Transformer-based Causal Language Models Perform Clustering

Anonymous ACL submission

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 a concern. Recent works (Ouyang et al., 2022; Rafailov et al., 2023; Zhang et al., 2023) have shown great improvements on the instruction-following capability via 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.

1 Introduction

Recent years have seen noteworthy progress with large language models (LLMs) like GPT-3 (Brown et al., 2020), GPT-4 (OpenAI, 2023), and LLaMa (Touvron et al., 2023a), showcasing impressive capabilities across various natural language tasks. A significant challenge with LLMs is the misalignment between the training objective and users' intentions. Typically, 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). To address this mismatch, 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 their parameters and the data they have been trained on, their analysis is extremely challenging. For example, it is difficult to determine which token to focus on for 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 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 see if the model has any inductive bias.

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 the Transformer-based CLM trained for instruction-following. One intuitive hypothesis is that the hidden states corresponding to the same task 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. In Section 3, we find experimental evidence supporting the former.

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 the 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 the long training schedule and high computational costs. Our simplified setting reveals its advantage by allowing us to constrain the scope of the tasks such that a relatively small model is able to fully learn the task in 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.

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 suggesting 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. Further, we validate our findings from the simplified task in a more realistic setting.

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 such as "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. However, the opposite does not hold since this will lead to an ill-defined problem.

The input can be either integrated into the instruction or separated out. For simplicity, we use a separate input. 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 by making several assumptions. 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, 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 to multiple different output elements. For simplicity, we assume uniqueness. 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 the 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*. 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 A. 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 via 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 existences of task-specific clusters as shown in Figure 1, 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

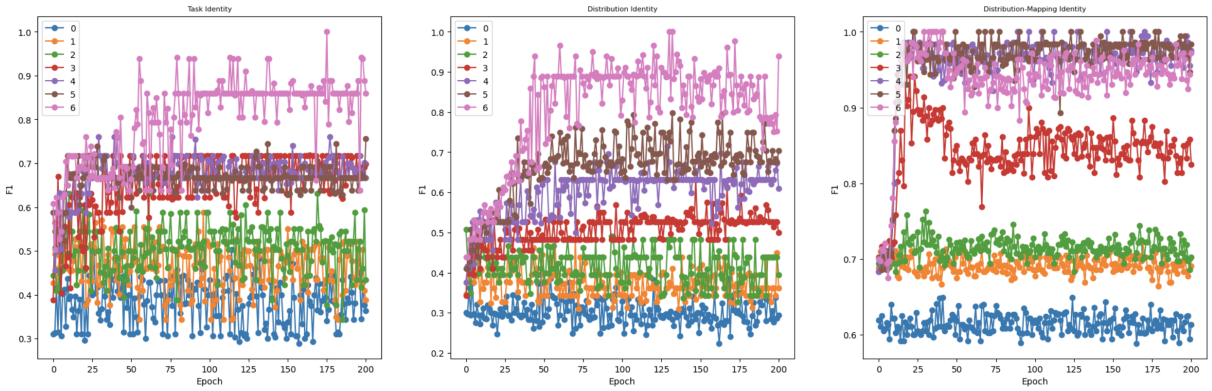
3.1 Implementation Details

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. Further details regarding the hyperparameters of the data generation process and statistics of the resulting datasets are in Appendix A.

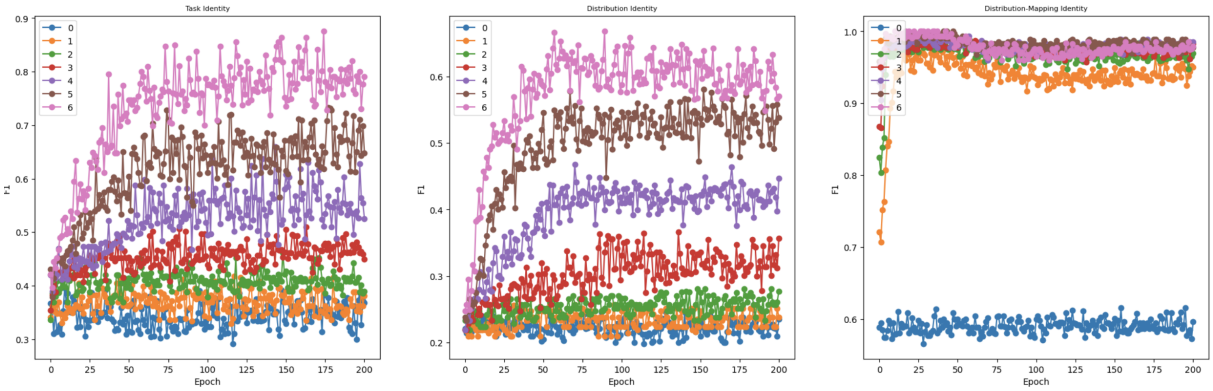
We train a six-layer Transformer model following the GPT-2 architecture (Radford et al., 2019). This model is optimized using an AdamW optimizer (Loshchilov and Hutter, 2017) and employs a cosine annealing learning rate schedule. We terminate training based on the best task accuracy achieved on the validation dataset. Additional specifics about the hyperparameters of the model and training process are in Appendix C. We perform all of experiments on a single NVIDIA A100 GPU.

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 employ 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



(a) Training subset



(b) Validation set

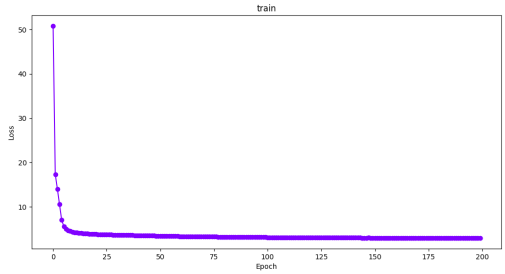
Figure 1: Clustering analysis on both of training subset (a) and validation set (b) across different layers throughout the training process: Different columns corresponds to uses of different identities as labels. Only shows results on F1 score here and see results on other evaluation metrics in Figure 4. Each dot represents a data point.

the subsequent clustering performance. We conduct extrinsic clustering evaluation on the clustering results, utilizing 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 regarding the clustering analysis and evaluation metrics are in Appendix B.

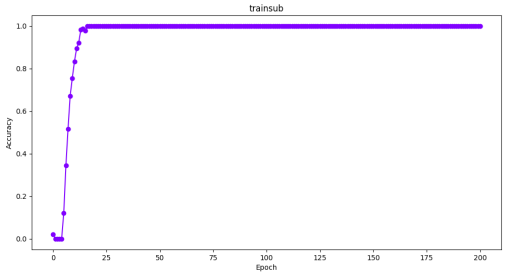
As shown in Figures 1, on both of training and validation splits, there exists a strong trend of improvement of the clustering performance based on task identities throughout the training process until saturating at some high values. Figure 4 in Appendix demonstrates results on all evaluation metrics. Especially, during late stage 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 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 2c, which may indicate correlation between the task performance and the clustering performance.

Moreover, clustering performance tends to improve 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 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. Besides, we design the simplified task to have many tasks share the same inputs by using a small task related vocabulary such that the model won't be able to identify a task solely from the inputs.

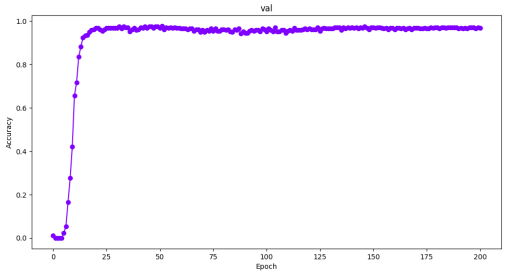
Also, interestingly, based on our formulation of the instruction-following task in Section 2.1,



(a) Training Loss



(b) Trainsub Task Accuracy



(c) Validation Task Accuracy

Figure 2: (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.

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 high F1 score of over 0.9 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 shows that the clusters evolve throughout the training process instead of appearing spontaneously.

We have found the 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 Figures 1, we found obvious clustering effects based on this setting, which reveals a 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.

Figure 2 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 2a. We hypothesize that the model initially learns task identification through the evolved clustering process by resolving various ambiguities introduced by us 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 large hidden dimension of 2048. See results on this additional models in Figures 5, 6 7 and 8 of the appendix.

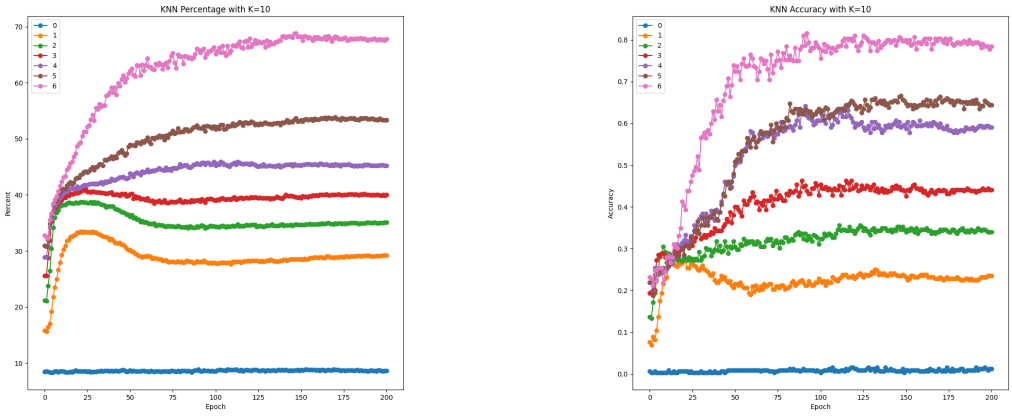
3.3 Advantages of Clustering

Next, we explore 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 3a shows the percentage of K nearest training instances of an unseen instance belonging to the same task identity, averaged over all instances. 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 the training proceeds. This suggests that 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) algorithms 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 record the task performance of our models based on the KNN during the training process. From Figure 3b, we observe that the 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 such that 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 an apparently 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.

3.4 Analysis of Natural Instruction-Following Task

One further question to ask is if the clustering phenomenon we discovered under the simplified setting generalizes to realistic settings. Therefore, we studied trained LLMs on a realistic instruction-following task based on natural language to supplement our analysis. We utilize tasks and their descriptions in natural language from (Hendel et al., 2023) of three categories: Knowledge, Linguistic, and Translation. 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 (Liu et al., 2023) 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 D. 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 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. Actually, the instruction-following data can be considered as a spacial kind of language data and naturally exist in the large-scale language data used for pre-training. Therefore, we will perform the same clustering analysis as in Section 3.2 on a number of different open LLMs either instruction tuned or not: LLaMa-7B, LLaMa-13B (Touvron et al., 2023a), GPT-J-6B (Wang and Komatsuzaki, 2021), LLaMa-2-7B-



(a) KNN Percentage

(b) KNN Accuracy

Figure 3: (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.

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 with using task identity, distribution identity and distribution-mapping identity as labels.

Instruct (Touvron et al., 2023b; Together, 2023) and Instruct-GPT-J-6B (Cloud; Taori et al., 2023), in which LLaMa-2-7B-Instruct and Instruct-GPT-J-6B are fine-tuned on instruction datasets. B (billion) refers to the number of parameters. See more details about the model sizes in Table 6 in the appendix.

We only report measurements of the layer with the best F1 score on clustering based on task identity. As shown 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. Besides, as we expected, the clustering phenomenon appears on LLMs either instruction tuned or not. We should note that the conclusions we made on the simplified setting may not completely extend to the realistic settings due to differences of data complexity and scales. However, we can still see some consistent results, which indicates some universality of the clustering phenomenon. We hope our analysis on both the simplified and realistic settings can shed some light on understanding the inductive biases of the Transformer-based LLMs for instruction following.

4 Related Work

4.1 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., 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.

4.2 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 instructions, 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.

4.3 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 in order to gain better controllability of the experiments and perform more in-depth analysis.

4.4 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 concentrate on studying clustering among tokens within a sequence. In contrast, our clustering analysis focuses on identifying clustering structures among different sequences.

5 Conclusion

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 from the clustering phenomenon for the model to handle unseen instances. We also further verify the existence of the clustering phenomenon we discovered from the simplified setting on a realistic setting. 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. Furthermore, this newfound understanding can inspire the development of more advanced algorithms to enhance LLM capability to effectively follow human instructions.

6 Limitations

At present, our 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 is an area we plan to explore in future research endeavors.

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.

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

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.

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*.

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.

Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Lin Zhao, Dajiang Zhu, Xiang Li, Ning Qiang, Dingang Shen, Tianming Liu, and Bao Ge. 2023. Summary of ChatGPT-related research and perspective towards the future of large language models. *arXiv:2304.01852*.

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*.

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*.

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*.

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 Simplified Setting

To construct data for the synthetic instruction-following task under the simplified setting, we firstly sample a task function consisting of a number of 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. 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 a number of 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 a number of instructions sampled from all of the distributions associated with each of the task. 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 sythetic instruction-following dataset in Table 2.

B 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 utilizes 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 $2 \times \frac{precision \times recall}{precision + recall}$. 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 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.

C Hyperparameters

Tables 3 shows the hyperparameters of the data generation process. Tables 4 contain the hyperparameters of our Transformer model and its training process. T-SNE related hyperparameters are listed in Table 5

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.

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 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
feedforwark network dimension	1024
droptout	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: T-SNE Hyperparameters (Van der Maaten and Hinton, 2008).

Model	Hidden Dimension	Parameter Count
Our model 768	768	23 million
Our model 32	32	55 thousand
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 6: 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.

D Natural Instruction-Following Task

D.1 ChatGPT Prompt Template

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

D.2 Realistic Setting

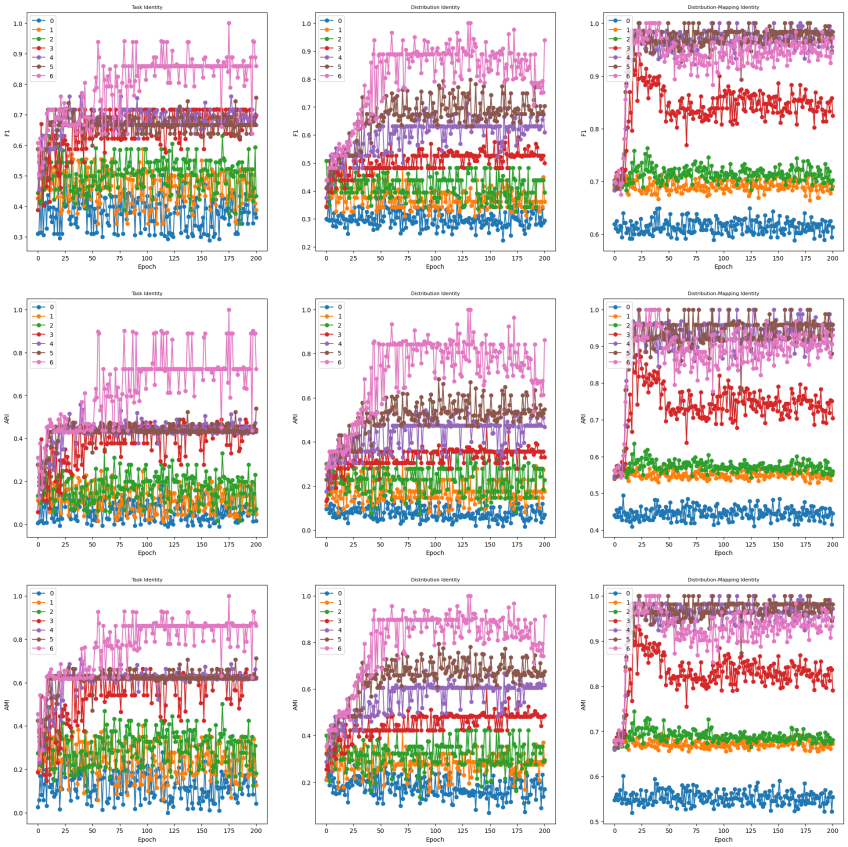
See Table 7 for the task descriptions used for constructing the dataset for the realistic setting as detailed in Section 3.4 and data statistics in Table 2.

E More Results

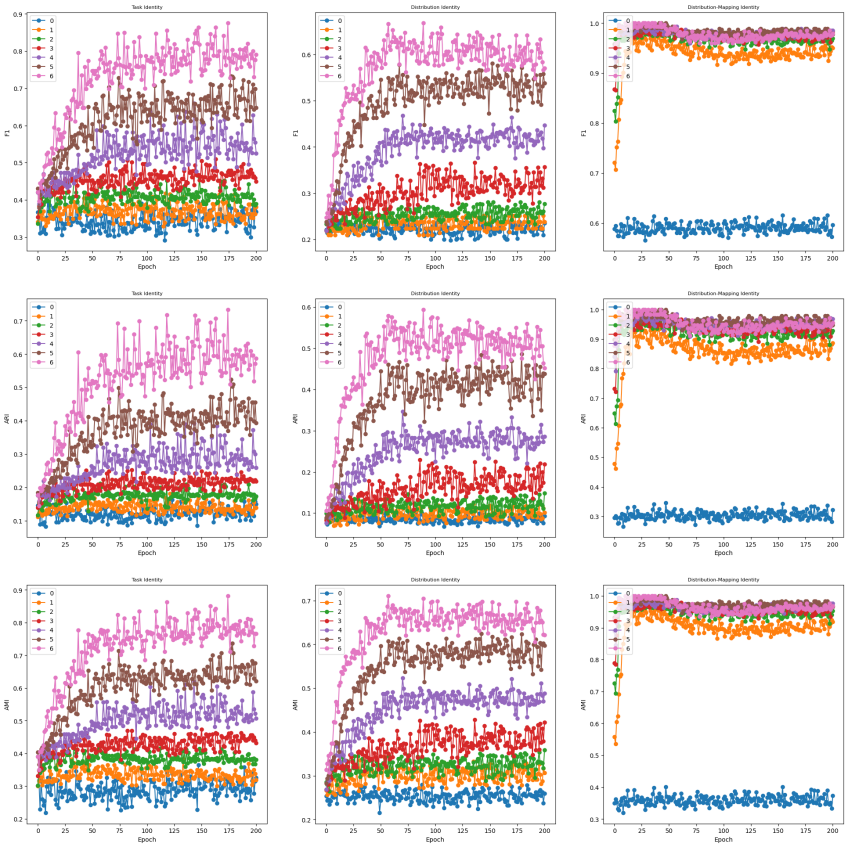
We present results obtained on various models here.

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 7: Task descriptions provided by (Hendel et al., 2023)

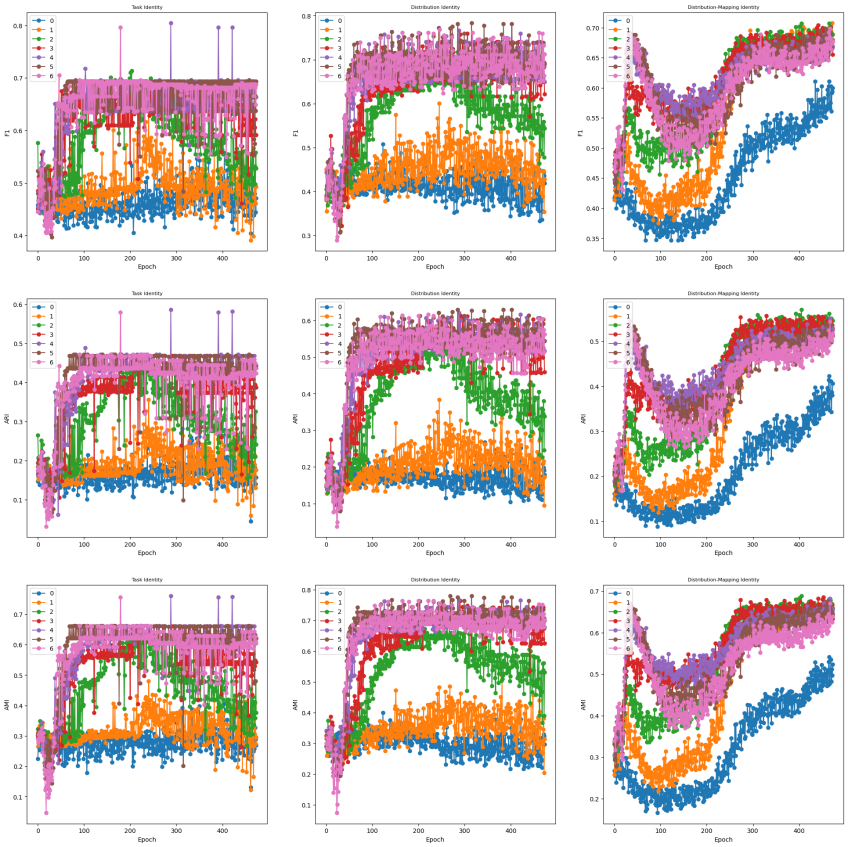


(a) Training subset

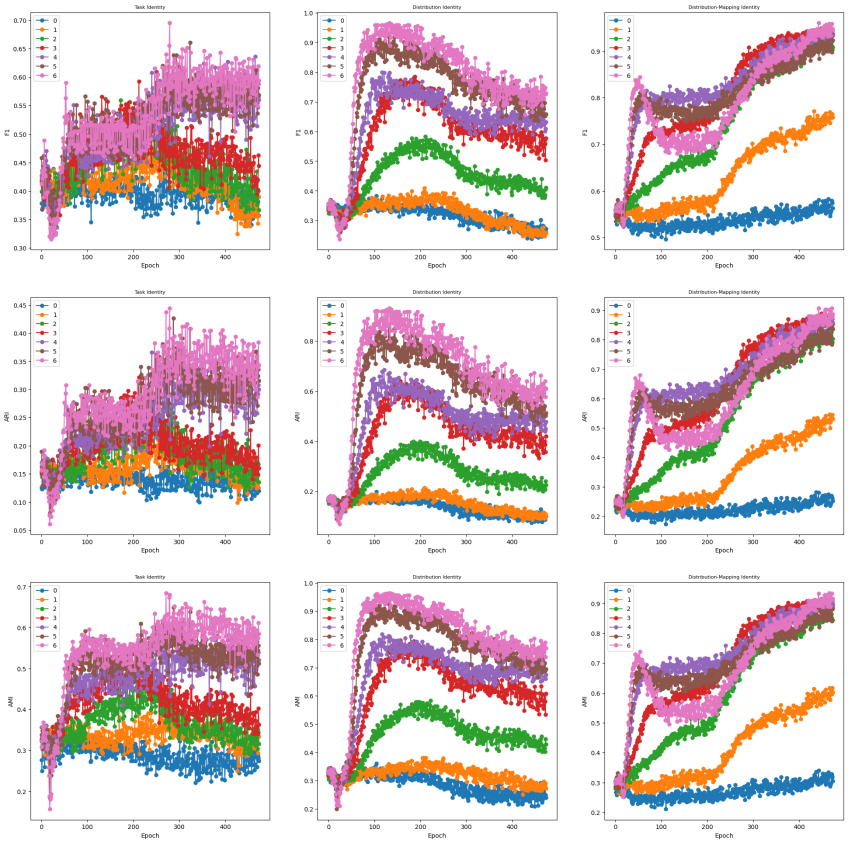


(b) Validation set

Figure 4: 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.

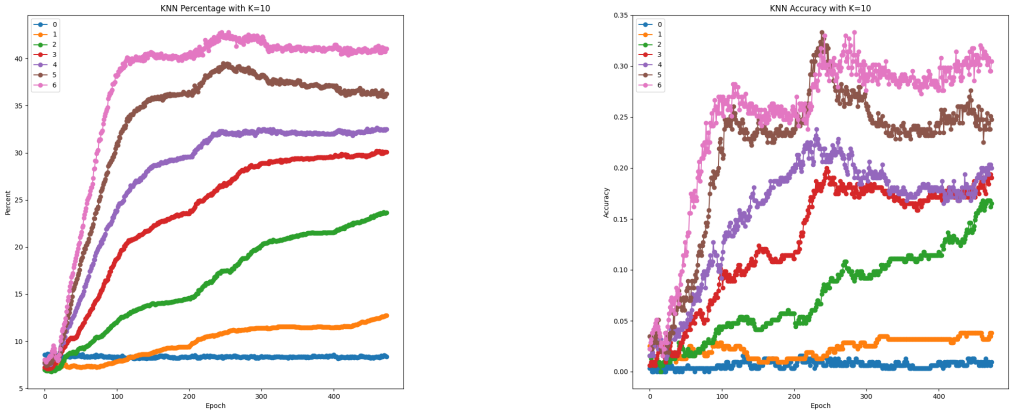


(a) Training subset



(b) Validation set

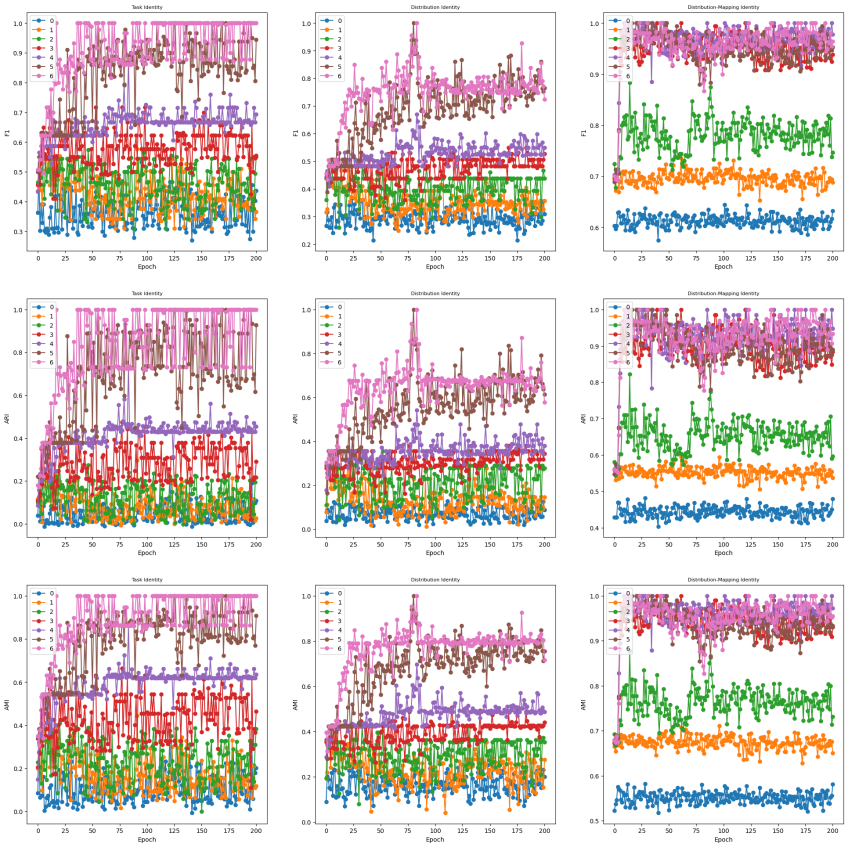
Figure 5: Clustering analysis on both of training subset (a) and validation set (b) across different layers throughout the training process: The results are shown for the model with 32 hidden dimension. We train this model for 500 epochs due to its slow convergence. Different columns corresponds to uses of different identities as labels.



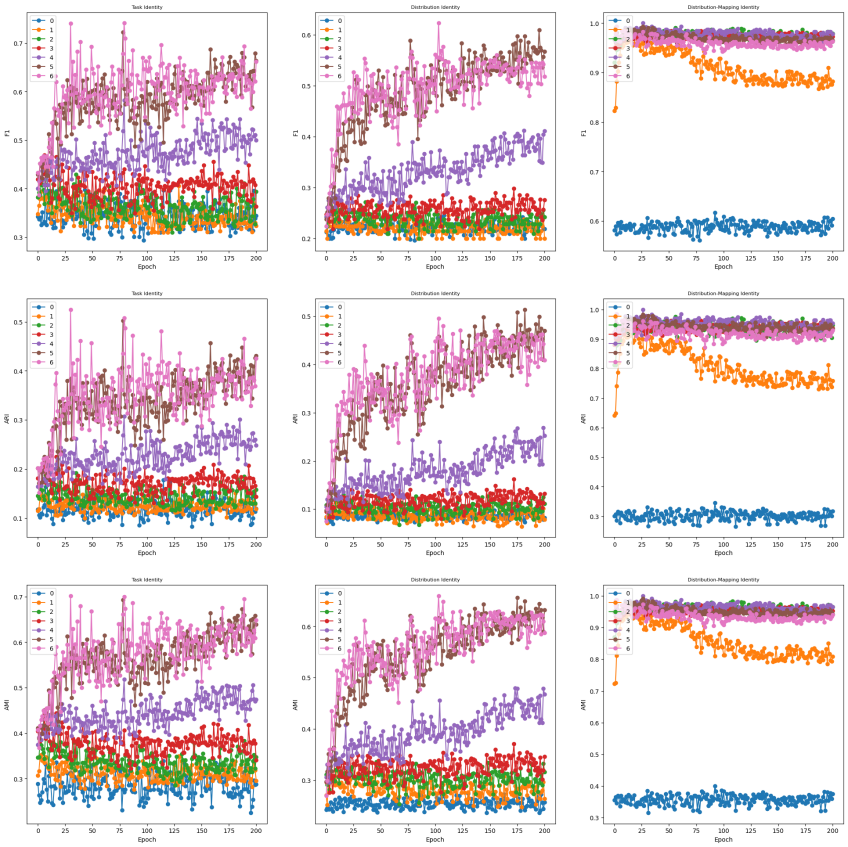
(a) KNN Percentage

(b) KNN Accuracy

Figure 6: (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 2048 hidden dimension.

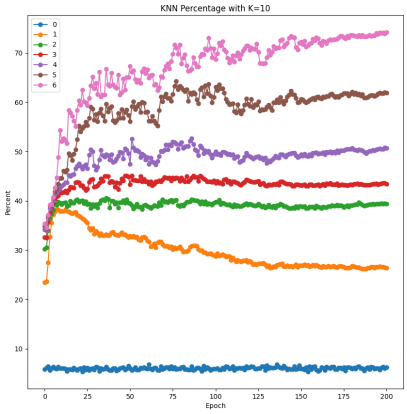


(a) Training subset

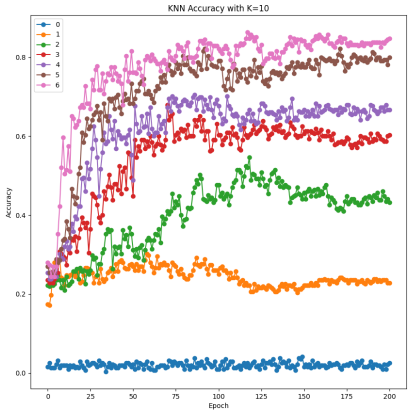


(b) Validation set

Figure 7: Clustering analysis on both of 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 corresponds to 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.