better agreement between clusterings.

D Hyperparameters

Tables 3 and 8 show the hyperparameters of the data generation processes. Table 4 contains the hyperparameters of our Transformer model and its training process. Hyperparameters for t-SNE, KMeans and KNN are listed in Tables 5, 6 and 7.

E Natural instruction-following task

E.1 ChatGPT prompt template

We use the following prompt template to query ChatGPT to generate different expressions of a task description: "Rewrite 50 different expressions of XXX", where "XXX" is a task description.

E.2 Realistic setting

See Table 10 for the task descriptions used for constructing the dataset for the realistic setting as detailed in Section 3.4 and data statistics in Table 2.

F More applications

F.1 Alignment

One goal in alignment is to reduce undesirable behaviors of LLMs such as toxicity via additional tuning. Aligning a model to reduce toxicity can be simulated in our simplified setting: We assume there is a toxic behavior represented as a task triggered by some instructions with toxic tendencies. We want to convert the toxic source task to a healthier target task that is associated with instructions of goodwill and shares the same input domain. Then, a model gives outputs corresponding to the healthier task, even if toxic instructions are seen. We consider the model to get exposure to the healthier task during its training because this is close to the reality, in which the model is trained on large-scale language data.

We first train a model on an instruction-following dataset generated according to our simplified setting, in which we label some tasks as toxic, some as their healthier counterparts, and others as regular tasks. Next, we construct a new and smaller dataset consisting of only the data relevant to the toxic tasks by updating the outputs of the toxic tasks according to the target tasks and keeping the toxic instructions unchanged. We fine-tune the model using the new dataset to perform the alignment.

A straightforward idea is direct fine-tuning, but this may lead to unwanted catastrophic forgetting of other tasks (Wang et al., 2023). To overcome this drawback, one can fine-tune only the linear language head by freezing the rest of the model. This approach could be useful in practice since we are only expected to make limited changes to the model's behavior as simple as updating outputs for certain tasks. However, updating the language head may still impact the performance of other tasks because all tasks share the same language head. Since clusters found by Kmeans may have good separability, we are inspired to use a switch network that activates a new language head for the source tasks and the original language head otherwise based on the hidden states of the last tokens from the last layer. Then, only the new language head and the switch network are trained during the fine-tuning process such that the performance of other tasks could largely remain. Since the clustering structure is clear according to our experiments in Section 3.2, we use instances of the source tasks and only one instance for each other task to train the switch network to classify whether an instance belongs to a source task or not.

We consider both a linear switch network implemented by a linear layer and a switch network con-

Hyperparameter	Value
Number of tasks	50
Maximum number of instruction distributions per task	6
Minimum number of instruction distributions per task	1
Number of instructions per distribution	10
Number of mappings per task	5
Number of tasks in training subset	5
Number of instructions per distribution in the training subset	all available
Number of tasks in the validation set	10
Number of instructions per distribution in the validation set	3
Number of different tasks in hard examples	5
Number of instructions per distribution in hard examples	3
Size of the task symbol vocabulary	25
Size of the instruction symbol vocabulary	35
Maximum number of metacharacters per regular expression	3
Minimum number of metacharacters per regular expression	1
Maximum number of characters per metacharacters	10
Minimum number of characters per metacharacters	3

Table 3: Hyperparameters used for the data generation process.

Hyperparameter	Value
Learning rate	1E-4
Number of epochs	200
Optimizer	AdamW
Max gradient normM	1.0
validation criterion	Task accuracy
Scheduler	Cosine Annealing
Number of layers	6
Number of heads	8
Hidden dimension	768
Feedforward network dimension	1024
Dropout	0.2

Table 4: Hyperparameters related to our model in the main experiment and its training.

Hyperparameter	Value
Number of Components	3
Perplexity	10
Number of iterations	2,000
Metric	Euclidean
Initialization method	PCA

Table 5: Hyperparameters for t-SNE (Van der Maaten and Hinton, 2008).

Hyperparameter	Value
Number of Clusters	ground truth number
Initialization method	k-means++
Maximum number of iterations	300
Algorithm	Lloyd (Lloyd, 1982)

Table 6: Hyperparameters for KMeans.

Hyperparameter	Value
Number of neighbors	10
Weight function	uniform
Metric	Minkowski

Table 7: Hyperparameters for KNN.

Hyperparameter	Value
Number of tasks	50
Number of source-target pairs	3
Maximum number of instruction distributions per task	6
Minimum number of instruction distributions per task	1
Number of mappings per task	5
Number of tasks in the validation set	10
Number of instructions per distribution in the validation set	3
Size of the task symbol vocabulary	65
Size of the instruction symbol vocabulary	35
Maximum number of metacharacters per regular expression	3
Minimum number of metacharacters per regular expression	1
Maximum number of characters per metacharacters	10
Minimum number of characters per metacharacters	3

Table 8: Hyperparameters of the data generation process for the alignment experiments in Section F.1. We do not consider hard examples in this setting.

Model	Hidden Dimension	Parameter Count
Our model 768	768	23 million
Our model 32	32	55 thousand
Our model 256	256	3 million
Our model 2048	2048	202 million
LLaMa-7B	4096	7 billion
LLaMa-13B	5120	13 billion
LLaMa-2-7B-instruct	4096	7 billion
GPT-J-6B	4096	6 billion
Instruct-GPT-J-6B	4096	6 billion

Table 9: Sizes of models used in this work in terms of parameter counts and size of hidden dimension. The names of our models trained in the simplified setting end with their hidden dimension sizes.

Category	Task	Description
Translation	French to English	Given a word in French, translate to English
	English to French	Given a word in English, translate to French
	Spanish to English	Given a word in Spanish, translate to English
	English to Spanish	Given a word in English, translate to Spanish
	Italian to English	Given a word in Italian, translate to English
	English to Italian	Given a word in English, translate to Italian
Linguistic	Antonyms	Given an English adjective, output an antonym
	plural to Singular	Given an English noun in plural form, output the singular form
	Singular to plural	Given an English noun in singular form, output the plural form
	Present to gerund	Given an English verb in present simple tense, output the corresponding gerund form
	Present to past perfect	Given an English verb in present simple tense, output the corresponding verb in past perfect
	Present to past simple	Given an English verb in present simple tense, output the corresponding verb in past simple
Knowledge	Country to Capital	Given a name of a country, output the name of the capital city
	Location to continent	Given a name of a location, output the name of its continent
	Religion	Given a name of a location or a person, output the associated religion
	Person to Language	Given a name of a person, output their native language

Table 10: Task descriptions provided by Hendel et al. (2023)

structed by a more capable three-layer multilayer perceptron (MLP). The intermediate dimension of the MLP is the same as the hidden dimension of the CLM model. We compare different methods by selecting a model based on the best average validation accuracy of updated source tasks.

Table 11 shows that all methods achieve perfect performance for the updated source tasks on a validation set, but the direct fine-tuning undergoes significant performance drops on other tasks. Note that perfect performance means a model provides updated and healthier outputs even if toxic instructions are given. The method that only fine-tunes the language head retains more performance on other tasks, but not as much as methods using the linear switch network and the MLP-based switch network. The MLP-based switch network is slightly better than the linear switch network. Moreover, a few additional examples of the other tasks are not only attainable in practice but also are sufficient to achieve a small performance degradation, which showcases good scalability of the method and further supports good separability. The performance of other tasks is not completely preserved due to the imperfection of the switch network and more examples may be used to obtain a better switch network.

G More results

We present additional experimental results in this section.

G.1 Various models

We show results obtained on models with various architectures in Figures 6-10.

G.2 Various random seeds

We also repeat our main experiment with three different random seeds and report the means and standard deviations in Figures 11, 12 and 13. The greater variations in the results based on training subsets in Figure 11a are due to instability from learning the hard examples.

G.3 Experiments on similar tasks.

In Figure 14, we present results based on the similar task setting described in Appendix B.1.

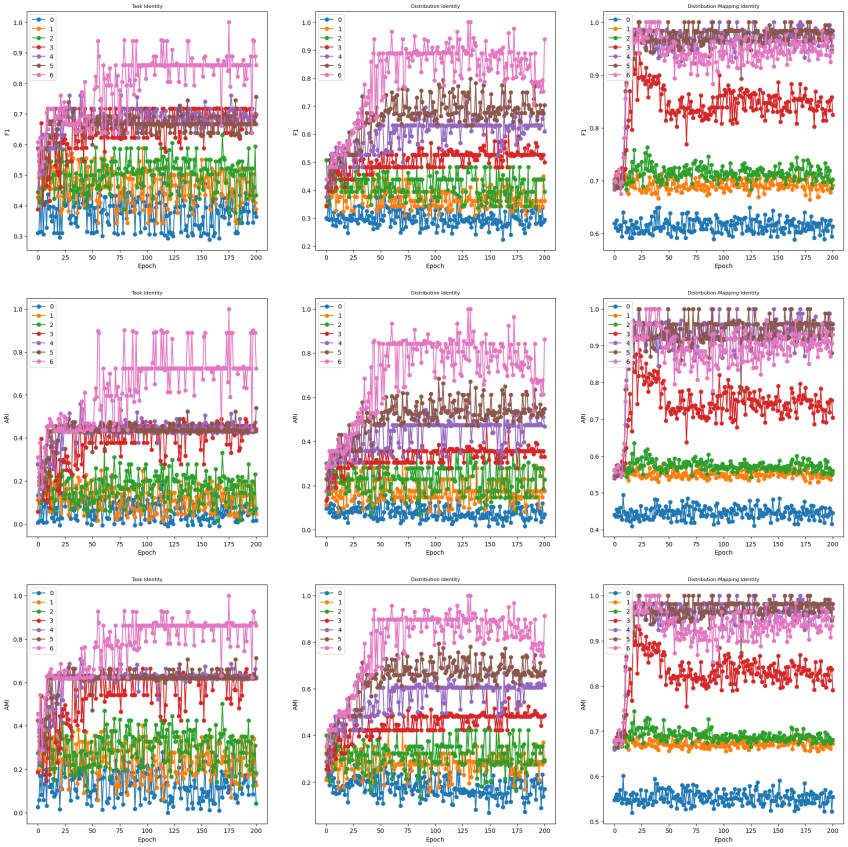
G.4 Correlation analysis

We conducted a correlation analysis between task performance and clustering performance. In Figure 15, Pearson correlation coefficients (Freedman et al., 2007) are calculated between task performance and clustering performance throughout the entire training process for each individual layer.

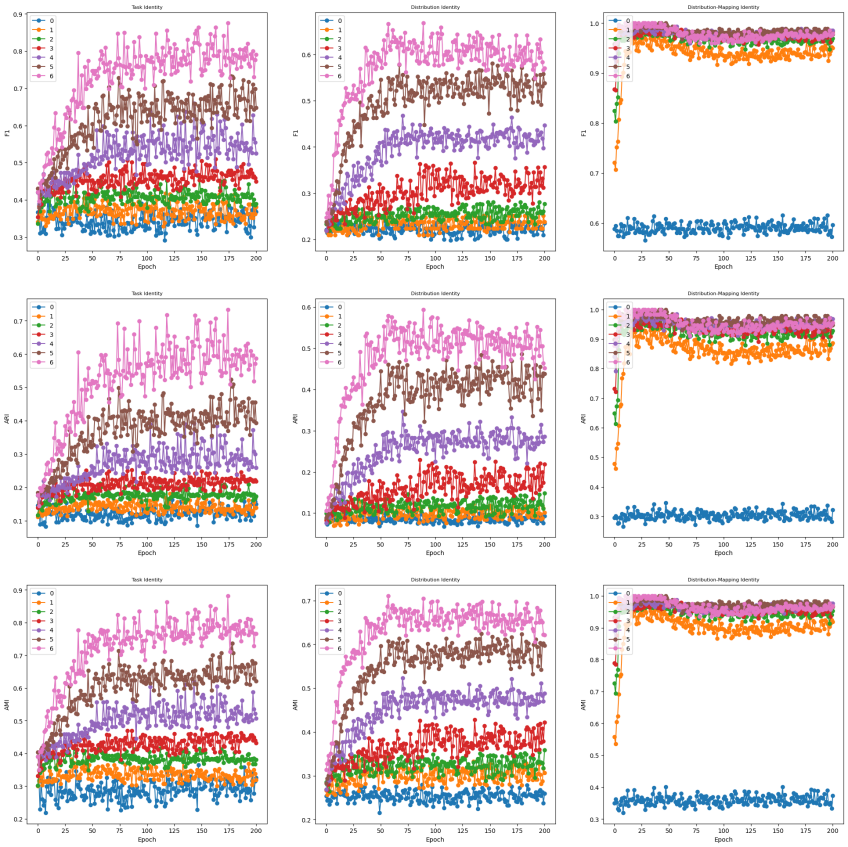
Method	Task Accuracy	Accuracy Drop
Direct fine-tuning	**1.00**	0.210
Language head fine-tuning	**1.00**	0.120
Ours with Linear switch network	**1.00**	0.087
Ours with MLP-based switch network	**1.00**	**0.080**

Table 11: Comparison of different alignment methods described in Section F.1 by their performances on a validation set. Task accuracy is measured for updated source tasks only and accuracy drop is measured for tasks other than the source tasks.

We observe that for all combinations, the correlation generally increases across layers, reaching values over 0.6 for accuracy and below -0.6 for loss, indicating a relatively strong correlation.

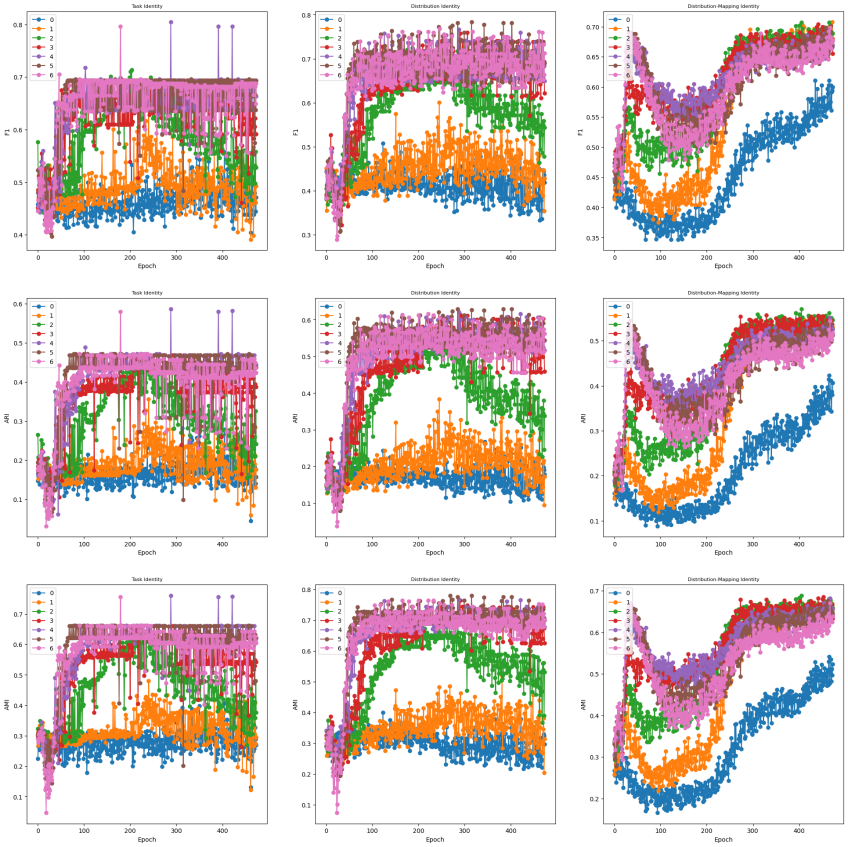


(a) Training subset

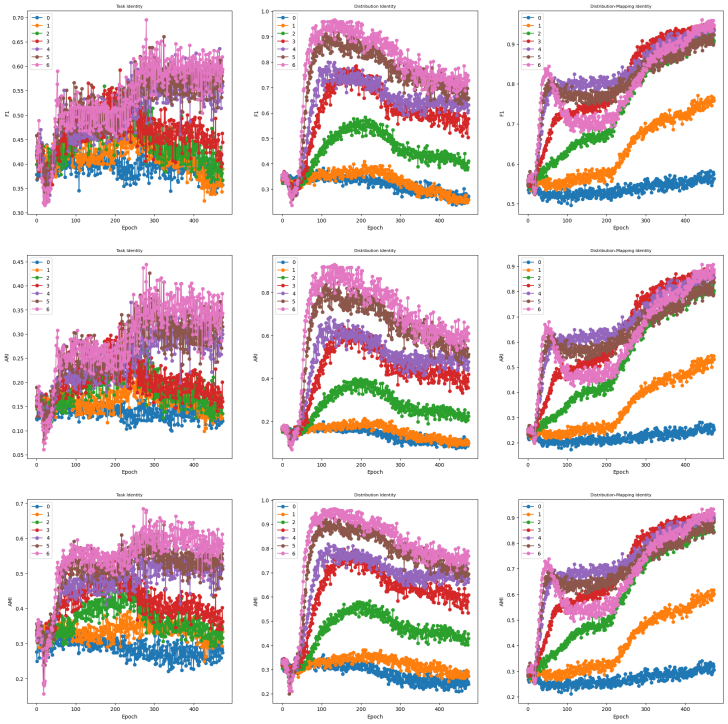


(b) Validation set

Figure 6: Clustering analysis on both training subset (a) and validation set (b) across different layers throughout the training process: Different columns correspond to uses of different identities as labels.

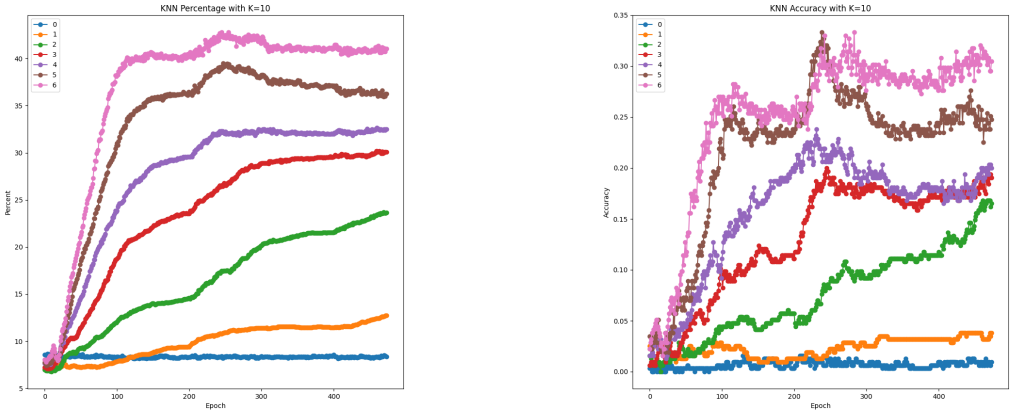


(a) Training subset



(b) Validation set

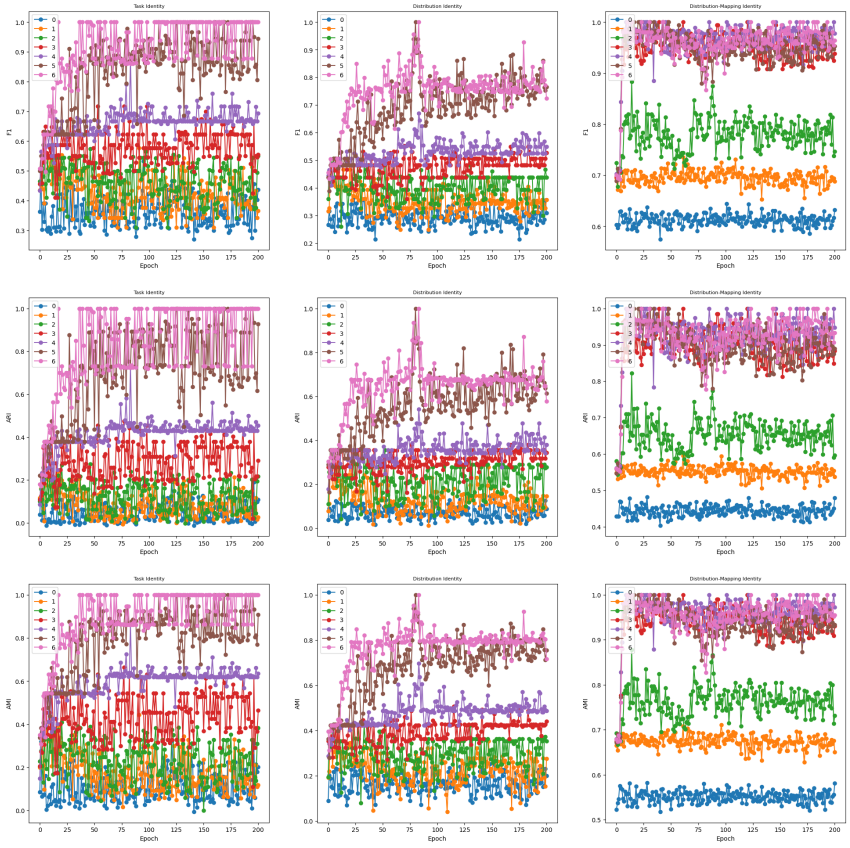
Figure 7: Clustering analysis on both training subset (a) and validation set (b) across different layers throughout the training process: The results are shown for the model with 32 hidden dimensions. We train this model for 500 epochs due to its slow convergence. Different columns correspond to the uses of different identities as labels.



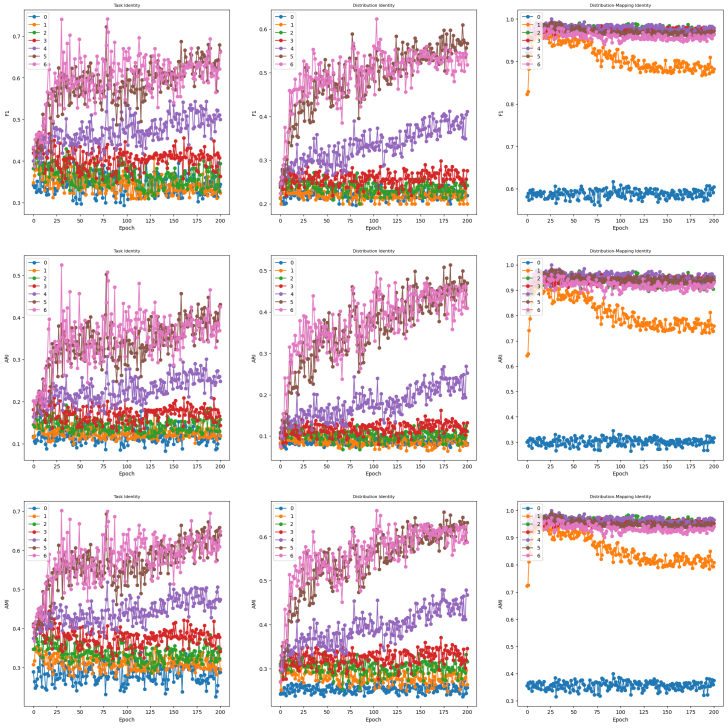
(a) KNN Percentage

(b) KNN Accuracy

Figure 8: (a) Percentage of K nearest neighbors in the training set of an unseen instance belonging to the same task identity. (b) K nearest neighbors accuracy. Measurements are performed across all of layers and throughout the training process. The results are shown for the model with 32 hidden dimension.

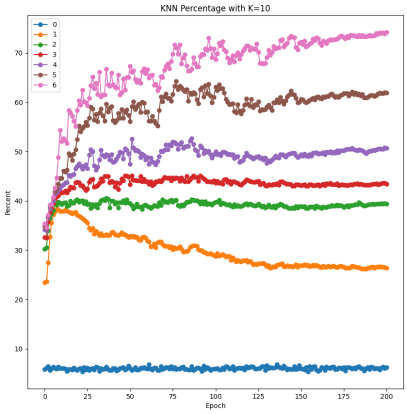


(a) Training subset

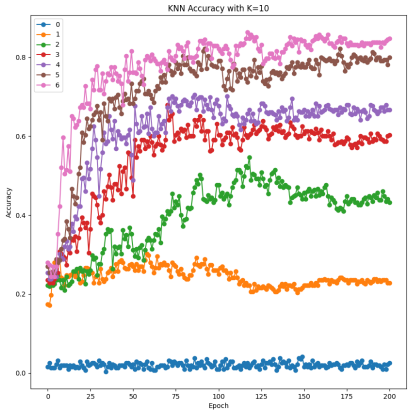


(b) Validation set

Figure 9: Clustering analysis on both training subset (a) and validation set (b) across different layers throughout the training process: The results are shown for the model with 2048 hidden dimension. Different columns correspond to the uses of different identities as labels.

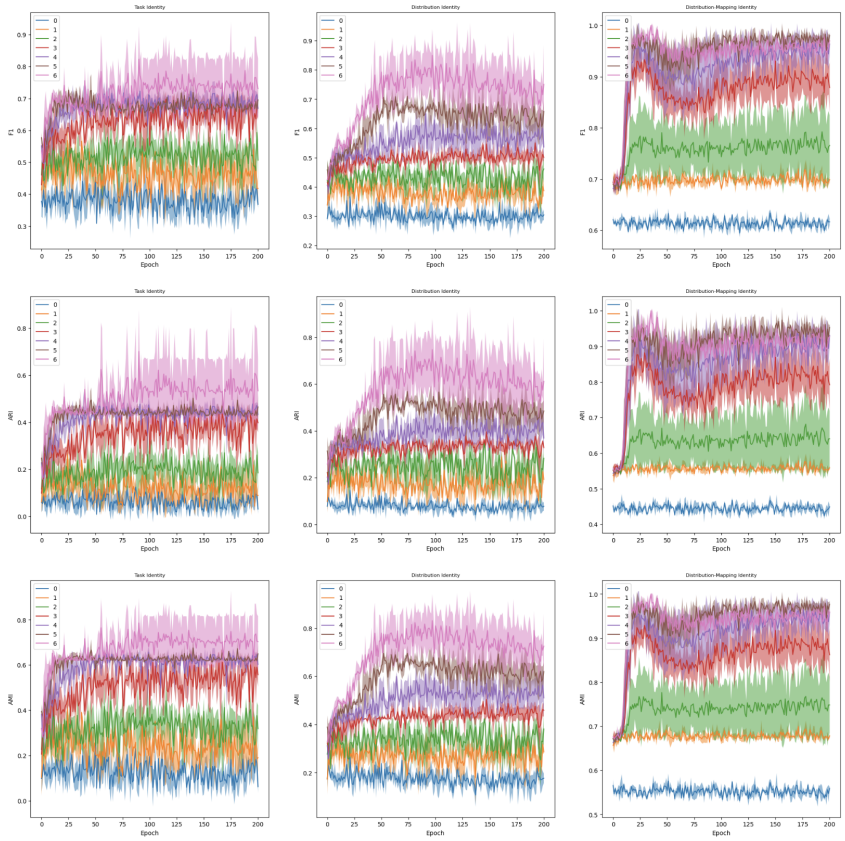


(a) KNN Percentage

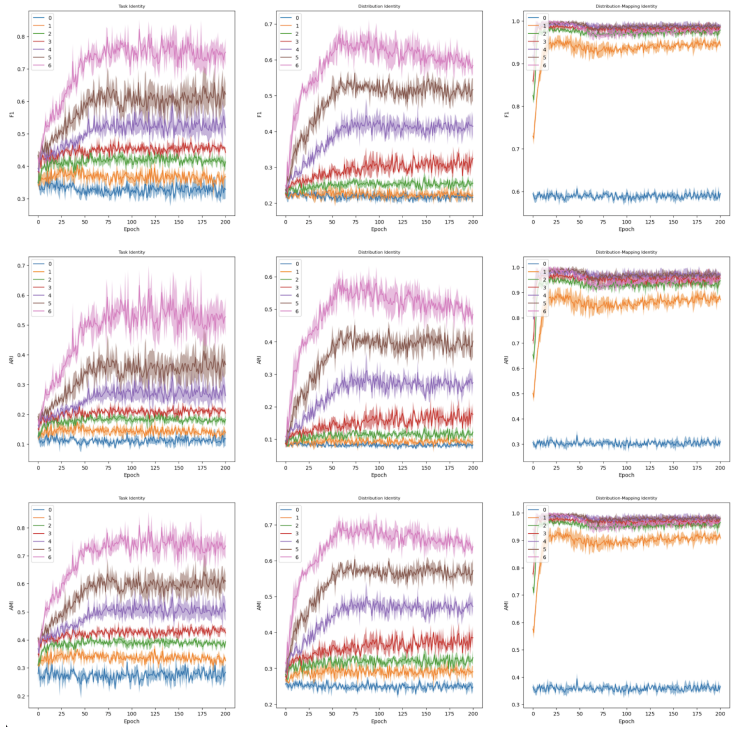


(b) KNN Accuracy

Figure 10: (a) Percentage of K nearest neighbors in the training set of an unseen instance belonging to the same task identity. (b) K nearest neighbors accuracy. Measurements are performed across all layers and throughout the training process. The results are shown for the model with 2048 hidden dimensions.

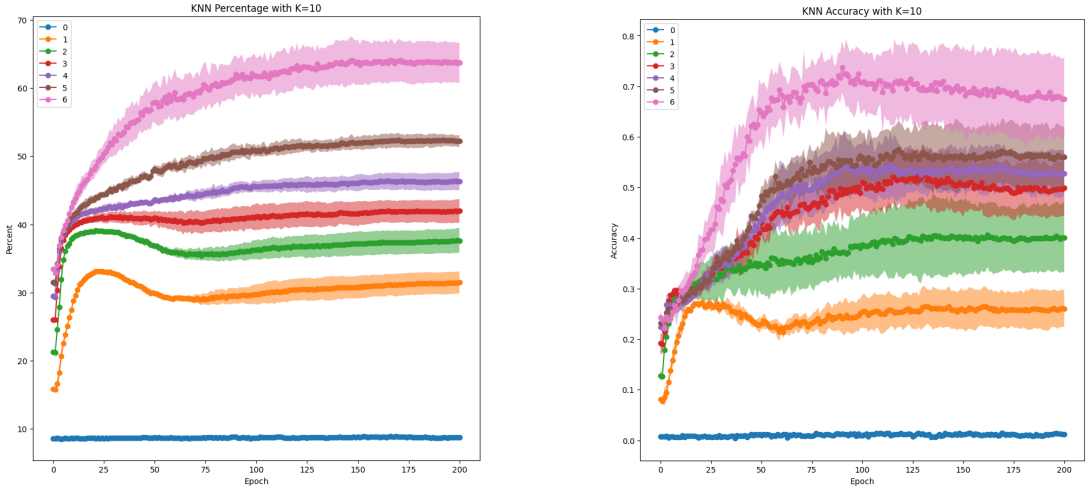


(a) Training subset



(b) Validation set

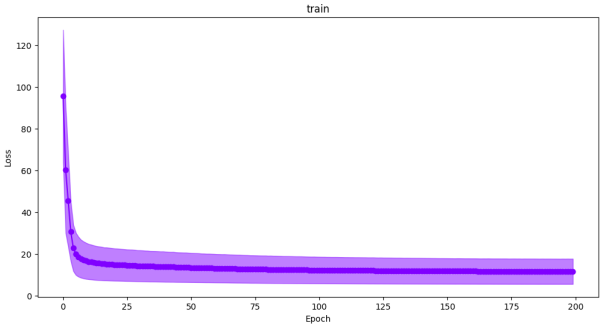
Figure 11: Clustering analysis on both training subset (a) and validation set (b) across different layers throughout the training process: The results are aggregated from repeated main experiments using different random seeds. Shaded areas represent variability via means and ± standard deviations. Different columns correspond to the uses of different identities as labels.



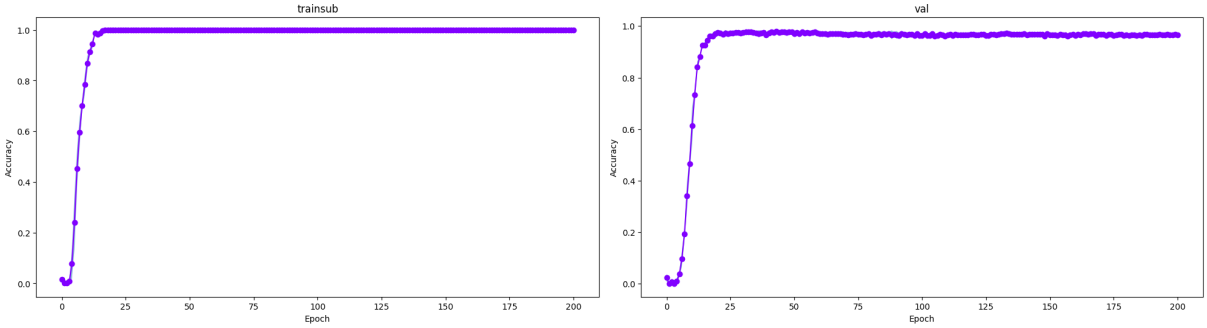
(a) KNN Percentage

(b) KNN Accuracy

Figure 12: (a) Percentage of K nearest neighbors in the training set of an unseen instance belonging to the same task identity. (b) K nearest neighbors accuracy. Measurements are performed across all of layers and throughout the training process. The results are aggregated from repeated main experiments using different random seeds. Shaded areas represent variability via means and ± standard deviations.



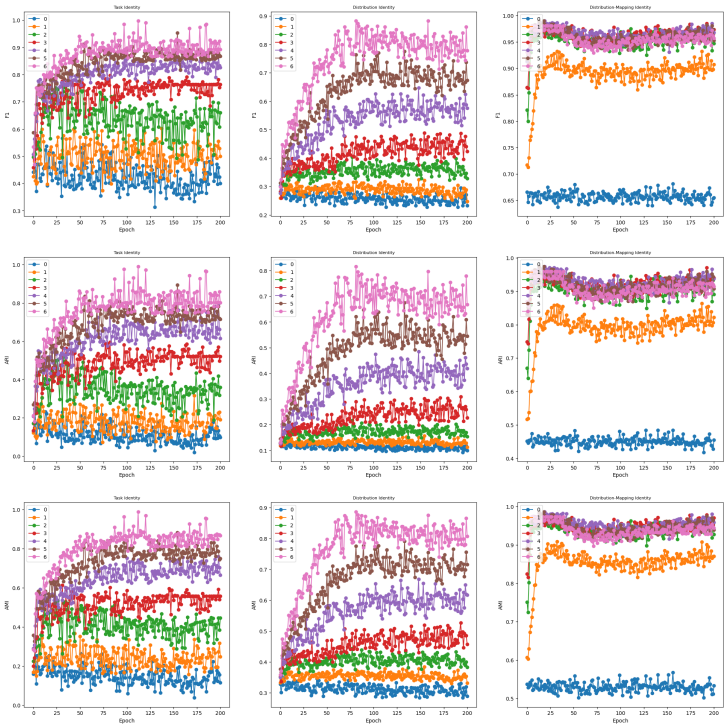
(a) Training Loss



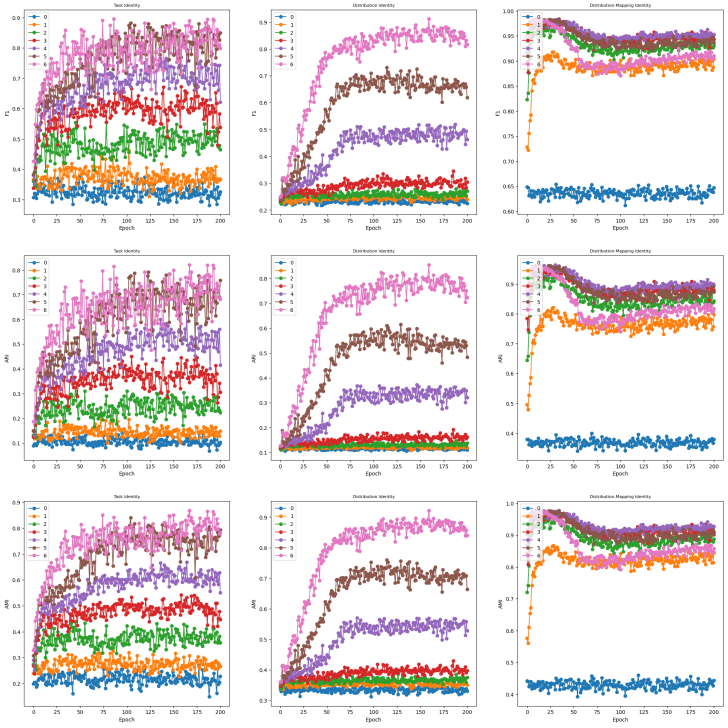
(b) Trainsub Task Accuracy

(c) Validation Task Accuracy

Figure 13: (a) Training loss, (b) Training subset task accuracy, and (c) validation task accuracy throughout the training process. Each dot represents a data point. Both (b) and (c) show dense dots with near-zero accuracy for the first few epochs. The results are aggregated from repeated main experiments using different random seeds. Shaded areas represent variability via means and ± standard deviations.



(a) Training subset



(b) Validation set

Figure 14: Clustering analysis on both training subset (a) and validation set (b) across different layers throughout the training process: The results are obtained by training a model on data with similar tasks. As detailed in Section B.1, instances sharing the same task identity have either identical or similar tasks, while those with the same distribution identity have identical tasks. We observe that instances with similar tasks form clusters, while instances with identical tasks form more specific, lower-level clusters.

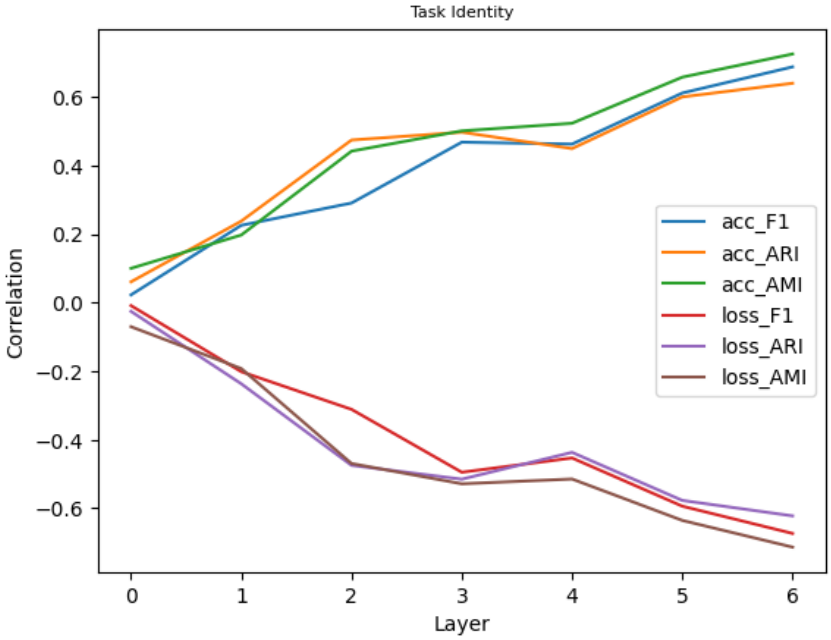


Figure 15: Correlation analysis between the task performance and the clustering performance. The task performance is measured by either the task accuracy or its cross-entropy loss. We follow the main experiments to use F1, ARI and AMI to measure the clustering performance based on the task identity. Experiments are performed on the validation data split. With a combination of different measurements shown in the legend, Pearson correlation coefficients (Freedman et al., 2007) are computed between the task performances and the clustering performances across the entire training process for each individual layer. We can observe that in all of the combinations, the correlation generally increases across layers and reaches a value over 0.6 (below -0.6) for performance measurements using accuracy/loss respectively, an indicator of a relatively strong correlation.