STG-MTL: Scalable Task Grouping for Multi-Task Learning Using Data Maps

Ammar Sherif [1] Abubakar Abid [2] Mustafa Elattar [1] Mohamed ElHelw [1]

Abstract

Multi-Task Learning (MTL) is a powerful technique that has gained popularity due to its performance improvement over traditional Single-Task Learning (STL). However, MTL is often challenging because there is an exponential number of possible task groupings, which can make it difficult to choose the best one, and some groupings might produce performance degradation due to negative interference between tasks. Furthermore, existing solutions are severely suffering from scalability issues, limiting any practical application. In our paper, we propose a new data-driven method that addresses these challenges and provides a scalable and modular solution for classification task grouping based on hand-crafted features, specifically Data Maps, which capture the training behavior for each classification task during the MTL training. We experiment with the method demonstrating its effectiveness, even on an unprecedented number of tasks (up to 100).

1. Introduction

Multi-Task Learning (MTL) has emerged as a powerful technique in deep learning (Zhang & Yang, 2022; Crawshaw, 2020) that allows for joint training of multiple related tasks, leading to improved model performance compared to traditional Single-Task Learning (STL). By leveraging shared representations and knowledge across tasks, MTL enhances generalization and mitigates overfitting. Furthermore, MTL promotes faster learning of related tasks and alleviates the computational requirements of deep learning, making it particularly valuable in scenarios with limited task-specific data. That is why MTL has gained significant attention in various domains, including computer vision (Fan et al., 2017; Misra et al., 2016; Standley et al., 2020), natural language processing (Zhang et al., 2022b; Peng et al., 2020; Jin et al., 2020; Bickel et al., 2008), speech recognition (Huang et al., 2022b; Zhang et al., 2019), and healthcare (Peng et al., 2020; Bao et al., 2022; Islam et al., 2021), and has shown promising results in improving accuracy, robustness, and efficiency. However, effectively harnessing the potential of MTL poses several challenges, including the identification of optimal task groupings (Song et al., 2022; Fifty et al., 2021; Standley et al., 2020) and the management of negative interference between tasks (Sener & Koltun, 2018; Wu et al., 2020; Maninis et al., 2019).

The task grouping problem in MTL is particularly challenging due to the exponential number of possible task combinations (Aribandi et al., 2021; Fifty et al., 2021; Standley et al., 2020; Song et al., 2022). What makes it worse for the exhaustive search is that each trial involves a complete training and evaluation procedure, leading to computational and optimization burden. Moreover, inappropriate task groupings may result in performance degradation due to negative transfer between tasks (Sener & Koltun, 2018; Wu et al., 2020; Maninis et al., 2019). Existing solutions have struggled to address these challenges, often suffering from scalability and modularity issues, making their practical application in real-world scenarios nearly infeasible.

In this paper, we propose a novel data-driven method for task grouping in MTL for classification tasks, which overcomes the scalability and modularity limitations. Our method utilizes the concept of Data Maps (Swayamdipta et al., 2020), hand-crafted features that capture the training behavior of each classification task during MTL training. By analyzing these data maps, we can identify task groupings, both hard and soft ones, that promote positive transfer and mitigate negative interference as much as possible. We demonstrate the effectiveness of our method through extensive experimentation, including experiments on an unprecedented number of tasks, scaling up to 100 tasks to emphasize the practicality of our approach.

The contributions of this paper can be summarized as follows:

- We propose a novel data-driven method for task grouping in MTL, addressing the challenges of scalability and modularity.

[1]Nile University, Giza, Egypt [2]Hugging Face, New York, United States. Correspondence to: Ammar Sherif <ammarsherif90@gmail.com>.

- We propose a mechanism that utilizes our soft-grouping results, enabling model specialization via loss weighting.

- We conduct extensive experiments, demonstrating the effectiveness of our method, even on a large number of tasks (scaling up to 100 classification tasks).

2. Related Work

MTL has been extensively studied to leverage the benefits of information sharing among related tasks, which can serve as an inductive bias to improve modeling performance (Caruana, 1997; Zhang & Yang, 2022). Another perspective on MTL is that it enables more efficient utilization of the model capacity by focusing on learning relevant features and reducing the impact of irrelevant signals, which contributes to overfitting, leading to better generalization. However, when tasks lack shared information, they compete for the limited model capacity, resulting in performance degradation (Sener & Koltun, 2018; Wu et al., 2020; Maninis et al., 2019). To address this challenge, task grouping has emerged as a promising solution to identify subsets of tasks that can be trained together, avoiding negative interference and promoting improved performance.

Traditionally, the decision of task grouping has been approached through costly cross-validation techniques or human expert knowledge (Zhang & Yang, 2022). However, these methods have limitations when applied to different problem domains and may not scale well. Some attempts have been made to approach the problem differently enabling the models to automate the search over which parameters to share among particular tasks (Zhang et al., 2022a; Misra et al., 2016). Methods such as Neural Architecture Search (Liu et al., 2018; Huang et al., 2018; Chen et al., 2023; Zhang et al., 2022a; Sun et al., 2020; Vandenhende et al., 2019), Soft-Parameter Sharing (Ruder et al., 2019; Long et al., 2017; Misra et al., 2016), and asymmetric information transfer (Lee et al., 2016; 2018; Huang et al., 2022a) have been developed. However, these models often exhibit poor generalization and struggle to perform well on diverse tasks and domains. Besides, they often require a large model capacity and do not thus scale well with a large number of tasks.

Therefore, gradient-based approaches (Fifty et al., 2021; Strezoski et al., 2019) have also been explored to determine task grouping in advance. The Task Affinity Grouping (TAG) approach (Fifty et al., 2021), which leverages gradients to determine task similarity, is an example of such an approach. Nevertheless, it has complex training paradigm and requires $\Theta(N^2)$ more forward and backward passes to compute the inter task affinities, putting an issue with scalability even if we enhance the solution's modularity. Another method, called Higher-Order Approximation (HOA) (Standley et al., 2020), reduces the exponential number of MTL training, from the exhaustive search, by considering only the quadratic pairs of task combinations. However, even with such relaxation, the scalability of HOA remains limited, particularly when dealing with a large number of tasks.

The task grouping problem has been addressed in recent studies through a Meta-Learning approach (Song et al., 2022), aiming to create a meta-learner that can estimate task grouping gains. Nevertheless, the computational demands of this approach pose practical challenges for real-world applications; it requires training MTL networks for every chosen task combination in the training set for multiple iterations. It furthermore outputs all the possible gains of every task combination, whose numbers grow exponentially, and runs a search algorithm over these exponentially growing gains to find the optimal grouping. As a result, the scalability of this solution is severely limited, making it less feasible for a larger number of tasks.

3. Task Clustering using Data Maps

Now, we elaborate in the components in our method in the next sections. We start with stating the notations we will use along with our MTL architecture we are using in our experiments in Section 3.1. Then, we move on to illustrate the data maps, which is crucial component of our method in Section 3.2. In Section 3.3, we talk regarding the approaches we use to cluster the tasks. We also introduce our evaluation mechanism of our task grouping in Section 3.4. Finally, we conclude this part with a simple theoretical comparison of our method and the literature from the perspective of scalability and modularity in Section 3.5. Figure 1 provides an overview of our method.

3.1. Preliminaries

Notations In our paper, we use the following notations consistently. The set of all tasks is denoted as $T = \{T_1, \dots, T_n\}$, where n represents the number of tasks and $|T| = n$. The total number of training data points is denoted as N. We calculate the data maps at specific epochs, and the set of epochs is represented as $E = \{E_1, \dots, E_k\}$, where E_i corresponds to the i^{th} epoch. The task clusters are denoted by $C = \{C_1, \dots, C_m\}$, and each cluster C_i has an associated centroid c_i. The participation of each task i in cluster j is represented by $w_{i,j}$, with the constraint that $\sum_{j=0}^{|C|} w_{i,j} = 1$, indicating the percentage of membership; W_i is the weight vector of the tasks in cluster j. The values of $w_{i,j}$ range from 0 to 1, where 1 signifies full membership and 0 indicates no membership.

MTL Architecture The MTL procedure for a given

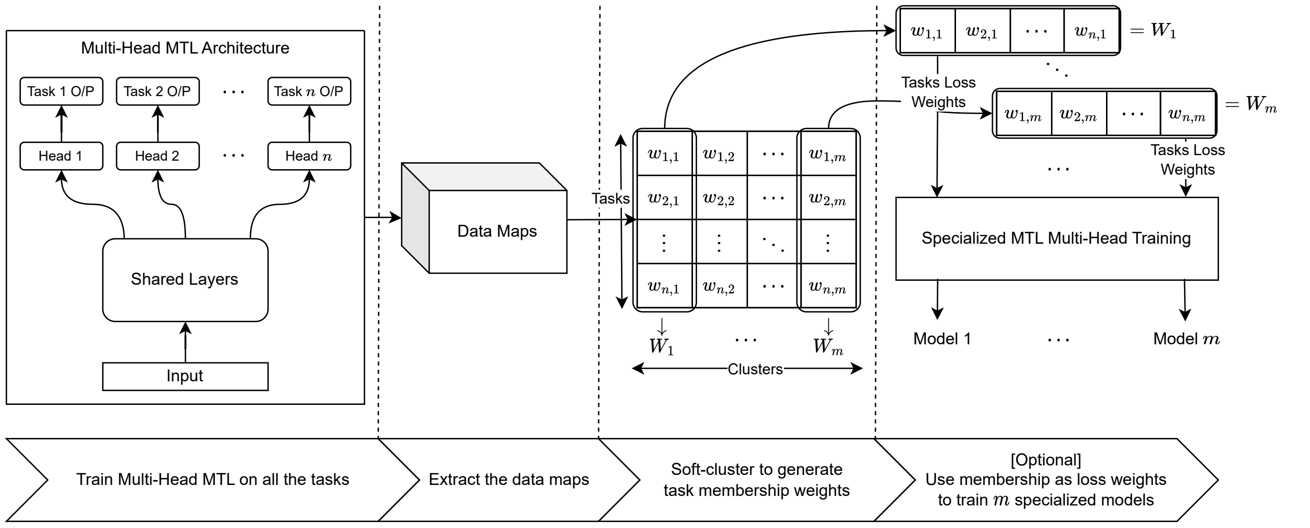

Multi-Head MTL Architecture
Task 1 O/P
Task 2 O/P
Task n O/P
Head 1
Head 2
Head n
Shared Layers
Input
Data Maps
Tasks
$w_{1,1}$ $w_{1,2}$ $w_{1,m}$
$w_{2,1}$ $w_{2,2}$ $w_{2,m}$
$w_{n,1}$ $w_{n,2}$ $w_{n,m}$
W_1
W_m
Clusters
$w_{1,1}$ $w_{2,1}$ $w_{n,1}$ $= W_1$
$w_{1,m}$ $w_{2,m}$ $w_{n,m}$ $= W_m$
Tasks Loss Weights
Tasks Loss Weights
Specialized MTL Multi-Head Training
Model 1
Model m
Train Multi-Head MTL on all the tasks
Extract the data maps
Soft-cluster to generate task membership weights
[Optional] Use membership as loss weights to train m specialized models

Figure 1. **Overview** of our method to cluster the tasks using Data Maps. (1) we use a single Multi-head Multi-Task Learning architecture to jointly train all the tasks. Each head is task-specific layers. (2) we extract the data maps of all the tasks across the epochs in E. (3) we use the data maps to cluster the tasks using kmeans and generate the memberships according to Equation 3. (4) to evaluate our clustering results, we train m models where each model represents a cluster focusing on particular tasks using the memberships as loss weights.

task combination, consisting of τ tasks denoted as $\{T_{a_1}, \ldots, T_{a_\tau}\}$, is defined as training with a joint objective for these tasks (Equation 1) where L is the accumulated loss value of the cluster, L_k is the loss of the k^{th} task, and $w_k \in [0, 1]$ is an optional task weight of the k^{th} task.

$$L = \sum_{k=1}^{\tau} w_k \cdot L_k \tag{1}$$

Following the previous approaches (Fifty et al., 2021; Standley et al., 2020; Song et al., 2022), we utilize a commonly employed hard-sharing multi-head architecture (Figure 1) for all our MTL experiments, where a single feature extractor is used to obtain shared representations, and separate task-specific heads are employed to output the result. Additionally, for all the experiments, we maintain the same data splits, via prior seeding, and keep the optimization algorithm and other hyperparameters fixed; this is to make sure any variability in the performance is only attributed to the task grouping and the corresponding weights if any.

3.2. Data Maps as Task Features

Data Maps (Swayamdipta et al., 2020), originally developed as a model-based tool for characterizing and diagnosing NLP datasets, serve as a valuable component in our approach. They leverage the model behavior concerning individual data point instances of the training set for each task. In our work, we employ Data Maps as task features due to their simplicity, scalability, and ability to extract them on the fly without prior knowledge of the model architecture, thus enhancing the modularity of our approach.

The concept behind Data Maps revolves briefly around extracting two essential values for each data point: the *model confidence* (μ) of the true class, which is the average probability of the true class over some epochs, and the *variability* (σ) of this confidence, which is the standard deviation of the true class probabilities over the same epochs. For a particular task, the data map shape is $(N, 2)$ where N is the training size. Figure 2 shows an example of the resulting Data Map for an example task extracted from CIFAR10 dataset (Krizhevsky et al., 2009).

Because their information is very task-dependent, we thought they can serve as task descriptors. To further enhance the expressiveness of the extracted features, we also extract data maps at various epochs, allowing us to gain insights into their evolution over time; the resulting shape in such case is $(|T|, |E|, N, 2)$. Therefore, by analyzing their characteristics, over the different epochs during training, we can capture crucial information about the relatedness of each task.

In the extraction of data maps, we employ two approaches. The first approach involves building a single MTL model that incorporates all tasks and extracting the data maps directly from this unified model. Alternatively, we utilize the second approach, where individual models are constructed for each task, resulting in multiple STLs, and merging the data maps obtained from each model. Our results are primarily based on the first approach, as it offers the advantage of a single training procedure, simplifying computational complexity, and streamlining experimentation, while having the same qualitative results as the STL.

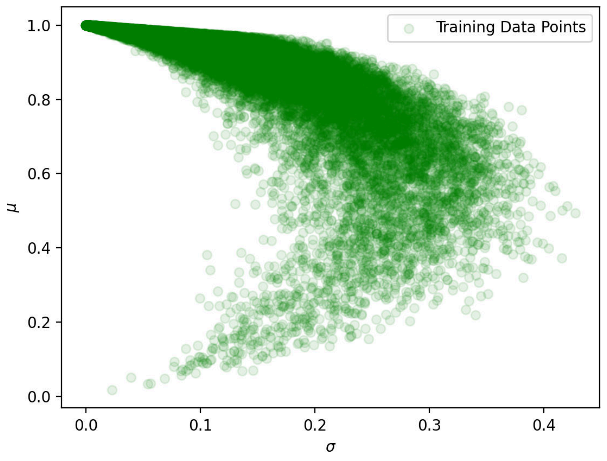


Figure 2. An example of a generated data map for the "Living being" task after 21 epochs of co-training on 15 tasks of G2 (Section 4.1)

3.3. Task Clustering

With the extracted data maps in hand, our next step is to group the tasks into clusters based on their similarity. We propose three distinct approaches for task clustering: soft clustering, hard clustering, and point-based soft clustering.

In both hard and soft clustering, we represent each task as a vector by concatenating the corresponding data maps. In the case of hard clustering, we employ the k-means algorithm (Lloyd, 1982) to cluster these task vectors, aiming to identify distinct clusters of tasks. To introduce a more nuanced representation of task similarities, we incorporate a modified version of the fuzzification step (Equation 3), from (Bezdek et al., 1984), into our approach, which enables soft clustering, where x_i represents the i^{th} task vector of the corresponding data maps and $F > 1$ represents the fuzzification index. This fuzzification process assigns soft memberships to tasks, allowing for more flexible and comprehensive clustering results. We predominantly rely on the soft clustering approach due to its effectiveness and reliability.

$$w_{i,j} = \frac{1}{\sum_{k=1}^{|T|} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{F-1}}} \tag{2}$$

$$= \frac{\|x_i - c_j\|^{\frac{-2}{F-1}}}{\sum_{k=1}^{|T|} \left(\|x_i - c_k\| \right)^{\frac{-2}{F-1}}} \tag{3}$$

In early experiments, though, we also explored a point-based clustering approach to determine the participation membership of tasks. This approach involves clustering each instance point per task within each data map. Each data point then serves as a vote for its corresponding task, and the participation membership of each task is calculated based on the percentage of data points within the cluster (Equation 4) where $d^i_{k,e}$ represents the k^{th} data point of the data map taken at epoch e for task i.

$$w_{i,j} = \frac{\sum_{k=1}^{N} \sum_{e=1}^{|E|} [d^i_{k,e} \in C_j]}{|E| \cdot N} \tag{4}$$

However, we do not heavily rely on this point-based approach in our method. This is because it treats the data points within a data map in isolation, failing to capture the abstract behavior specific to each task and overlooking the evolution of data maps over different epochs.

3.4. Model Specialization through Loss Weighting

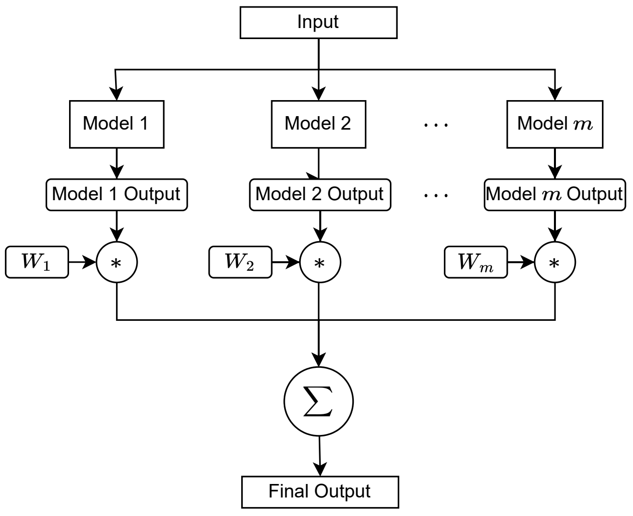


Figure 3. The procedure to use our specialized trained models to infer the results

In order to assess the effectiveness of our task grouping results, we use loss weighting as a method of model specialization. We construct MTL models that are tailored to specialize in specific sets of tasks based on the membership weights obtained from soft clustering results. For each cluster, we build an individual MTL model that focuses on the tasks assigned to that cluster according to their corresponding weights (Equation 1). To evaluate the performance of our solution, we apply the weighted average of the models' outputs according to the membership weights as in Equation 5, where O is our final output, O_k is the output according to the k^{th} cluster, and W_k is the weight vector of the k^{th}

Table 1. Comparison of asymptotic growth of the equivalently trained number of MTL models to get task grouping of various methods and ours

METHOD	EFFECTIVE MTL MODELS (↓)
EXHAUSTIVE SEARCH	$\Theta\left(2^N\right)$
HOA	$\Theta\left(\binom{N}{2}\right) = \Theta\left(N^2\right)$
TAG	$\Theta\left(N^2\right)$
MTG-NET	$\Theta\left(N \cdot K\right)$
STG-MTL	$\Theta\left(1\right)$

cluster. Figure 3 provides an overview of these operations while inferring the output. By comparing the resulting values with both STL and traditional MTL schemes, we can gain insights into the benefits and improvements brought by our task grouping approach.

$$O = \sum_{k=1}^{m} W_k \cdot O_k \tag{5}$$

3.5. Theoretical Scalability Comparison

In the theoretical scalability comparison, we evaluate our method against existing literature, focusing on the number of models required for clustering. Table 1 presents the comparison, where lower numbers indicate better scalability. Our approach stands out with excellent scalability, as it only necessitates training a single MTL model to extract data maps and perform clustering, or $\mathcal{O}(N)$ if we consider extracting data maps from STL models. This offers the most promising scalability potential for a larger number of tasks. That is why we can scale our experiments to a very large number of tasks as in Section 4.

Notice TAG requires one single MTL training, yet this is a customized training procedure where each epoch is effectively processed $\Theta(N^2)$, $\binom{N}{2}$ in particular, times to compute the inter-task affinities pairs, which like the other methods limits its scalability. Therefore, one MTL training of TAG utilizes the same compute of $\Theta(N^2)$ MTL models trained within the other methods normally.

Furthermore, our method's data map computation is performed on the fly, making it both model and task agnostic. This feature enhances the modularity of our approach, enabling effortless adaptation to different model architectures and tasks without manual intervention.

4. Experiments

In this section, we present a comprehensive overview of our experiments, focusing on assessing the effectiveness of our method and presenting the corresponding results. Section 4.1 outlines the specifics of the datasets utilized in our experimentation, as well as the tasks employed. In Section 4.2, we delve into the details of the model architecture and the hyperparameters used during experimentation. The outcomes of the soft clustering of tasks are presented in Section 4.3, where we highlight the effectiveness of our approach in grouping related tasks. Finally, in Section 4.4, we evaluate the quality of the obtained clustering results comparing them to STL and MTL results.

4.1. Datasets and Tasks

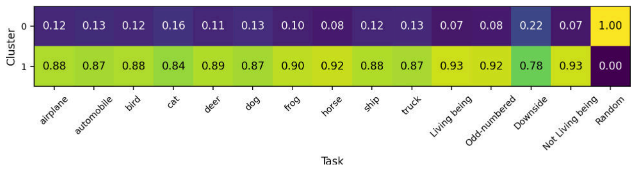
Our task generation is based on the CIFAR10 and CIFAR100 datasets (Krizhevsky et al., 2009). We define three groups of tasks for our experiments. In Group 1 (**G1**), we include binary classification tasks that determine whether an image belongs to a specific label in CIFAR10 or not. G1 consists of 10 tasks: {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck}.

Group 2 (**G2**) expands on G1 by introducing additional tasks on CIFAR10. These tasks include {Living being, Odd-numbered, Downside, Not living being, random}. The "Living being" task aims to detect whether an image contains a living being, which includes images with the labels {bird, cat, deer, dog, frog, horse}. Similarly, the "Not living being" task focuses on identifying non-living beings; these are { airplane, automobile, ship, truck} classes in CIFAR10. Notably, "Living being" and "Not living being" are intentionally designed to be similar tasks for testing purposes. The "Odd-numbered" task identifies whether the label of a CIFAR10 image is odd or not, encompassing {automobile, cat, dog, horse, truck} classes. Additionally, we flip half of CIFAR10 images and create a task to train the model to recognize vertically flipped images, the "Downside" task. Lastly, the "random" task assigns random binary labels to the entire dataset with a predefined seed for consistency and reproducibility. It is worth mentioning that while the original tasks in G1 are imbalanced, the extra tasks in G2 are all balanced ones.

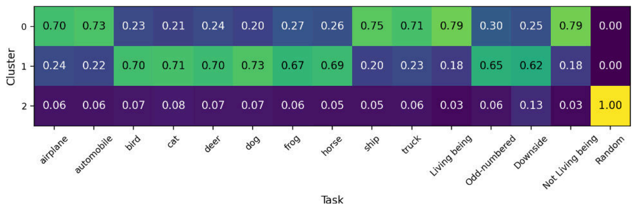
Group 3 (**G3**), similar to G1, consists of 100 binary classification tasks using the CIFAR100 labels. We also utilize the 20 super labels of CIFAR100 as our ground truth for task clustering evaluation. It is worth mentioning that CIFAR100 super labels are not intended for task grouping, so they are not grouped based on visual similarities like our method's objective. Instead, they are mostly clustered semantically, even though there are some exceptions like mushrooms and the classes of vehicles 1 and 2. Still, we think they serve as an informative indicator of the effectiveness of our approach, especially in the visually coherent superclasses.

4.2. Model Architectures and Hyper-Parameters

For all our experiments, we adopt the RESNET18 architecture (He et al., 2016) as our base model. Our method is

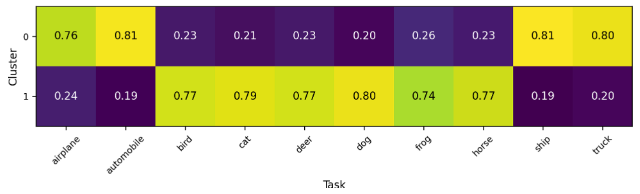


(a) 2 clusters

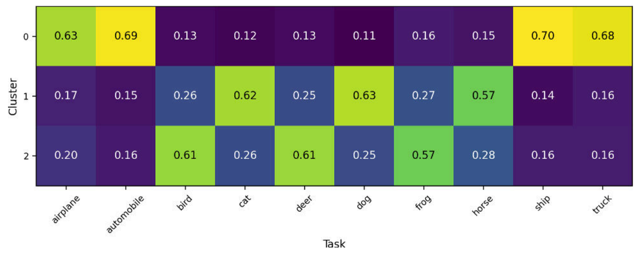


(b) 3 clusters

Figure 4. Task grouping of G2 with $F = 2$



(a) 2 clusters



(b) 3 clusters

Figure 5. Task grouping of G1 with $F = 2$

model-agnostic, so we have also experimented with models with much less complexity, yet we use RESNET18 considering it has moderate model capacity. Furthermore, we utilize it without any pre-training, ensuring that the model starts from scratch for each task grouping scenario. The last fully connected layer of RESNET18 serves as the task heads, with the number of output neurons corresponding to the number of tasks. Each neuron in the task heads represents a specific classification task. Throughout our experiments, the rest of the network, excluding the task heads, is shared among all tasks. Also, we train the model for 50 epochs in all our experiments: to extract data maps and to evaluate the models. Additionally, in our clustering process, we primarily set the fuzzification index (F) to 2, unless explicitly mentioned otherwise. The fuzzification index controls the level of fuzziness in the soft clustering algorithm, so increasing it produce softer decisions.

In terms of the loss function, we utilize Binary Cross Entropy as the binary classification loss for our tasks. However, to address the issue of task imbalance, we incorporate a penalty on positive instances for each task. By applying this penalty, we ensure that the model pays more attention to the minority label during training, thereby mitigating the impact of the imbalance and promoting better overall performance. Finally, it worth mentioning that we do not perform any kind of tuning to any model. We use the same basic settings in all of our experiments.

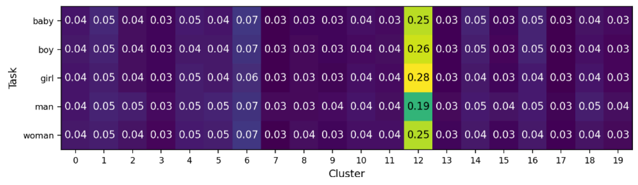
4.3. Task Clustering Results

Results of our task clustering experiments are presented for all groups. We initially experimented on G2, generating their data maps as described in Section 3.2 and Clustering them as in Section 3.3, as depicted in Figure 4. Notably, our method successfully clustered the "random" task separately, indicating its dissimilarity to the other tasks. Furthermore, throughout all our experiments, the tasks "Living being" and "Not living being" consistently exhibited the same membership distribution, which is reasonable considering their equivalence.

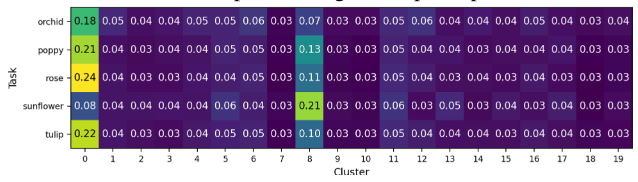
Moreover, when focusing solely on the first 10 tasks from G2 without any additional tasks, our clustering algorithms demonstrated some semantic clustering capabilities, as shown in Figure 4b. The algorithm successfully grouped images of living beings, including {bird, cat, deer, dog, frog, horse}, while another group consisted of images of non-living beings such as {airplane, automobile, ship, truck}. Nevertheless, this might be due to the impact of the "Living being" and "Not living being" tasks; we therefore conducted a similar experiment on G1, generating their data maps and clustering the tasks, without any extra tasks.

As illustrated in Figure 5, even without additional tasks, our method performed the same reasonable clustering for G1, grouping living beings together and non-living things together 5a. Additionally, Figure 5b demonstrates the clustering using three clusters, revealing that the living being cluster was divided into two groups: cluster 1 and cluster 2. Cluster 1 predominantly contained quadruped animals {cat, dog, horse}, while cluster 2 included {bird, frog, deer} that represented the other living creatures except for the deer. These results showcase the effectiveness of our clustering algorithm in capturing semantic similarities among tasks based on the visual data leading to meaningful task groupings.

In addition to our experiments on G1 and G2, we conducted a comprehensive evaluation of an unprecedented number of tasks, specifically 100 tasks from CIFAR100, in G3. As part of this evaluation, we compared our task clustering results against the predefined superclasses provided by CIFAR100. It is important to note that the superclasses in CIFAR100 primarily rely on semantic relationships as illustrated in



(a) Membership clustering of People superclass



(b) Membership clustering of Flowers superclass

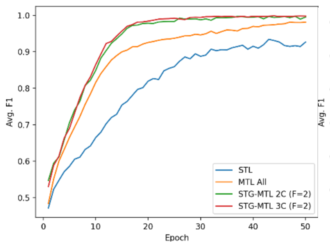
Figure 6. Task grouping of G3 (100 Tasks of CIFAR100) into 20 clusters with $F = 2$

Section 4.1. That is why we focus on coherent superclasses like people and flowers, as examples.

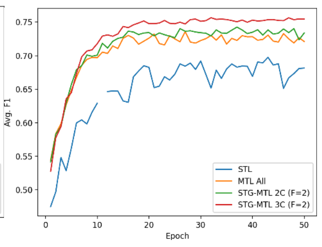
In Figure 6, we showcase an example of the clustering results for a group of super tasks. It is noteworthy that our method successfully clusters certain groups of tasks in alignment with the predefined CIFAR100 superclasses, as illustrated in Figure 6a. However, it is important to acknowledge that there are cases where the clustering may not be perfect, as depicted in Figure 6b; we think this is primarily because our method focus one visual similarities, which is exploited during training rather than semantics. Nevertheless, even in such instances, our clustering algorithm manages to allocate significant weights of all tasks into distinctive clusters, such as clusters 0 and 8 in Figure 6b. Notably, in cluster 8, the participation percentages of the tasks {orchid, poppy, rose, tulip} are the 2^{nd} highest across all clusters, indicating a close relationship with the missclassified task sunflower, yet our method suggests that the other four tasks are more visually related. We further discuss all the clustering result details of the 100 Tasks in Appendix A.

4.4. Evaluation Analysis

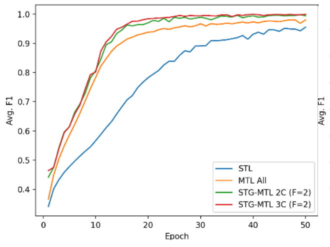
To further validate the effectiveness of our method, we conducted a comprehensive evaluation as described in Section 3.4 on all task groups. Figure 7 presents the average F1 score for both the training and test sets of all the three sets of tasks. Our method is denoted by `STG-MTL xxC (F=2)` where `xx` represents the number of clusters. The MTL curve represents the results obtained from training an MTL model on all tasks without any grouping, while the STL curve represents the results obtained by training separate STL models for each task and merging their outputs. We compare the performance of our method against the MTL and STL approaches in both G1 & G2 and against the MTL approach only in G3 because the STL performance is poorer than the MTL, as it overfits.



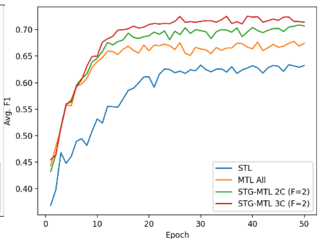
(a) Training Set Results (G1)



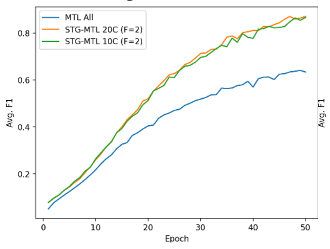
(b) Test Set Results (G1)



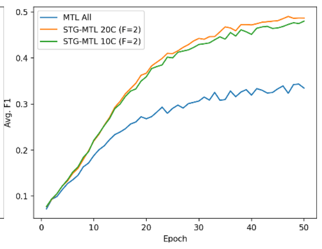
(c) Training Set Results (G2)



(d) Test Set Results (G2)



(e) Training Set Results (G3)



(f) Test Set Results (G3)

Figure 7. Average F1 Scores of all the Groups on both the training and test sets

Overall, our method consistently outperforms both the MTL and STL approaches, indicating that the task grouping provides valuable information for improving task performance. Notably, although our method tends to overfit and achieves excellent training performance, it also achieves the best performance on the test set. This suggests that if the models were further fine-tuned, even greater gains could be achieved, yet we refrain from tuning any of the models in this study to guarantee fairness in comparison.

5. Conclusion and Future Work

In conclusion, we have presented STG-MTL, which is a novel scalable approach for task grouping in multi-task learning (MTL) settings. Our method utilizes data maps (Swayamdipta et al., 2020) to identify task similarities and group them accordingly. We showed is superior scalability theoretically in comparison to TAG (Fifty et al., 2021), HOA (Standley et al., 2020), and MTG-Net (Song et al., 2022). We have also demonstrated the effectiveness of our method through our experiments on CIFAR10 and CIFAR100 datasets (Krizhevsky et al., 2009), where we pushed the boundaries by experimenting with **100 tasks**, which has never been done before in the literature proving its scalability. We have also compared our clustering results against the predefined superclasses in CIFAR100, further validating the effectiveness of our approach. Furthermore, our method outperformed traditional MTL and single-task learning (STL) approaches, showcasing the quality of task grouping and its ability to improve multi-task learning performance.

For future work, we plan to expand the scope of our experiments by including a wider range of datasets and task types, enabling a more comprehensive evaluation of our approach's effectiveness and applicability. Furthermore, as our Data Maps are currently limited to classification tasks, we aim to explore their generalization to other task types, such as regression. Additionally, we hope our research could open a new research direction in the MTL community to explore the development of new features that can capture the training dynamics efficiently, other than data maps. By advancing this research direction, we can unlock new possibilities for enhancing performance and driving further advancements in the field of MTL.

Acknowledgments We gratefully thank the Fatima Fellowship[1] for supporting this research especially during its early stages.

References

Aribandi, V., Tay, Y., Schuster, T., Rao, J., Zheng, H. S., Mehta, S. V., Zhuang, H., Tran, V. Q., Bahri, D., Ni, J., et al. Ext5: Towards extreme multi-task scaling for transfer learning. *arXiv preprint arXiv:2111.10952*, 2021.

Bao, G., Chen, H., Liu, T., Gong, G., Yin, Y., Wang, L., and Wang, X. Covid-mtl: Multitask learning with shift3d and random-weighted loss for covid-19 diagnosis and severity assessment. *Pattern Recognition*, 124:108499, 2022.

Bezdek, J. C., Ehrlich, R., and Full, W. Fcm: The fuzzy c-means clustering algorithm. *Computers & geosciences*, 10(2-3):191–203, 1984.

Bickel, S., Bogojeska, J., Lengauer, T., and Scheffer, T. Multi-task learning for hiv therapy screening. In *Proceedings of the 25th international conference on Machine learning*, pp. 56–63, 2008.

Caruana, R. Multitask learning. *Machine learning*, 28: 41–75, 1997.

Chen, Z., Shen, Y., Ding, M., Chen, Z., Zhao, H., Learned-Miller, E. G., and Gan, C. Mod-squad: Designing mixtures of experts as modular multi-task learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11828–11837, 2023.

[1]`https://www.fatimafellowship.com/`

Crawshaw, M. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

Fan, J., Zhao, T., Kuang, Z., Zheng, Y., Zhang, J., Yu, J., and Peng, J. Hd-mtl: Hierarchical deep multi-task learning for large-scale visual recognition. *IEEE transactions on image processing*, 26(4):1923–1938, 2017.

Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huang, H., Ye, D., Shen, L., and Liu, W. Curriculum-based asymmetric multi-task reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022a.

Huang, S., Li, X., Cheng, Z.-Q., Zhang, Z., and Hauptmann, A. Gnas: A greedy neural architecture search method for multi-attribute learning. In *Proceedings of the 26th ACM international conference on Multimedia*, pp. 2049–2057, 2018.

Huang, Z., Rao, M., Raju, A., Zhang, Z., Bui, B., and Lee, C. Mtl-slt: multi-task learning for spoken language tasks. In *Proceedings of the 4th Workshop on NLP for Conversational AI*, pp. 120–130, 2022b.

Islam, M., Vibashan, V., Lim, C. M., and Ren, H. St-mtl: Spatio-temporal multitask learning model to predict scanpath while tracking instruments in robotic surgery. *Medical Image Analysis*, 67:101837, 2021.

Jin, N., Wu, J., Ma, X., Yan, K., and Mo, Y. Multi-task learning model based on multi-scale cnn and lstm for sentiment classification. *IEEE Access*, 8:77060–77072, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lee, G., Yang, E., and Hwang, S. Asymmetric multi-task learning based on task relatedness and loss. In *International conference on machine learning*, pp. 230–238. PMLR, 2016.

Lee, H. B., Yang, E., and Hwang, S. J. Deep asymmetric multi-task feature learning. In *International Conference on Machine Learning*, pp. 2956–2964. PMLR, 2018.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K.

Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 19–34, 2018.

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

Long, M., Cao, Z., Wang, J., and Yu, P. S. Learning multiple tasks with multilinear relationship networks. *Advances in neural information processing systems*, 30, 2017.

Maninis, K.-K., Radosavovic, I., and Kokkinos, I. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1851–1860, 2019.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3994–4003, 2016.

Peng, Y., Chen, Q., and Lu, Z. An empirical study of multi-task learning on bert for biomedical text mining. *arXiv preprint arXiv:2005.02799*, 2020.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4822–4829, 2019.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Song, X., Zheng, S., Cao, W., Yu, J., and Bian, J. Efficient and effective multi-task grouping via meta learning on task combinations. In *Advances in Neural Information Processing Systems*, 2022.

Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., and Savarese, S. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pp. 9120–9132. PMLR, 2020.

Strezoski, G., van Noord, N., and Worring, M. Learning task relatedness in multi-task learning for images in context. In *Proceedings of the 2019 on international conference on multimedia retrieval*, pp. 78–86, 2019.

Sun, X., Panda, R., Feris, R., and Saenko, K. Adashare: Learning what to share for efficient deep multi-task learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 8728–8740. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/634841a6831464b64c072c8510c7f35c-Paper.pdf.

Swayamdipta, S., Schwartz, R., Lourie, N., Wang, Y., Hajishirzi, H., Smith, N. A., and Choi, Y. Dataset cartography: Mapping and diagnosing datasets with training dynamics. *arXiv preprint arXiv:2009.10795*, 2020.

Vandenhende, S., Brabandere, B. D., and Gool, L. V. Branched multi-task networks: Deciding what layers to share. *CoRR*, abs/1904.02920, 2019. URL http://arxiv.org/abs/1904.02920.

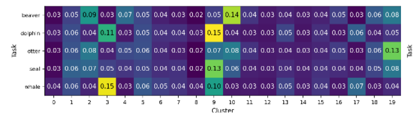
Wu, S., Zhang, H. R., and Ré, C. Understanding and improving information transfer in multi-task learning. *arXiv preprint arXiv:2005.00944*, 2020.

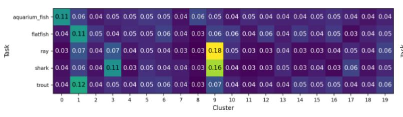
Zhang, L., Liu, X., and Guan, H. A tree-structured multi-task model recommender. In *First Conference on Automated Machine Learning (Main Track)*, 2022a. URL https://openreview.net/forum?id=BEl4CgaHLlc.

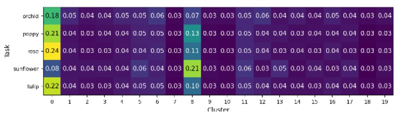
Zhang, Y. and Yang, Q. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2022. doi: 10.1109/TKDE.2021.3070203.

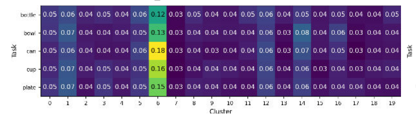
Zhang, Z., Wu, B., and Schuller, B. Attention-augmented end-to-end multi-task learning for emotion prediction from speech. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6705–6709. IEEE, 2019.

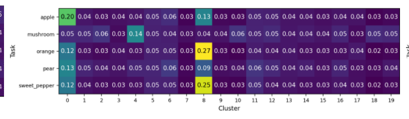
Zhang, Z., Yu, W., Yu, M., Guo, Z., and Jiang, M. A survey of multi-task learning in natural language processing: Regarding task relatedness and training methods. *arXiv preprint arXiv:2204.03508*, 2022b.

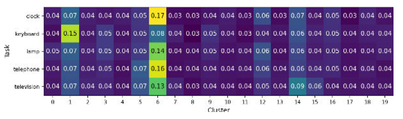

(a) aquatic mammals

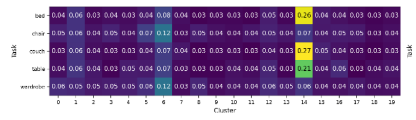

(b) fish

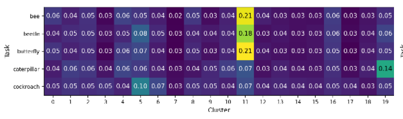

(c) flowers

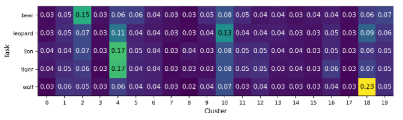

(d) food containers

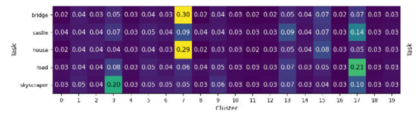

(e) fruit and vegetables

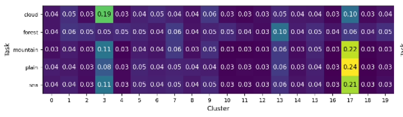

(f) household electrical devices

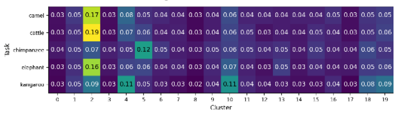

(g) household furniture

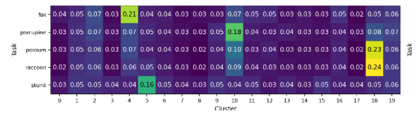

(h) insects

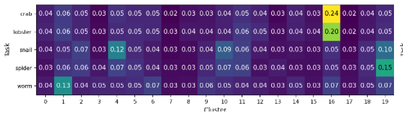

(i) large carnivores

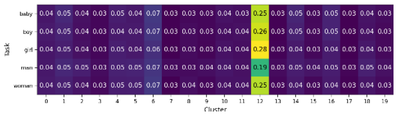

(j) large man-made outdoor things

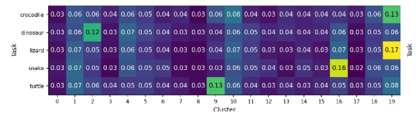

(k) large natural outdoor scenes

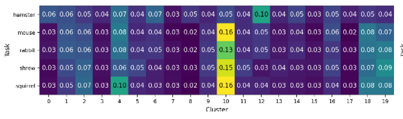

(l) large omnivores and herbivores

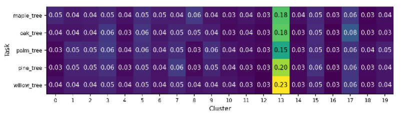

(m) medium-sized mammals

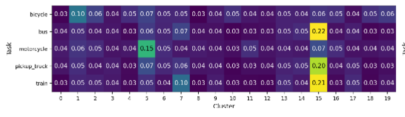

(n) non-insect invertebrates


(o) people


(p) reptiles


(q) small mammals


(r) trees


(s) vehicles 1

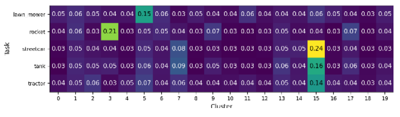

(t) vehicles 2

Figure 8. Task grouping of G3 (100 Tasks of CIFAR100) into 20 clusters with $F = 2$

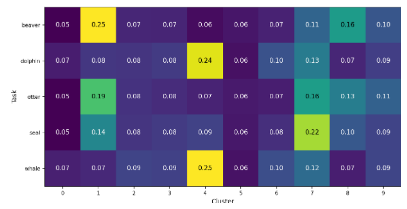
A. Task Clustering Results of 100 Tasks

In this appendix, we present detailed insights into the task clustering results of 100 tasks, building upon the experimental setup outlined in Section 4.3. Figure 8 showcases the clustering results using 20 clusters, while Figure 9 illustrates the results with 10 clusters. Each image in the figures represents the clustering outcomes for one superclass from CIFAR100.

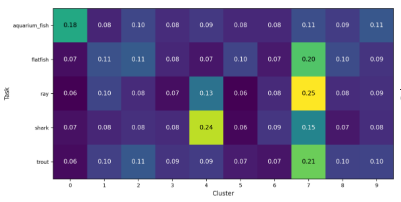
Notably, the clustering with 20 clusters demonstrates successful grouping in many categories, such as {People, Trees, Food Containers, Flowers, Household Electrical Devices}. Figures 8s and 8t highlight the close association between Vehicles 1 and 2, as they are almost merged into the same cluster (Cluster 15). When reducing the number of clusters to 10, we observe enhanced coherence in the assigned tasks. For instance, the tasks related to "Fruit and Vegetables" are nearly clustered together after reducing the number of clusters.

Moreover, our method effectively captures the logical associations between categories. Figures 9f and 9g showcase the plausible merge between "Household Electrical Devices" and "Household Furniture." Additionally, we observe similar distribution patterns among related categories, such as { Large Carnivores, Large Omnivores and Herbivores, Medium-Sized Mammals} in Figures 9i, 9l, and 9m respectively.

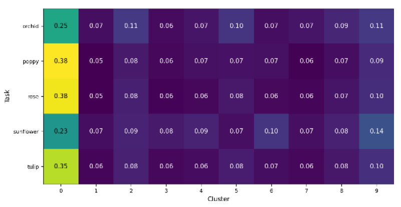
Overall, these results reveal the qualitative success of our approach in clustering a very large number of tasks, highlighting the effectiveness of our method even in such challenging scenarios.



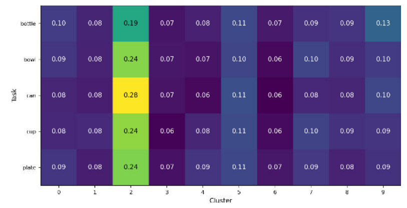
(a) aquatic mammals



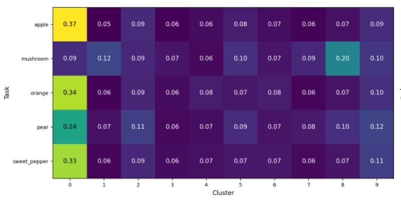
(b) fish



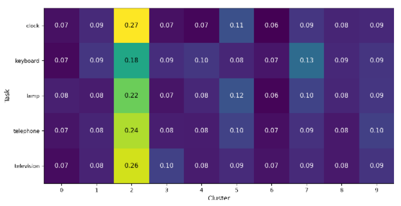
(c) flowers



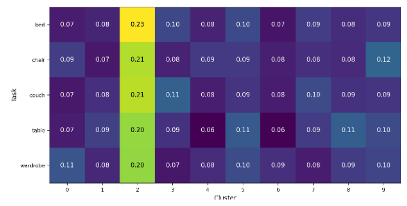
(d) food containers



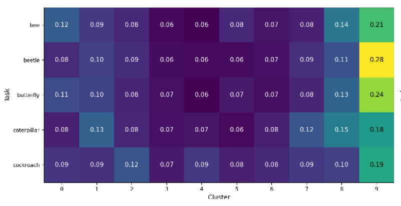
(e) fruit and vegetables



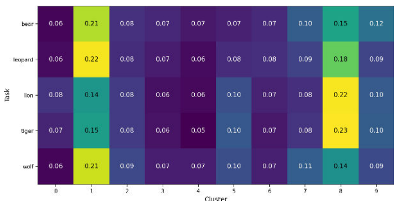
(f) household electrical devices



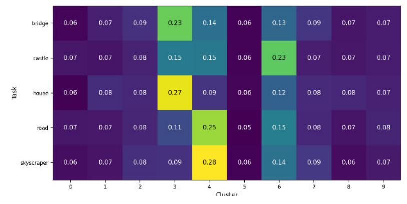
(g) household furniture



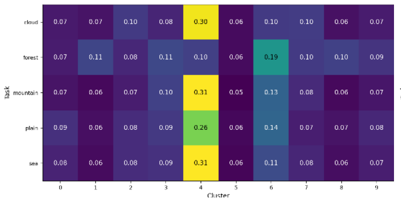
(h) insects



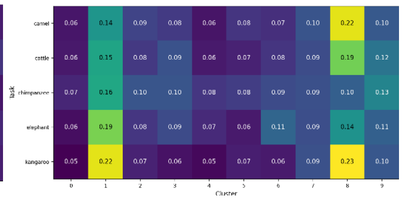
(i) large carnivores



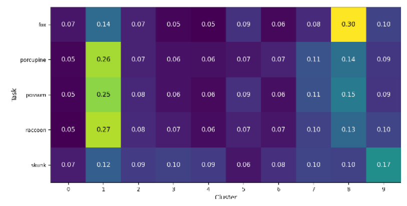
(j) large man-made outdoor things



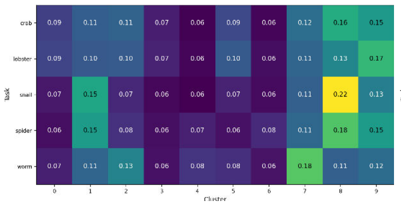
(k) large natural outdoor scenes



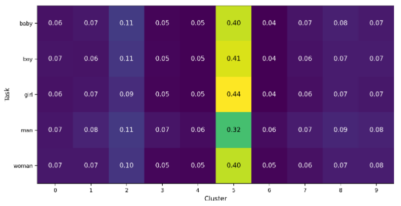
(l) large omnivores and herbivores



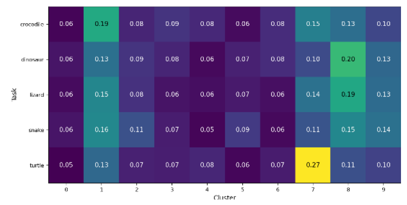
(m) medium-sized mammals



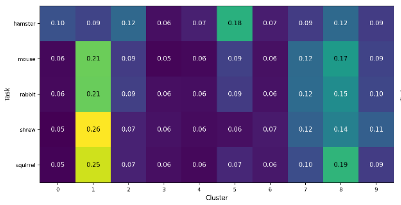
(n) non-insect invertebrates



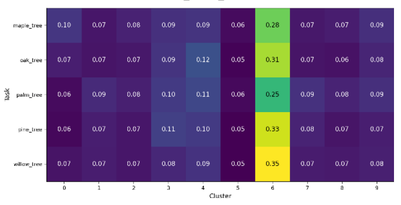
(o) people



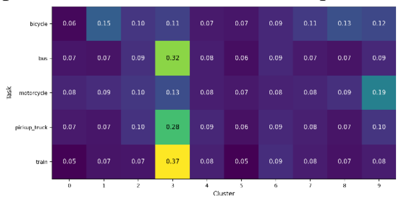
(p) reptiles



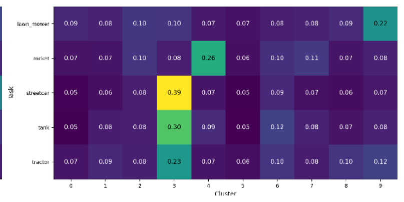
(q) small mammals



(r) trees



(s) vehicles 1



(t) vehicles 2

Figure 9. Task grouping of G3 (100 Tasks of CIFAR100) into 10 clusters with $F = 2$