A EXPERIMENT SETTINGS

This section provides a detailed dataset description, baseline description, and training details.

Dataset details. Following Task Arithmetic (Ilharco et al., 2023), Ties-Merging (Yadav et al., 2023), we study multi-task model merging on eight image classification datasets below.

- **SUN397** (Xiao et al., 2016) is a scene classification dataset, which contains images in 397 classes, with a total of 108,754 images, and each class has at least 100 images.
- **Stanford Cars (Cars)** (Krause et al., 2013) is a car classification dataset, which contains 196 classes of cars and a total of 16,185 images. Each class in the training set and test set is divided at a ratio of 1:1.
- **RESISC45** (Cheng et al., 2017) is a remote sensing image scene classification data set. It contains 45 classes of scenes and a total of 31,500 images, of which there are approximately 700 images in each class.
- **EuroSAT** (Helber et al., 2019) is a satellite image classification dataset containing 27,000 labeled and geo-referenced images in 10 classes.
- **SVHN** (Yuval, 2011) is a real-world digital classification data set extracted from house numbers in Google Street View images. There are 10 classes in total. The training set contains 73,257 samples, the test set contains 26,032 samples, and 531,131 additional simple samples can be used as additional training data.
- **GTSRB** (Stallkamp et al., 2011) is a traffic sign classification dataset, which contains 43 classes of traffic signs with a total sample size of more than 50,000.
- **MNIST** (LeCun, 1998) is a benchmark dataset for image classification. It contains grayscale images of handwritten digits in 10 classes. The number of images in the training and test sets is 60,000 and 10,000 respectively. The number of images in each class is balanced.
- **DTD** (Cimpoi et al., 2014) is a texture classification data set, which contains 47 classes, a total of 5,640 images, and each class has approximately 120 images.

Baseline details. Our experiments involve the following seven comparison methods and four variations of our method.

- **Individual** means that each task uses an independent fine-tuned model, which has no interference between tasks, but cannot perform multiple tasks simultaneously.
- **Traditional MTL** collects the original training data of all tasks together to train a multi-task model. It can be used as a reference *upper bound* for model merging work.
- **Weight Averaging** is the simplest method of model merging, which directly averages the parameters of multiple models. It can be used as a *lower bound* for model merging.
- **Fisher Merging** (Matena & Raffel, 2022) calculates the Fisher information matrix (Fisher, 1922) to measure the importance of each parameter when merging models, and model merging is performed according to the guidance of this importance.
- **RegMean** (Jin et al., 2023) imposes a constraint when merging models, that is, the L_2 distance between the merged model and a single model is required to be as small as possible.
- **Task Arithmetic** (Ilharco et al., 2023) first defines the concept of “task vectors” and merges task vectors into a pre-trained model to execute multi-task learning.
- **Ties-Merging** (Yadav et al., 2023) further solves the task conflict problem in Task Arithmetic (Ilharco et al., 2023). It eliminates redundant parameters and resolves symbol conflicts through three steps: Trim, Elect Sign, and Disjoint Merge.
- **Task-wise AdaMerging (Ours)** is based on Task Arithmetic (Ilharco et al., 2023), which uses an unsupervised method to automatically learn the merging coefficient of the task vector in Task Arithmetic.
- **Task-wise AdaMergign++ (Ours)** is based on Ties-Merging (Yadav et al., 2023), which uses an unsupervised approach to learn a merging coefficient for each task vector in Ties-Merging.
- **Layer-wise AdaMerging (Ours)** automatically learns a merging coefficient for each layer of each task vector in Task Arithmetic (Ilharco et al., 2023).
- **Layer-wise AdaMergign++ (Ours)** uses an unsupervised approach to learn a merging coefficient for each layer of each task vector in Ties-Merging (Yadav et al., 2023).

Implementation details. For the seven baseline methods, we follow the experimental settings in Task Arithmetic (Ilharco et al., 2023) and Ties-Merging (Yadav et al., 2023). In our experiments, the merging coefficient λ of Task Arithmetic and Ties-Merging is set to 0.3 by default. For our four variants, we initialize all coefficients $\{\lambda_k\}_{k=1}^{K}$ (or $\{\lambda_k^l\}_{k=1,l=1}^{K,L}$) to 0.3 by default before learning and then update them unsupervised. We use an Adam optimizer (Kingma & Ba, 2014) to update the merging coefficients, with the learning rate set to 0.001, the momentum to (0.9, 0.999), and the batch size to 16. Pre-trained models ViT-B/32, ViT-B/16 and ViT-L/14 from CLIP (Radford et al., 2021).

B EXPERIMENT RESULTS

B.1 PERFORMANCE, GENERALIZATION, ROBUSTNESS

Performance. Tab. 5 shows the average accuracy of merging ViT-B/16 on eight tasks. We can observe that: (i) Ties-Merging alleviates the conflict problem of task vectors in Task Arithmetic, thus achieving a 3.2% performance improvement compared to Task Arithmetic. (ii) Our Task-wise AdaMerging and AdaMerging++ automatically learn a merging coefficient for each task vector in Task Arithmetic and Ties-Merging, thus bringing about 2.2% and 1.0% performance improvements, respectively. (iii) Our Layer-wise AdaMerging and AdaMerging++ further adaptively learn a merging coefficient for each layer of each task vector in Task Arithmetic and Ties-Merging, ultimately achieving performance improvements of 11.1% and 8.7%. These results further demonstrate the effectiveness of our AdaMerging scheme in multi-task model merging.

Table 5: Multi-task performance when merging ViT-B/16 models on eight tasks.

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	**Avg Acc**
Task Arithmetic (Ilharco et al., 2023)	61.1	65.9	74.0	76.2	88.0	73.9	98.4	53.0	73.8
Ties-Merging (Yadav et al., 2023)	69.1	72.5	80.5	84.0	85.0	71.5	98.1	54.9	77.0
Task-wise AdaMerging (Ours)	64.4	64.2	75.4	86.7	86.3	86.7	97.6	46.9	76.0
Task-wise AdaMerging++ (Ours)	67.8	70.2	79.9	89.2	87.5	79.2	98.3	51.9	78.0
Layer-wise AdaMerging (Ours)	70.2	80.7	81.6	94.8	91.6	95.8	98.5	66.2	84.9
Layer-wise AdaMerging++ (Ours)	71.8	80.8	84.1	94.3	91.9	94.5	98.7	69.8	85.7

Generalization. As shown in Tab. 6, we demonstrate the generalization of Layer-wise AdaMerging under the ViT-B/16 architecture. In the two unseen test tasks of EuroSAT and MNIST (their corresponding task vectors are not merged), our AdaMerging improved the average accuracy by 2.3% compared to Task Arithmetic. On two unseen tasks, RESISC45 and SVHN, the average accuracy increased by 1.1%. This shows that AdaMerging has better generalization properties.

Table 6: Generalization results on two unseen tasks when merging ViT-B/32 models on six tasks.

	Seen Tasks							Unseen Tasks		
Method	SUN397	Cars	RESISC45	DTD	SVHN	GTSRB	**Avg Acc**	EuroSAT	MNIST	**Avg Acc**
Task Arithmetic	68.1	73.0	81.6	59.1	89.1	83.8	75.8	43.9	87.5	65.7
AdaMerging (Ours)	69.1	79.3	90.0	66.2	95.2	94.4	82.4	45.9	90.1	68.0
Method	SUN397	Cars	GTSRB	EuroSAT	DTD	MNIST	Average	RESISC45	SVHN	Average
Task Arithmetic	69.0	73.8	81.1	87.6	58.2	98.4	78.0	56.0	67.7	61.8
AdaMerging (Ours)	72.9	81.0	97.1	96.4	66.5	99.2	85.5	52.3	75.6	63.9

Robustness. Tab. 7 shows the robustness test of AdaMerging and Task Arithmetic based on ViT-B/16 on seven corruption test datasets. Fig. 6 shows an example of corruption. We can observe that in the test datasets Motion Blur, Impulse Noise, Gaussian Noise, Pixelate, Spatter, Contrast and JPEG Compression where the distribution drifts, the average accuracy of AdaMerging is 9.9%, 8.2%, 7.8%, 6.8%, 12.4%, 9.5% and 9.7% higher than that of Task Arithmetic, respectively. This shows that our AdaMerging is more robust to test data distribution shifts than Task Arithmetic.

B.2 ANALYSIS EXPERIMENT

Merging Coefficients Visualization. Fig. 7 and Fig. 8 show the merging coefficients of eight task vectors learned by Layer-wise AdaMerging and AdaMerging++ on ViT-B/16 respectively. In addition, Fig. 9 and Fig. 10 show the coefficients learned under Vit-L/14. We can clearly observe that in different layers of different task vectors, the learned merging coefficients are different. Finding the merging coefficients of so many layers through grid search is almost impossible.

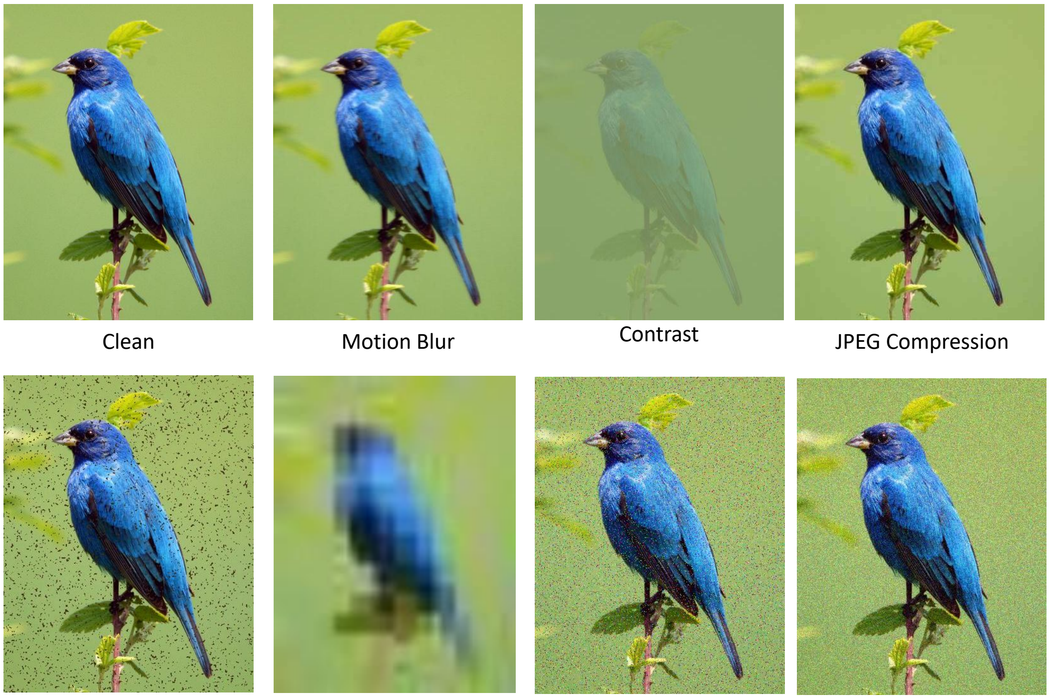
Table 7: Robustness results when merging ViT-B/16 models on four tasks.

	Clean Test Set					Corruption Test Set (Motion Blur)				
Method	Cars	EuroSAT	RESISC45	GTSRB	**Avg Acc**	Cars	EuroSAT	RESISC45	GTSRB	**Avg Acc**
Task Arithmetic	75.3	96.3	85.3	80.5	84.3	73.5	70.9	83.9	72.2	75.1
AdaMerging (Ours)	83.4	97.2	88.6	97.5	91.7	81.3	75.9	87.4	95.6	85.0

	Corruption Test Set (Impulse Noise)					Corruption Test Set (Gaussian Noise)				
Method	Cars	EuroSAT	RESISC45	GTSRB	**Avg Acc**	Cars	EuroSAT	RESISC45	GTSRB	**Avg Acc**
Task Arithmetic	70.4	59.5	75.2	54.0	64.8	72.2	60.8	78.5	51.0	65.6
AdaMerging (Ours)	77.6	42.1	81.9	90.2	73.0	79.1	58.9	81.2	74.5	73.4

	Corruption Test Set (Pixelate)					Corruption Test Set (Spatter)				
Method	Cars	EuroSAT	RESISC45	GTSRB	**Avg Acc**	Cars	EuroSAT	RESISC45	GTSRB	**Avg Acc**
Task Arithmetic	03.8	38.0	24.8	71.3	34.5	72.1	58.4	79.9	60.1	67.6
AdaMerging (Ours)	04.1	46.4	23.6	91.3	41.3	79.3	60.9	85.8	93.7	80.0

	Corruption Test Set (Contrast)					Corruption Test Set (JPEG Compression)				
Method	Cars	EuroSAT	RESISC45	GTSRB	**Avg Acc**	Cars	EuroSAT	RESISC45	GTSRB	**Avg Acc**
Task Arithmetic	73.4	62.5	81.3	76.9	73.5	75.1	73.1	84.8	64.7	74.4
AdaMerging (Ours)	81.4	68.1	85.8	96.8	83.0	81.9	76.0	87.3	91.0	84.1



Clean

Motion Blur

Contrast

JPEG Compression

Spatter

Pixelate

Impulse Noise

Gaussian Noise

Figure 6: An example of corruption data visualization, in which the corruption image generation method refers to Hendrycks & Dietterich (2019).

Visualization of Correlation between Entropy and Loss. As shown in Fig. 11, we analyze the correlation between the entropy and the model's prediction loss for eight tasks (or datasets). As described in Sec. 3.2.2, in each dataset, we sort the entropy on the test samples from small to large into eleven groups and observe the average loss of sample prediction within each group. We observe that groups with smaller entropy generally have smaller average losses. Therefore, it is reasonable to take entropy minimization as an optimization surrogate objective for loss minimization.

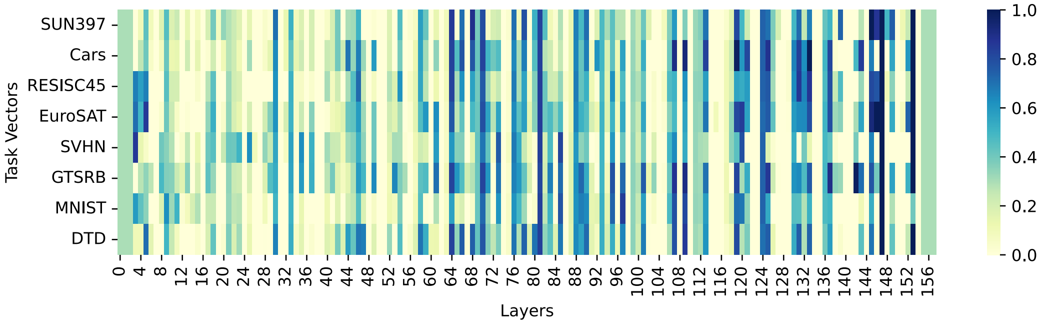


Figure 7: Learned model merging coefficients of **Layer-wise AdaMerging on ViT-B/16**. The k-th row represents the k-th task vector, the l-th column represents the l-th layer, and the intersection point represents the coefficient λ_k^l.

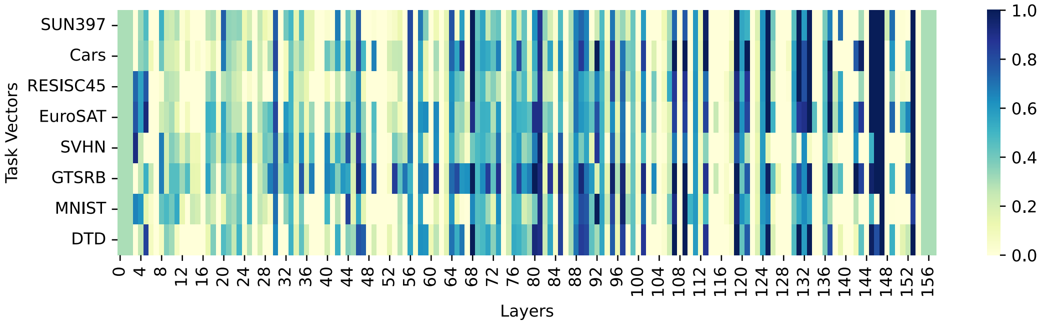


Figure 8: Learned model merging coefficients of **Layer-wise AdaMerging++ on ViT-B/16**. The k-th row represents the k-th task vector, the l-th column represents the l-th layer, and the intersection point represents the coefficient λ_k^l.

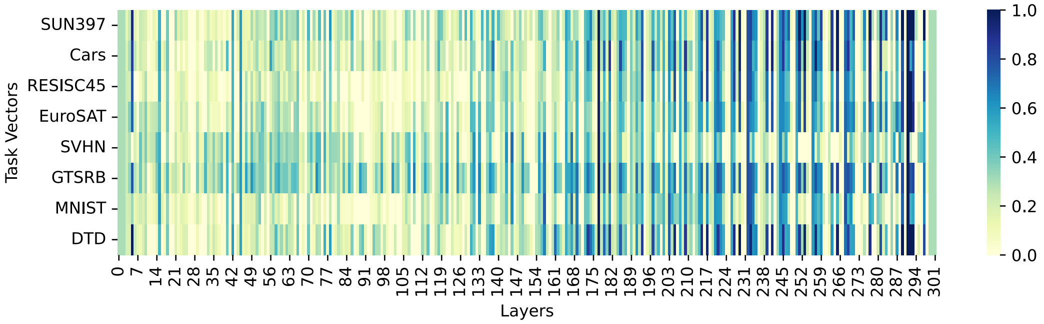


Figure 9: Learned model merging coefficients of **Layer-wise AdaMerging on ViT-L/14**. The k-th row represents the k-th task vector, the l-th column represents the l-th layer, and the intersection point represents the coefficient λ_k^l.

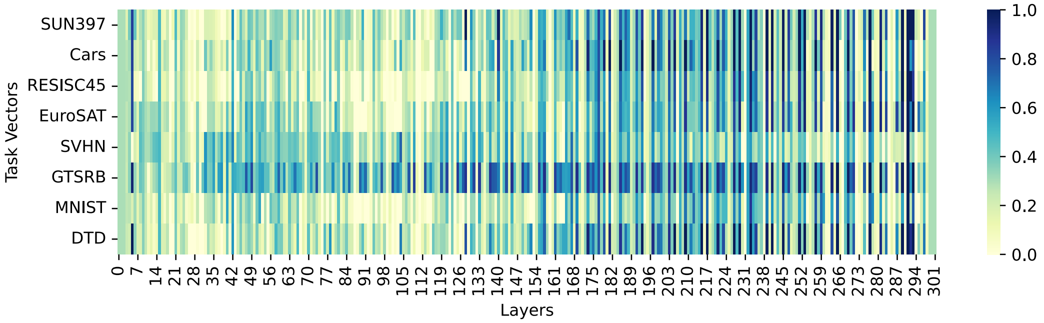


Figure 10: Learned model merging coefficients of **Layer-wise AdaMerging++ on ViT-L/14**. The k-th row represents the k-th task vector, the l-th column represents the l-th layer, and the intersection point represents the coefficient λ_k^l.

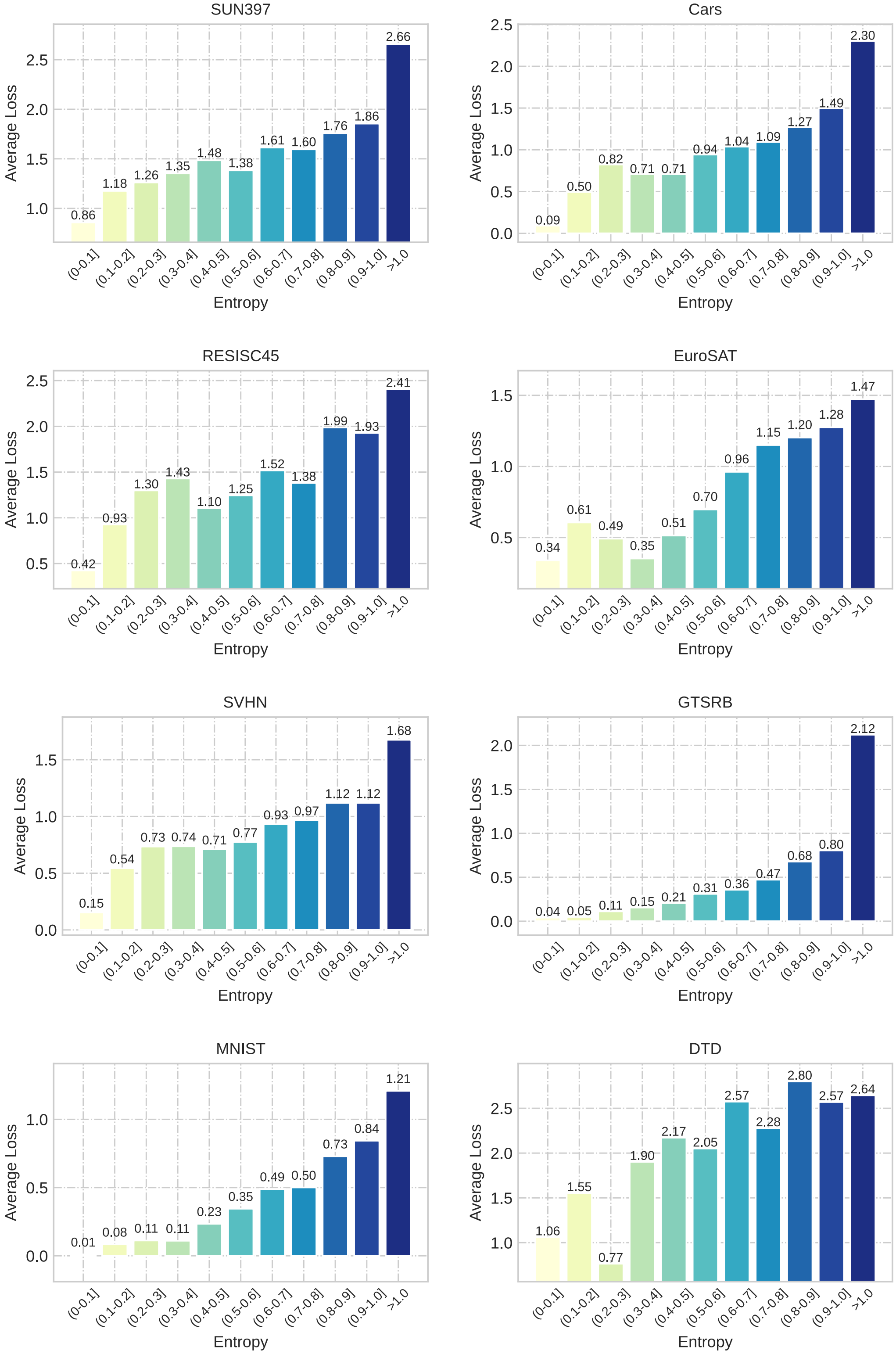


Figure 11: Correlation analysis of *entropy* and *average loss* on eight tasks (or datasets). We can observe that there is a **high positive correlation** between entropy and prediction loss on each dataset.