

## A EXPLANATION OF MODEL EVOLUTION

In this section, we provide a more comprehensive explanation of the principles underlying the Differential Evolution algorithm, furthermore, we offer an in-depth elucidation of the mutation process, providing a visual representation in Figure 8 to enhance clarity and understanding. Overall, the fundamental principle of the differential evolution algorithm involves randomly selecting three distinct individuals, performing a mutation operation to create a new candidate solution, using a crossover operation to refine the solution, and replacing the original solution if the new one performs better. This iterative process continues until certain stopping criteria are met.

To better illustrate our model evolution method, we have created a flowchart as shown in algorithm 1. The details of combining our proposed method with other models are provided in Step 3. The approach involves calculating an overall score by using model merging on the mutated individuals along with the non-mutated individuals in the current evolution. In contrast, the simple evolver determines the success of mutation based on the score of individual entities, while the combined approach assesses it based on the score of the entire population after individual mutation.

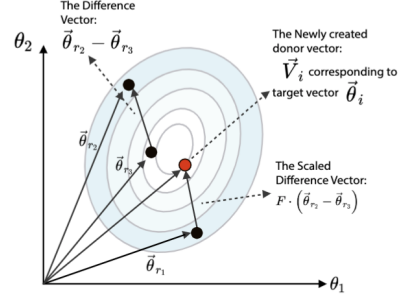


Figure 8: An illustration of the mutation process in difference evolution.

---

### Algorithm 1 Model Evolution

---

```

1: Step 1 - Initializing the Population
2: Initialize population  $\Theta$ 
3:  $generation \leftarrow 0$ 
4:  $converged \leftarrow \text{False}$ 
5: while not converged do
6:   Step 2 - Evolution Process: Mutation and Recombination
7:   for each candidate solution  $\theta_i$  in  $\Theta$  do
8:     Randomly select  $\theta_{r1}$  and  $\theta_{r2}$ 
9:      $F \leftarrow$  Random scaling factor
10:     $Cr \leftarrow$  Random crossover rate
11:    Compute mutated solution  $\theta_i^*$  using  $\theta_i$ ,  $\theta_{r1}$ ,  $\theta_{r2}$ , and  $F$ 
12:    Perform recombination of  $\theta_i^*$  based on  $Cr$  and  $\theta_i$ 
13:   Step 3 - Model Inference
14:   if not combined with other model merging methods then
15:     Evaluate the performance of  $\theta_i^*$  on development data
16:   else
17:     Merging  $\theta_i^*$  with other models
18:     Evaluate the performance of merged model on development data
19:   end if
20:   end for
21:   Step 4 - Updating the Population
22:    $converged \leftarrow \text{True}$ 
23:   for each candidate solution  $\theta_i$  in  $\Theta$  do
24:     if  $\theta_i^*$  outperforms  $\theta_i$  then
25:       Replace  $\theta_i$  with  $\theta_i^*$ 
26:        $converged \leftarrow \text{False}$ 
27:     end if
28:   end for
29:    $generation \leftarrow generation + 1$ 
30: end while

```

▷ A population of candidate solutions  
 ▷ Initialize generation counter  
 ▷ Convergence flag

▷ Select random solutions  
 ▷ Control parameter for mutation  
 ▷ Control parameter for recombination

▷ Assume convergence  
 ▷ Comparing performance  
 ▷ Update population  
 ▷ Reset convergence flag

▷ Increment generation counter

---

## B IMPACT OF DEVELOPMENT DATASET

The availability of development datasets directly impacts the effectiveness of our model evolution approach. However, many publicly available datasets either do not provide development sets or widely use them as test sets. In our case, the development set of the GLUE dataset is used as a test set, so we utilize a small portion of the training dataset (approximately 5%) for model evolution. For non-i.i.d. partition methods, we also use only a subset of the same training data samples. In the case of the unified emotion dataset, we separately extract 10% of data from each of the five high-resource datasets for model evolution, following the same partitioning method as employed in Jin et al. (2022).

For our model evolution approach, the quality of the development dataset can significantly impact performance, making the selection of a high-quality development dataset a crucial consideration. To address this, we conducted experiments on the emotion dataset using different lengths of model evolution methods. We performed experiments with both the simple evolver and regmean evolver, evolving all five domain-specific models. The experimental results are shown in Table 5, indicating that even with a short development dataset, model evolution can still be effective. However, as the length of the development dataset increases, the performance of model evolution tends to improve. Additionally, we included the test scores of simple methods as baselines for comparison.

Length	None	1/4	1/2	1
Evolver	23.18	30.14	32.03	33.27
Regmean.Evolver	38.74	39.43	39.57	39.87

Table 5: The performance of model evolution with different length of development dataset. *None* means evolver is not conduct and the test score of *simple averaging* and *regmean* method is recorded.

## C METRICS, DATASET AND TRAINING DETAILS

### C.1 MERGING MODELS TRAINED ON NON-I.I.D. PARTITIONS.

Merging models initially trained on non-i.i.d. partitions of the same dataset is started, which is achieved by simulating synthetic data splits across the 8 tasks within the GLUE benchmark. Each task involves dividing the training data into two partitions, each containing 1,000 training examples with distinct label distributions. Following this, we perform fine-tuning on these two partitions for 8 pairs of individual models and merge each pair of models. The evaluation of these merged models takes place on the official validation sets, which portray a joint distribution of both partitions.

### C.2 METRICS AND COMPARED METHODS

In evaluating merged models trained for non-i.i.d. partitions of the same dataset, we assessed their performance using a unified test set characterized by a joint distribution of all partitions. For merged models trained across different domains or tasks, we measured their performance across individual domains or tasks incorporated into the merger and derived their macro-average. Similarly, when evaluating out-of-domain performance, we computed the macro-average of their performance across the out-of-domain test set.

The performance of individual models involved in merging are reported: (1) the average performance of all individual models (**Avg.**  $f_{1..N}$ ); (2) the performance of the best *single* individual model (**Best.**  $f_{1..N}$ ), as determined by using the

	Train	Dev	Test
<i>In-domain</i>			
DialyDialog	72,085	10,298	20,596
CrowdFlower	27,818	3,974	7,948
TEC	14,735	2,105	4,211
Tales-Emotion	10,339	1,477	2,955
ISEAR	5,366	766	1,534
<i>Out-of-domain</i>			
Emoint			7,102
SSEC			4,868
ElectoralTweets			4,056
GroundedEmotions			2,585
AffectiveText			1,250

Table 6: Statistics of emotion classification datasets.

validation set; (3) the performance of the individual models corresponding to the training data set for each test set (**Domain-Specific**).

### C.3 EMOTION CLASSIFICATION

In order to investigate the performance of the sentiment classification task, we selected a diverse and challenging set of datasets. Among them, DailyDialogs (Li et al., 2017), CrowdFlower, TEC (Mohammad, 2012), Tales-Emotion (Alm et al., 2005), and ISEAR (Scherer & Wallbott, 1994) is utilized to train domain-specific model. For accessing OOD generalization performance, we use EmoInt (Mohammad & Bravo-Marquez, 2017), SSEC (Schuff et al., 2017), ElectoralTweets (Mohammad et al., 2015), GroundedEmotions (Liu et al., 2017), and AffectiveText (Strapparava & Mihalcea, 2007). For OOD evaluation, we focus exclusively on the fundamental emotions: anger, disgust, fear, joy, sadness, and surprise. A detailed overview of the datasets and statistics is provided in Table 6.

### C.4 GLUE BENCHMARK

In the GLUE dataset experiments, we utilized multiple tasks, including CoLA (Warstadt et al., 2019), SST-2 (Socher et al., 2013), MRPC (Dolan & Brockett, 2005), STS-B (Cer et al., 2017), MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), QQP, and RTE (Giampiccolo et al., 2007). These tasks cover various natural language understanding problems such as text classification, text similarity, and natural language inference. To assess our merged models, we tested them on the official development sets. We performed experiments by training models on non-i.i.d. partitions, creating various partition scenarios through random sampling. Each partition is uniformly sub-sampled to yield a total of 1,000 training examples per partition.

### C.5 TIME COST

Initial Population for Evolving	T5-base	RoBERTa-base	DistilBERT-base	DeBERTa-large	GPT2
All Domain Specific Models on Emotion Datasets	24.5	19.2	18.7	21.3	20.1
Pairwise Models on Emotion Datasets	9.3	7.3	7.1	8.1	7.6
None-iid Pairwise Models on GLUE Benchmark	7.2	5.7	5.5	6.2	5.9
Cross Tasks Pairwise Models on GLUE Benchmark	8.7	7.1	6.8	7.8	7.5

Table 7: Time cost (in the unit of minutes) of RegMean\_Evolver on different experiments with 20 generations. T5-base is tested on single A800 GPU and other models are tested on single A6000 GPU. The time cost is mainly related to the size of model and the length of development dataset when conducting model evolution.

We report the time cost of the scheme of model evolution. T5-base is tested on single NVIDIA A800 80G GPU and other models are tested on single RTX A6000 48G GPU. We find that all task of model evolution can be completed within half an hour, which is very cost-efficient in improving the model performance without further training.

## D INTEGRATION WITH COEFFICIENT SEARCH

Model	Simple (coefficient search)	Evolver (scale factor search)	Fisher (coefficient search)	Fisher_Evolver (scale factor search)	RegMean (coefficient search)	RegMean_Evolver (scale factor search)
RoBERTa-base	37.78 ( <b>38.83</b> )	39.13 ( <b>39.98</b> )	37.11 ( <b>38.96</b> )	40.34 ( <b>41.23</b> )	46.56 ( <b>46.82</b> )	46.89 ( <b>47.03</b> )
DistilBERT-base	36.76 ( <b>37.63</b> )	38.85 ( <b>39.67</b> )	34.52 ( <b>37.54</b> )	40.37 ( <b>41.31</b> )	43.09 ( <b>43.14</b> )	43.22 ( <b>43.31</b> )
T5-base	38.82 ( <b>39.91</b> )	40.21 ( <b>41.11</b> )	38.08 ( <b>39.22</b> )	41.46 ( <b>42.55</b> )	47.35 ( <b>47.84</b> )	47.92 ( <b>48.06</b> )

Table 8: **Coefficient Search Result** when merging pairwise emotion classification models. Simple, Fisher and RegMean are model merging algorithms for comparison. All the results we reported are averages of 10 ( $C_5^2$ ) runs after paring models from a set of 5.

The coefficient search is a promising scheme to improve the model merging performance, by searching the optimal  $\alpha$ . The proposed model evolution can also be integrated with the coefficient search method by searching the optimal scale factor  $f$ . We have performed the grid search of  $\alpha$  and  $f$  with intervals of 0.05 from 0.1 to 0.9. We present the result in Table 8. In the setting of Simple, Fisher and RegMean, the results show that a default version of evolver (with scale factor  $f = 0.5$ ,  $cr = 0.5$ ) outperforms the coefficient search results. Notably, the integration with scale factor search further

boosts the performance of the evolver, which is worth further investigation. Especially, the crossover ratio  $Cr$  in model evolution could also be the subject of coefficient search. Many adaptive schemes are also available in the realm of evolution algorithms, like SADE (Qin & Suganthan, 2005) and SHADE (Tanabe & Fukunaga, 2013), providing a possibility of future algorithmic development.