COMP6258 REPRODUCIBILITY CHALLENGE: MODEL LEGO: CREATING MODELS LIKE DISASSEMBLING AND ASSEMBLING BUILDING BLOCKS

Casper Dancy **Annika Catulli** **Ajayesh Saini**

Github Repository: https://github.com/COMP6258-Reproducibility-Challenge/Reproducibility-Challenge-LEGO

1 INTRODUCTION

The paper proposes Model Disassembling and Assembling (MDA) to isolate task-specific components of deep neural networks. These components can be combined to create new models that perform well in both isolated tasks, removing the need to train a new model on the specific combination of tasks from scratch. Additionally, they claim MDA can be applied to decision route analysis, model compression and knowledge distillation. Although the experiments are performed mostly on CNNs, the authors claim the results apply to deep neural networks in general.

In this report, we question the claim that models can be split into task aware components. Although there is sufficient evidence to show the component allocation technique isolates components that work well for a task, the authors do not prove that this performance is task-specific. We test this using our novel mismatch experiment.

We also claim that the parameter scaling strategy is not sufficient. In the experiments of the original paper, many of the assembled models fail to reach base accuracy even after fine-tuning. This suggests the parameter scaling strategy fails to eliminate bias in the assembled model. To test this, we implement an experiment using unbalanced datasets.

2 EXPERIMENTAL SETUP

We trained all the required models using the code provided. However, we added several features to enhance reproducibility, including a command line argument to control the random seed and a feature that allows saving trained models with custom names for better organization. We trained each model using three different seed values for 200 epochs to ensure consistent and robust evaluation.

In accordance with the paper, 1% of the most confident predictions were selected as representative samples for each class. This corresponded to 50 samples per class for CIFAR-10, and 5 samples per class for both CIFAR-100 and TinyImageNet, given their larger class counts.

To generate class-specific masks used in model disassembly, the selected samples are passed through the network to collect intermediate activations. This is the most resource-intensive part of the experiment and can consume up to 50 GB of RAM in the case of ResNet50. To mitigate this, we modified the code to allow the batch size to be adjusted, as it was originally hardcoded to 256.

The disassembly step uses the generated masks to prune the model. It relies on a dependency graph to ensure structural consistency while removing unnecessary channels, both from intermediate layers and the final classification layer.

To ensure the assembled model functions properly, we apply a parameter scaling step, which adjusts the magnitude of parameters in the final layer. This process compensates for the difference in feature magnitudes across disassembled models from different source domains. While the original paper emphasizes the importance of balancing feature magnitudes—especially in the final fully connected layer—it also describes a theoretical formulation for scaling across layers. However, in practice, the authors' code only applies this normalization to the last layer, and we followed the same approach. Our implementation calculates confidence scores for correctly predicted samples in each class and derives a scaling factor to rescale the weights and biases of the last layer accordingly.

The assembly process was performed using the original code without modification.

When implementing, we encountered numerous difficulties with GoogLeNet and ResNet-50. The given disassembling code did not work on either, with all disassembled models being over-pruned. They would give near-zero accuracies for target tasks. Following Mingbao et al. (2021), we attempted to prune ResNet-50 by pruning only the first two layers of the convolutional network. However, this did not fix the issue. We tried various other methods which did not work. We believe this to be due to the skip connections in ResNet-50 and due to the Inception blocks in GoogLeNet.

3 EXPERIMENTS

3.1 MODEL DISASSEMBLING

Dataset	Disa. Task	Base (%)	Disa. (%)	Orig. Paper Base (%)	Orig. Paper Disa. (%)
CIFAR-10	0	94.33	100.00 (+5.67)	94.40	100.00 (+5.60)
	1	97.01	100.00 (+2.99)	96.50	100.00 (+3.50)
	0-2	94.01	91.56 (-2.45)	93.87	95.47 (+1.60)
	3-9	92.60	91.70 (-0.90)	92.49	92.27 (-0.22)
	3	85.17	100.00 (+14.83)	–	–
	4-8	93.61	93.80 (+0.19)	–	–
CIFAR-100	0	87.00	100.00 (+13.00)	84.00	100.00 (+16.00)
	1	92.00	100.00 (+8.00)	87.00	100.00 (+13.00)
	0-19	70.53	79.06 (+8.53)	71.05	82.50 (+11.45)
	20-69	70.17	77.95 (+7.78)	72.74	79.66 (+6.92)
	15-44	70.20	77.79 (+7.59)	–	–
Tiny-ImageNet	0	–	–	82.00	100.00 (+18.00)
	1	–	–	70.00	100.00 (+30.00)
	0-69	59.36	41.04 (-18.32)	50.17	55.49 (+5.32)
	70-179	55.61	51.20 (-4.41)	45.36	47.95 (+2.59)

Table 1: Disassembly results for models on different datasets, including VGG-16 results from original paper

As is evident in Table 1, for CIFAR-10 and CIFAR-100, the accuracy often only approaches the base accuracy of the model, though it follows similar patterns in gains. Tiny-ImageNet always provided a decrease in accuracy, rather than an increase from the original paper, which may be due to cherry-picked models or the difference in batch-size. Our results include additional tasks to test for selection bias. The task of dissembling took $\sim$ 2 mins for the CIFAR-100 dataset using an A100 GPU.

3.2 MODEL ASSEMBLING

Table 2 demonstrates our experiments were unable to replicate the same base accuracy as the original paper on average, with the outlier of CIFAR-10 + Tiny-imageNet with assembled task 0-2 + 0-69. On average, in VGG16, the original paper had a loss of 7.99% in accuracy when comparing base and assembled models. Our tests provided an average accuracy loss of 8.83%. The difference between our results and the original paper's results is likely due to either the higher batch size used by the original paper, or due to further testing of weight initialization of the base models. The task of assembling itself took $\sim$ 30 seconds for the cifar-100 dataset on an A100 GPU, though this does not include the time taken to disassemble both models for reassembly.

3.3 MISMATCH EXPERIMENT

As mentioned, there is little evidence that the "task-aware" components are not just pruned models that perform well on multiple tasks. We test this by calculating the accuracy of a single disassembled model across multiple tasks of roughly the same size. This results in two statistics, the accuracy on the target task and the average accuracy on other tasks. To implement this, we edited the code to test on the entire dataset then extract accuracies for specific tasks. This verifies the claim that the models

Dataset	Assembled Task	VGG-16		Original Paper	
		Base. (%)	Asse. (%)	Base. (%)	Asse. (%)
CIFAR-10 + CIFAR-100	0-2 + 0-19	73.63	70.51	74.03	74.17
	3-9 + 20-69	72.91	68.97	75.16	73.72
	1-5 + 70-99	73.69	70.92	–	–
	4-8 + 15-44	73.54	71.44	–	–
CIFAR-10 + Tiny-ImageNet	0-2 + 0-69	59.64	48.03	51.97	53.20
	3-9 + 70-179	57.81	53.68	–	–
	1-5 + 44-111	59.21	54.33	–	–
	4-8 + 50-199	59.39	65.55	–	–
CIFAR-100 + Tiny-ImageNet	0-19 + 0-69	61.84	35.09	69.86	50.66
	20-69 + 70-179	60.16	40.44	71.48	50.08
	70-99 + 44-111	61.08	44.07	–	–
	15-44 + 150-199	63.11	47.07	–	–

Table 2: Assembly results for VGG-16

achieve accuracy on single class tasks while the original testing procedure made the component appear to have 100% accuracy on every single subtask.

Dataset	Disa. Task	Average Task (%)	Original Task (%)
CIFAR-10	0	0.00	100.00
	3-9	1.47	91.73
	4-8	0.00	93.56
	1-5	0.00	88.06
CIFAR-100	0	0.00	100
	1	0.00	100
	0-19	32.75	75.75
	70-99	62.18	75.29
	15-44	55.78	72.27

Table 3: Results for disassembled models on non-target tasks

The CIFAR-10 results (shown in Table 3) support the claims of the paper, suggesting that this strategy works well with small numbers of classes. However, the claim is weaker for CIFAR-100. Although the accuracy for the disassembled components are higher, the models perform well on other tasks. While this experiment shows that component allocation does favour the target class, why the component is able to generalise to other tasks requires further research. This may be because the component allocation method does not optimally locate task-specific features when the task range is large. An explanation for this is that, as the number of classes per task increases, more abstract features are extracted that are features of classes outside the task range. Further work may scrutinize the paper's component allocation technique by testing it against existing pruning techniques. One such a method is a pruning strategy using Shapley values Chabrier et al. (2024).

3.4 Unbalanced Disassembly Samples Experiment

In the original paper, the model disassembly process takes 50 samples that give the highest confidence from each class and uses those samples to identify relevant features for disassembly. In this experiment, we change the number of samples for each class to be a random number between 50 and 150 to analyse whether using an unbalanced sample set causes performance differences in the disassembled model. The results from Table 4 are from ResNet-50.

From the results, we can see that single task accuracy stayed the same between the original. This is expected as the samples were never reduced below the original amount of 50 per class, so for

Dataset	Disa. Task	Original Samples (%)	Unbalanced Samples (%)
CIFAR-10	0	100.00	100.00
	0-2	94.01	92.56
	3-9	92.60	89.90
	6	100.00	100.00
	1-5	91.09	89.29
CIFAR-100	0	87.00	100.00
	0-19	79.06	77.12
	20-69	77.95	73.95
	60-65	76.17	76.00
	79	100.00	100.00

Table 4: Disassembly results for models on unbalanced samples

single class tasks, any task that was already reaching the 100% accuracy threshold before would also reach that threshold with a higher amount of samples. In multi-class tasks, there is a decrease in accuracy. This likely shows the unbalance in samples causes a bias towards over-represented classes, something that would be detrimental in many applications. This can be further seen in the per task accuracy of 3-9, where the accuracy of class 3 was 96% compared to the balanced dataset gave 91% for class 3, and class 3 had the highest number of samples in the unbalanced dataset at 200. Compare this to class 4, which had 50 samples, and there was a performance drop from 95% to 85%. Further work would summarise these imbalances, aiming to infer a mathematical relationship between balance of the dataset and bias.

4 Reflection

The method introduced in this paper allows models to be efficiently adapted to subsets and combinations of existing tasks. Since training larger models has the advantage of increased expressivity, this method is likely to create better models for problems which focus on a subset of a large dataset. Disassembled models may be suitable for low-power devices which only need to perform certain tasks, or for more environmentally aware models. While assembled models outperform random baselines, they fail to match the accuracy of base models, making full retraining on a combined dataset a better approach, especially if models aren't already pre-trained. On the other hand, our mismatch experiment shows that the disassembling strategy surpasses basic pruning by generating components that genuinely perform better on specific tasks. We have shown contribution allocation is reproducible for VGG-16 and could be developed to find feature-specific subsets in a model. However, owing to the strong ability of some disassembled models to generalize to other tasks, it is first necessary to compare the component allocation technique with other attribution methods to verify whether these components are optimally disassembled. This is a key limitation of the paper. Additionally, we have shown that the scaling strategy is insufficient for applications where classes may be imbalanced.

The assembly method could be expanded to assemble disassembled models which use different datasets but attempt the same tasks (e.g. classify cat vs dog) to further improve classification performance. Model assembly could be expanded to use several datasets rather than 2.

In conclusion, the paper is largely reproducible. Although there are some issues with pruning certain models, the key claims of the paper are largely justified.

References

Lin Mingbao, Ji Rongrong, Li Shaojie, Wang Yan, Wu Yongjian, Huang Feiyue, and Ye Qixiang. Non-parametric adaptive network pruning. 01 2021. doi: 10.48550/arXiv.2101.07985.

Lisa Chabrier, Anton Crombach, Sergio Peignier, and Christophe Rigotti. Effective pruning for top-k feature search on the basis of shap values. *IEEE Access*, 12:163079–163092, 2024. doi: 10.1109/ACCESS.2024.3489958.