Heterogeneous Language Model Optimization in Automatic Speech Recognition

Anonymous ACL submission

Abstract

The rising data privacy risks make it difficult for automatic speech recognition (ASR) systems to acquire complete training data in practical application. Recently, the merge paradigm for acoustic model has been proposed to solve the issue. However, ASR still suffers from another salient issue on language model. Current efforts mainly focus on isomorphic neural network models, while language model optimization is characterized by merging and matching heterogeneous models including n-gram and neural network models. In this paper, we propose a novel Match-and-Merge paradigm to fill up the vacuum for the language model optimization. Based on different training datasets, we train multiple language model pairs. In order to merge them into a target pair with the best performance, we first propose a Genetic Match-and-Merge (GMM) method that can be specifically adopted to optimize heterogeneous models. To improve the algorithm efficiency, we further propose a Reinforced Match-and-Merge (RMM) method, which maintains superior recognition accuracy while reducing convergence time. Extensive experiments demonstrate the effectiveness and generalization of our proposed methods, which significantly establishes the new state-of-the-art.

1 Introduction

Automatic speech recognition (ASR) has been widely used in many fields such as communication, education and automobiles. With the development of deep learning, training recognition models requires a huge amount of language and speech data from various domains as support. However, in recent years, it has become increasingly infeasible to train models with complete data due to the increasing data privacy risk which makes distinct curators reluctant to share their own private data.

Compared with hard-to-obtain private data, it seems more realistic and reliable to utilize exist-

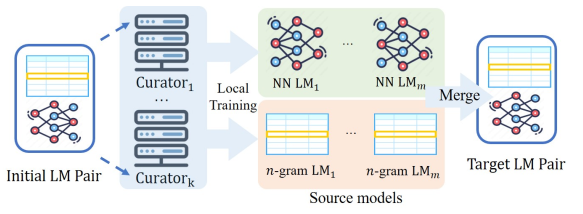

Figure 1: The optimization of heterogeneous language models in ASR. The n-gram LM and the NN LM are heterogeneous, different curators use their own private data to train the initial model pair locally. Multiple pairs of source LMs need to be merged and matched to obtain a better quality target model pair.

ing ASR models at hand that have been trained with different datasets. A feasible strategy is to distribute the initial model to different curators, who possesses their own private data. Each curator uses their own private data to train the model locally and then sends the model back instead of the data. The model parameters of different versions trained by different curators will be merged into parameters of one version as the final training result of the initial model through a merge paradigm. The merged model performs better than any local training model, and can be used as a pre-trained model for future task transfer.

The main challenge of this strategy is how to design a merge paradigm. Averaging all model parameters is a simple and straightforward practice, and has been proved to be effective (McMahan et al., 2017). However, whether there is a better merge paradigm is still worth discussing. (Tan et al., 2020) first proposes a divide-and-merge paradigm and successfully applies it to the acoustic model merge. However, this method is limited to neural network models, and cannot be applied to other machine learning models that are not neural network structures in ASR.

Although end-to-end speech recognition model has been extensively studied in the research community, the real industrial deployment of ASR still contains two distinct components: the acoustic model (AM) and the language model (LM) (Li et al., 2020). For AM, the end-to-end deep neural network (DNN) is the first choice (Graves et al., 2006; Graves, 2012a; Chan et al., 2016), and for LM, n-best list is the most common practice (Si et al., 2012, 2013; Sundermeyer et al., 2015). Different from AM that only uses DNN, LM first uses the n-gram model to identify the n-best list or lattice, and then uses a high-level model like DNN to re-mark these sentences to obtain the recognition result. Note that the n-gram model model is a non-neural network structure and is heterogeneous with the DNN model.

In this paper, we do not solely focus on isomorphic model optimization, but propose an unexplored heterogeneous language model optimization problem in ASR as shown in Figure 1. Different from AM optimization, the main challenges for LM optimization can be summarized as the following aspects: 1) The existing AM optimization methods are only designed for isomorphic model, but the structure of n-gram LM has intra-class different structure from the DNN which is heterogeneous. 2) The language part is a heterogeneous combination of the n-gram LM and the DNN LM, two models require mutual matching to achieve the best performance. 3) Compared with AMs, LMs have larger parameters and are more difficult and time-consuming to train. Therefore, existing approaches for AM optimization could not be applied to LM optimization.

To tackle the above issues, we propose a novel match-and-merge paradigm for LM optimization in ASR. We first propose a genetic match-and-merge method named GMM, which treats n-gram and DNN LMs as two populations merged separately, and then matches the two populations. The merge results caused by the operation of GMM are random. In order to further improve efficiency, then we propose a novel reinforced optimization method named RMM to utilize the current merge result as feedback information to guide the next merge operation. We set up the problem by formalizing the language model optimization into a mathematical problem. The agent observes the performance of target model pairs and select an action to modify the optimization strategy. Information thus feeds back from the validation stage to the training stage. Experiments show that RMM still obtains comparable merged model while reducing much computation overhead.

Contributions in this paper include: 1) We identify a new task of merging heterogeneous language models in ASR. 2) We propose two novel strategies for the matching and merging of heterogeneous LMs with defined formulation. 3) Extensive experiments on public datasets demonstrate that the ASR model transferred by our proposed method can effectively improve the recognition accuracy and time efficiency.

2 Related Work

2.1 Automatic Speech Recognition

Automatic speech recognition is widely used in the industry. Considering the practical implementation, current mainstream framework for industrial product is mainly composed of AM and LM (Li et al., 2020). The AM extracts features from the original speech signal and captures the relationship between feature vectors and phonemes. The LM predicts the word sequence with the highest probability. In addition, a dictionary counts phoneme-to-word mappings to connect AM and LM.

For traditional ASR, the AM is represented by the the Gaussian Mixture Model - Hidden Markov Model (GMM-HMM) (Rabiner, 1989; Gales and Young, 2007). This framework uses GMM to predict the observation probability of the speech state, and then uses HMM to predict the transition probability of the speech state. The LM usually adopts the n-gram LM (Ullmann, 1977; Suen, 1979) based on a large-scale text or audio corpus to count the frequency of adjacent words. Some subsequent efforts further improve the performance of GMM-HMM framework, including speaker adaptation (Gauvain and Lee, 1994; Leggetter and Woodland, 1995) and discriminative training (Bahl et al., 1986; Juang et al., 1997).

However, since the milestone breakthrough of deep learning (Senior et al., 2012; Bengio, 2009), deep neural network (DNN) has swept the speech field and has become the mainstream research direction in both academic and industrial circles. For AM, some works attempt to combine DNN with HMM and replace the original GMM. Typical representatives of this DNN-HMM framework are FFDNN-HMM (Dahl et al., 2011), CNN-HMM(Sainath et al., 2013) and RNN-HMM (Graves et al., 2013). Other works focus on end-to-end speech recognition solutions, such as connectionist temporal classification (CTC) (Graves et al.,

2006) and Transducer (Graves, 2012b). Recently, the application of Attention (Bahdanau et al., 2016; Chiu et al., 2018) and Transformer (Dong et al., 2018; Zhou et al., 2018) in the end-to-end ASR has also achieved obvious improvements. However, to the best of our knowledge, the AM based on the DNN-HMM framework has achieved the state-of-the-art performance at present.

Compared with the AM, the LM has attracted slightly less attention. In addition to the traditional n-gram LM, some DNN based LMs have recently been applied to ASR. Especially, key Transformer-based models, such as GPT (Brown et al., 2020), BERT (Devlin et al., 2018) and XLNet (Yang et al., 2019), have achieved breakthrough improvements compared to previous LMs. ASR product usually uses the n-gram LM for rough selection, and uses the DNN LM for refinement. Although there have been many end-to-end LM researches in academic circles, the hybrid n-best list practice with two distinct components n-gram LM and DNN LM still dominates the industrial area due to its robustness, and flexibility and modularization.

2.2 Language Model Optimization

We note that some efforts (Pickhardt et al., 2014; Liu et al., 2010; Goodman, 2000) focus on language model combination. Though model combination can deal with heterogeneous LMs, it is completely different from this task setting and we should clarify their differences. For model combination, it does not require merging all models into a single one. In inference, all models predict the results separately, and these results are combined to get the final output. Therefore, model combination is more similar to ensemble learning (Zhou, 2012). However, our model merging is to distribute the model first, then merge the model parameters obtained from local training on different datasets, and use the merged parameters as the result of the model fitting on multiple datasets. Model combination methods increase the target model size as the number of source models increases, while model merging methods can maintain the size of target model the same as initial model, which is more suitable for large-scale industrial deployment.

In this paper we study the language model optimization which requires the merged model to be improved iteratively. Existing works for model merging only focus on AM optimization. As for acoustic model, recently Tan *et al.* (Tan et al., 2020) first proposes a paradigm of model division and merging with two novel algorithms. This technique has been successfully applied to the acoustic model, but they are limited to deep neural networks based models. However, as for language model, n-best list is still the best performing system for practical implementation which consists of n-gram LM and DNN LM. The n-gram LM and the NN LM are heterogeneous, and are a non-neural network structure and a neural network structure, respectively. The existing AM optimization methods mainly study end-to-end neural network models. Therefore, these methods cannot be applied to heterogeneous language models containing non-neural network structures. In this paper, we are the first to explore the LM optimization task in ASR. It requires simultaneous optimization of neural network and non-neural network models, and the matching performance of the two heterogeneous models should be better than the performance of source model pairs.

3 Problem Formulation

Suppose that there is a set of n private datasets for different business scenarios, $T = \{T_1, ..., T_n\}$. Each dataset T_i is trained on a pair of LM $M_i = (M_{N,i}, M_{R,i})$, where $M_{N,i}$ represents the ith n-gram LM and $M_{R,i}$ represents the ith DNN LM. These n-gram LMs share the homogenous structure but different parameters, and so are the DNN LMs. The n-gram LM and the DNN LM are heterogeneous with each other. We name these n paired LMs as the source models. In this task, all the source models will be merged into one paired target model $M_T = (M_{N,T}, M_{R,T})$, which has better performance than source models but still shares the same architecture.

The n-gram LM is an n-dimensional matrix, and each element of the matrix is based on the word sequence frequency of the corpus. We denote each column vector of the matrix $M_{N,i}$ as $A^j_{N,i} (1 \leq j \leq n)$. For each DNN LM $M_{R,i}$ we assume it has L DNN layers, and we denote the parameters of lth layer as $W^l_{R,i} (1 \leq l \leq L)$, which contains all types of trainable parameter including weight and bias. For notational simplicity, we utilize the add operation on model $M_{N,i}$ and $M_{R,i}$ to represent the merge operation on model parameters.

For evaluation setting, we require some samples to validate the the performance of M_T, and we name these samples as the validation dataset. Al-

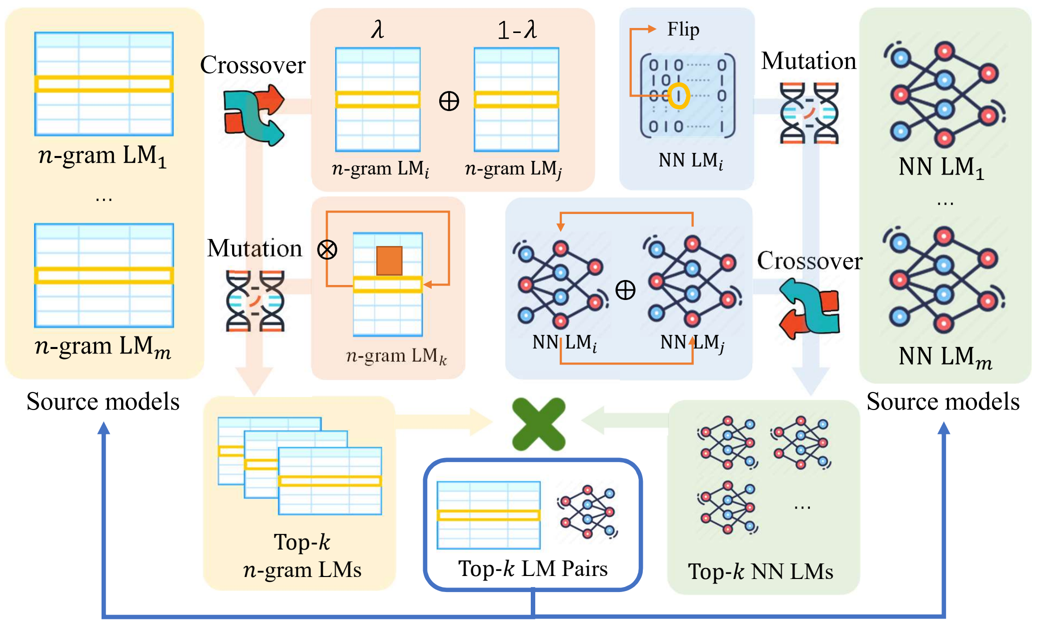

Figure 2: Overview of Genetic Match-and-Merge. The heterogeneous language models are regarded as different populations, and each population executes genetic operators separately to produce the next generation population. The individuals of the two populations are matched one by one, and the combination with the highest matching fitness is selected and reproduced to the next generation.

though extra validation dataset is essential, only very few data can significantly improve the performance of the model, which will be discussed in the following section. It implies that the LM can achieve comparable results with fewer training samples. For fair comparison, we also need some extra datasets in the unknown business scenarios to test M_T, and we name those data as the test dataset. Besides, private datasets T is referred as the training dataset. We adopt the widely used metric Character Error Rate (CER) to assess the merging quality of LMs. It measures the the levenshtein distance between the ground truth and our recognition hypothesis according to the number of characters in ground truth.

4 Genetic Match-and-Merge

In order to achieve the evolution of models, a straightforward idea is to apply the genetic algorithm (Sampson, 1976).

We introduce a Genetic Match-and-Merge (GMM) algorithm, which is calibrated for the matching and merging paradigm of two different heterogeneous models n-gram LM and DNN LM.

As shown in Figure 2, we can regard the source models as the initial population. The operators of generating offspring includes three classical practices in the genetic algorithm: reproduction, mutation and crossover. The main difficulty lies in adapting these three operators to the two heterogeneous LMs. We design the mutation and the crossover operators as follows.

For DNN LM, we follow some common practices (Gupta and Wadhwa, 2014; McMahan et al., 2017).

- Mutation. It modifies the binary file of the selected model and reverses one bit at random.
- Crossover. It selects two adjacent models and randomly taking the lth $(1 \leq l \leq L)$ layer as the exchange point. The first l layers of one model and the back $L-1$ layer of another model will be combined to generate a new model. For example, for $M_{R,i}$ and $M_{R,i+1}$ we can obtain two generated offsprings:

$$M_{R,i'} = \{W^1_{R,i'}, ..., W^l_{R,i'}, W^{l+1}_{R,i+1'}, ..., W^L_{R,i+1'}\}$$
$$M_{R,i+1'} = \{W^1_{R,i+1'}, ..., W^l_{R,i+1'}, W^{l+1}_{R,i'}, ..., W^L_{R,i'}\}$$

For n-gram LM, the optimization of $M_N = \{M_{N,1} ... M_{N,n}\}$ can be considered as combining all the elements of two parent matrices in a weighted way to generate a new one.

- Mutation. It randomly selects one column of the selected model and adds the elements of this column by a coefficient k, where k randomly samples from $(0, 1)$, i.e.,

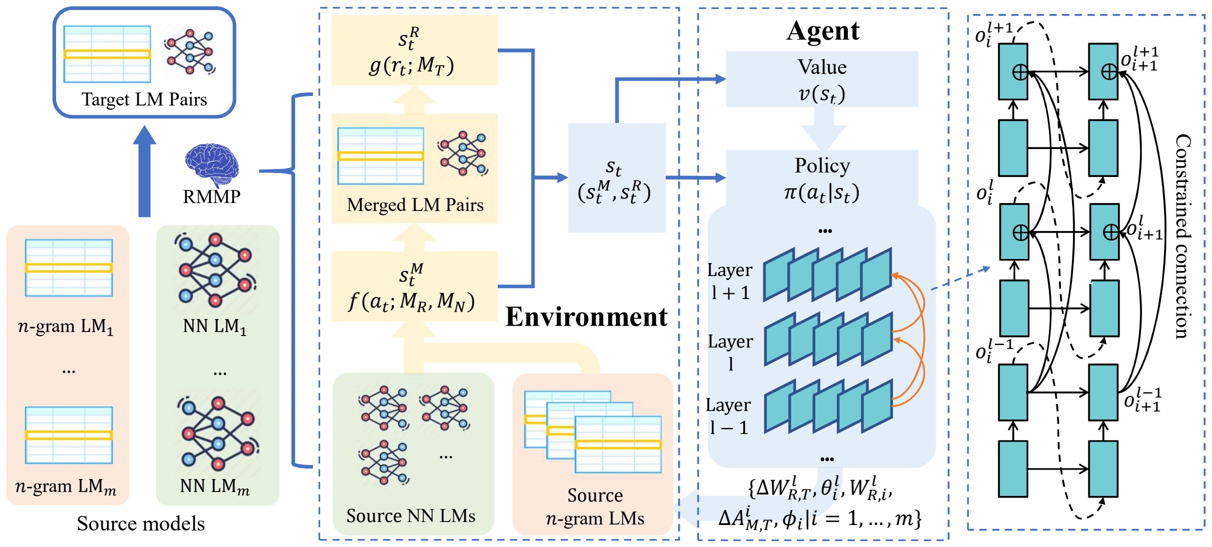

Figure 3: Overview of Reinforced Match-and-Merge. With a mathematical formulation, the merging problem can be converted into a variable selection problem. Then a recurrent network predicts each variable for the LM with a dynamic constraint. The reinforced architecture minimizes the CER of merged models on the validation dataset.

$$M_{N,i'} = [0^1, ..., 0^{j-1}, k*A^j_{N,i}, 0^{j+1}, ..., 0^n] + M_{N,i}$$

- Crossover. It crosses two adjacent models in pairs and sums them weighted to generate a new model, i.e.,

$$M_{N,i'} = \lambda M_{N,i} + (1-\lambda) M_{N,i+1}$$

where λ is a trade-off weight sampled from $(0,1)$ at random.

For reproduction, in addition to copying the models to the next generation, it also needs to match heterogeneous models. We can regard the n-gram LM and DNN LM as two different populations, and define the fitness as the degree of matching between the two populations. The reproduction determines the parents of next generation for each population according to this new fitness. It first matches the best K n-gram LMs with the best K DNN LMs one by one, and then measures the fitness of each paired heterogeneous models. Finally it copies the K groups of models with the highest fitness to the next generation. To assess the fitness of the population, we adopt CER as the metric and the lower the CER, the better the fitness. Note that these K groups may have the same model, and as the selection number K increases, the diversity of the population increases. Although expanding the number of population can enlarge the search space and obtain better results, it also increases the computational overhead. Therefore, we evaluate the CER of each pair of heterogeneous LMs on the validation dataset for each generation. The top K pairs with the lowest CERs will be chosen as the the parents of the next generation.

5 Reinforced Match-and-Merge

Although GMM can generate high quality LMs, it still takes much training time to get a accurate solution. This is because GMM is purposeless random crossover and mutation leading to the slow convergence speed of the algorithm. Therefore, the genetic algorithm is still some distance from the real world application. To solve the above bottleneck, we propose a Reinforced Match-and-Merge (RMM) paradigm which adopts a reinforcement learning agent (Mnih et al., 2013). It utilizes the feedback of each evolution to guide the following optimization, which can produce comparable results as GMM but more efficient.

The major challenge of this idea is to convert the merging problem for LM into a mathematical formulation that the agent can task actions. In this work, we use the rigorous proposition from (Tan et al., 2020) to present the following mathematical formulation of DNN LM:

$$\begin{aligned} W^l_{R,T} &= \sum_{i=1}^{n} \theta^l_i W^l_{R,i} + \Delta W^l_{R,T} \\ \text{s.t.} \quad \theta^l_i &> 0, \sum_{i=1}^{n} \theta^l_i = 1 \end{aligned} \tag{1}$$

where $(1 \leq l \leq L)$ and the extra variable $\Delta W^l_{R,T}$ denotes the changes caused by the mutation operator. The summation term $\sum_{i=1}^{n} \theta_i^l W_{R,i}^l$ stimulates the crossover operator.

From Equation 1 we can conclude that the offspring generated by the operators of the genetic algorithm follows the above pattern. Therefore, similar to the above proposition, we can observe that the merge of n-gram LM can be presented by the following formula:

$$\begin{aligned} M_{N,T} &= \sum_{i=1}^{n} \phi_i M_{N,i} + \sum_{j=1}^{n} \Delta A_{M,T}^{j} \\ \text{s.t.} \quad & \phi_i > 0, \sum_{i=1}^{n} \phi_i = 1 \end{aligned} \tag{2}$$

where $(1 \leq l \leq L)$ and the extra variable $A_{M,T}^{j}$ denotes the changes caused by the mutation operator. The summation term $\sum_{i=1}^{n} \phi_i M_{N,i}$ stimulates the crossover operator.

Now, we have already formulate the pattern of n-gram LM and DNN LM. The target model is to follow this pattern and the overall optimization problem can be defined as:

$$\begin{aligned} & min_{\Delta W_{R,T}^l, \theta_i^l, W_{R,i}^l, \Delta A_{M,T}^j, \phi_i} \ell(M_{N,T}, M_{R,T}) \\ \text{s.t.} \quad & W_{R,T}^l = \sum_{i=1}^{n} \theta_i^l W_{R,i}^l + \Delta W_{R,T}^l \\ & M_{N,T} = \sum_{i=1}^{n} \phi_i M_{N,i} + \sum_{j=1}^{n} \Delta A_{M,T}^{j} \\ & \theta_i^l > 0, \phi_i > 0, \sum_{i=1}^{n} \theta_i^l = 1, \sum_{i=1}^{n} \phi_i = 1 \end{aligned} \tag{3}$$

where $M_{N,T}$ and $M_{R,T}$ are parameters of the target n-gram LM and DNN LM, respectively. $\ell(M_{N,T}, M_{R,T})$ is the loss function of paired models assessed on validation sets.

Next we describe a novel reinforced paradigm for match-and-merge upgrade. The framework is illustrated in Figure 3. Typically, the set of source models can be viewed as an environment. According to the above mathematical formulation, we can define the action set $\mathcal{A}$ as a sequence of tokens $[a_1, ..., a_t]$ that decide what variables to participate in the merge function, and thus update the environment state. The state s_t is made up of two components s_t^M and s_t^R. s_t^M is the output model pairs of the merge function $f(a_t; M_R, M_N)$ guided by actions a_t. s_t^R is the measurement output produced by a non-differentiable evaluation operation that executed on the validation dataset. After taking actions at each timestep t, the agent receives a reward signal r_t from the environment. The agent tends to predict more favorable evolution strategy with the positive reward.

The agent is an actor-critic structure (Konda and Tsitsiklis, 2000) containing a policy part and a value part. The policy part implements a recurrent neural network (Zaremba et al., 2014) sering as a actor. It provides a policy $\pi(a_t|s_t)$ that represents the probability of selecting each action under the state s_t. Different from common architecture, the output sequence is subject to the Equation 1 and 2. Therefore, we design a constrained connection to convert the static structure into a dynamic one. At timestep t, a coefficient v is added to the network. It is the sum of previous predictions at timestep $1, ..., t-1$. It controls the activation function ∂ to limit its output under the constraint of previous layers. Each ∂ produces the current state and the next state as follows:

$$o_t = \partial(v^T \tanh(W_p s_{t-1} + U_c o_{t-1})) \tag{4}$$

where s_t represents the hidden state of the agent at timestep t, o_t represents the output state at timestep t, and the matrices W_p, U_c and v are trainable parameters. We sample from the output results to determine the current action, and use it for the input and control variables of the next layer. Note that this constraint exists only between homogeneous models, but not between heterogeneous models.

The value part serves as the critic which approximates the expected value under the state s_t. Here the agent receives the value of $1 - CER$ as the reward signal r_t on the validation dataset at convergence. The training process will not be finished until the reward exceeds the threshold. Since the value part is a non-differentiable function, we can define the following policy gradient strategy based on the REINFORCE rule (Williams, 1992):

$$\nabla_{\omega_a} J = [\nabla_{\omega_a} \log \pi(a_t|s_t; \omega_a)] E_{P(a_{1:t};\omega_a)}[r_t], \tag{5}$$

where ω_a represents the parameters of the agent, and $J(\omega_a)$ denotes the expected reward $E_{P(a_{1:T};\omega_a)}[r_t]$ for the proposed permutation and combination of actions. Once the agent predicts the actions for merging optimization, the merged models are built and matched. The parameters ω_a are optimized according to the Equation 5 to minimize the expected CER of the validation dataset.

6 Experiment

6.1 Experimental Setup

To ensure that our method described in this paper is reproducible, we select public available datasets to conduct all experiments. Specifically, we choose seven speech datasets from the OpenSLR [1] website, which are SLR18, SLR33, SLR38, SLR47, SLR62, SLR68, SLR93. Each dataset can be regarded as the private data owned by each curator, and does not participate in the model training of other curators. Some datasets have been divided into training sets, validation sets and test sets in advance, and we conduct experiments according to the existing divisions. For some datasets without splitting, we randomly split them into training, validation and test datasets with proportions 60%:20%:20%. The CER of the merged model on the held-out validation dataset will be recorded.

For fair comparison, we use SLR33, SLR38, SLR47, SLR62 and SLR93 to train five pairs of n-gram LMs and DNN LMs, respectively. These five paired models can be viewed as the source models (M_N, M_R). SLR18 and SLR68 are used as the unknown dataset to evaluate the generalization performance of the merged model. Each pair of LMs is trained by the same training criterion for the DNN model named maximum mutual information (Bahl et al., 1986) with the same initialization. Due to the low efficiency of GMM, we collect the validation sets of the five datasets and randomly sample 10% of the data for evaluation.

We use the open source toolkit Kaldi (Povey et al., 2011) to establish the ASR system. Its built-in "Chain" model can be utilized as the acoustic model which consists of the pre-trained HMM and DNN. Here we adopt the Time Delay Neural Network (TDNN) (Waibel et al., 1989) as the structure of the acoustic neural network. For LM, we choose the RNN (Mikolov et al., 2010) as the DNN LM and the tri-gram model as the backoff n-gram LM. The n-gram LM is trained by the open-source toolkit SRILM (Stolcke, 2002). All experiments are conducted on Intel Xeon CPU of 72 cores, NVIDIA Tesla K80 GPU and 314GB memory.

To the best of our knowledge, we are the first to study the merging of heterogeneous models and there is no related methods reporting results. So we design the following baseline methods and compare our model with them: (a) Fine-tuning. We use the pre-trained source n-gram LM and DNN LM and fine-tune them on the validation dataset. (b) Direct Average. We directly average the parameters of all source models to obtain the target model. (c) GMM. (d) RMM. All baseline methods are assessed on the validation dataset to optimize the target paired models by default.

[1]http://www.openslr.org/resources.php

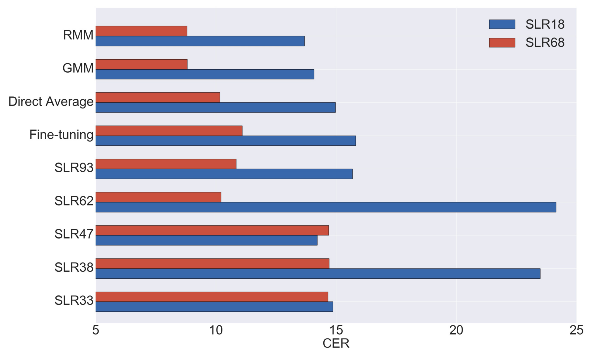

Figure 4: CER on the test sets of SLR18 and SLR68. Lower values are better.

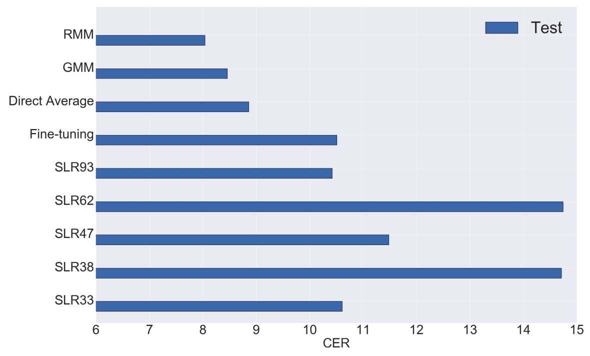

Figure 5: Average CER on test sets of SLR33, SLR38, SLR47, SLR62 and SLR93. Lower values are better.

6.2 Effectiveness Evaluation

We compare the performance of direct average, GMM, RMM and fine-tuning. All these methods use the same source models and we ensure that all models have been trained to convergence. In detail, GMM runs for 100 generations, and each population in each generation selects the 15 best individuals for matching.

We reports the CERs of all the models, including the source models. All results are evaluated on the test set of each dataset. Figure 5 presents the performance of all models on test datasets SLR33, SLR38, SLR47, SLR62 and SLR93. We can easily conclude that the four baseline methods achieve better results than the source models. This is because the source model only fits better on their

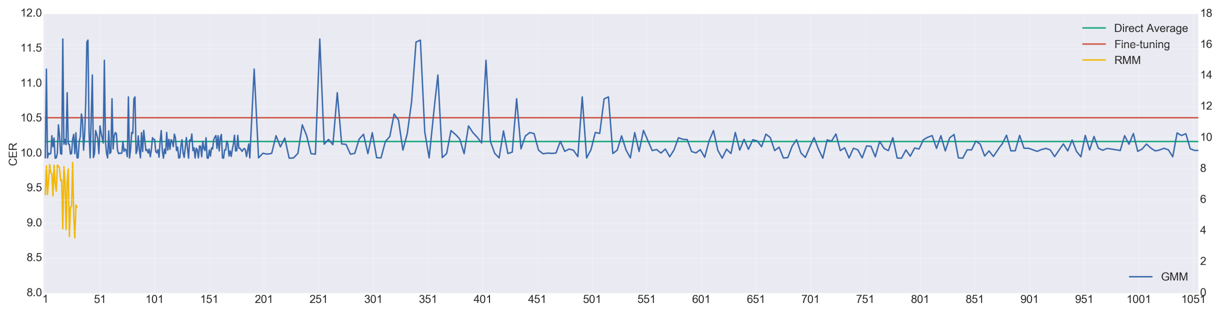

Figure 6: Convergence curves of GMM and RMM. The secondary axis corresponds to GMM, and the primary axis corresponds to the remaining methods.

respective datasets, but cannot be well generalized to other domains. The performances of different source models also vary a lot due to the distinct size and quality of the training dataset. The differences between the source models increase the challenge of model merging. Among them, RMM obviously surpasses all other optimization methods and proves to be the best method.

In order to further study the model generalization, we assess all models on other test datasets of SLR18 and SLR68, respectively, and the results are reported in Figure 4. Although fine-tuning can slightly improve the quality of the target model and surpasses most source models, it is still worse than source model SLR62 and SLR93. The SLR62 and SLR93 datasets contains more speech records than other ones, and the data distribution of the curator's private dataset is more general. Other source models reports the highest CER on both two datasets. The direct average method outperforms the source models and fine-tuning, but it is obviously not as good as the results achieved by the GMM and RMM. GMM and RMM report the lowest CER on both two datasets. Between them, RMM achieves sightly lower CER than GMM and sets up a new state-of-the-art.

6.3 Efficiency Evaluation

We compare the efficiency between GMM and RMM, and the CERs of the best-so-far merged models over iterations are summarized in Figure 6. We define the iteration as one model passes through the validation dataset, and GMM should take more than 1000 iterations to evaluate each pair of models on the validation dataset. By contrast, RMM only needs to run for 30 iterations. We can observe that RMM converges more quickly than GMM. The CER of RMM drops significantly after just one iteration, and the network finally converges to the optimal in less than 30 iterations. On the contrary, GMM performs optimization in a very slow process. In the first 60 iterations, the performance of its top-K models did not exceed the direct average models and the transfer learning models. Although it can generate a better pair of models than the above two methods after 660 iterations, it still falls behind RMM. Under the same computing resources and experimental setting, GMM takes 15 days to merge models, while RMM only takes 2 days to complete model merging. This suggests the benefits of RMM, which utilizes the CER of the merged models as a reward to guide the network to search for a better merging strategy. GMM lacks information support, which makes the search efficiency much low. Though both two methods can generate comparable results in terms of the merged model quality, RMM substantially performs more accurately and efficiently than GMM, and is more practical to apply to the large-scale data processing.

7 Conclusion

In this paper, we explore the optimization of heterogeneous language models in ASR and propose a novel Match-and-Merge paradigm. We train multiple pairs of n-gram LM and DNN LM on different datasets as source models. In order to merge them into the target paired models which have better quality, we propose two novel algorithms: GMM and RMM. GMM is based on the genetic evolution, and RMM utilizes the strategy of reinforcement learning with a novel mathematical formulation. Experiments demonstrate that both two algorithms can significantly improve the performance of language model with a few-sample validation dataset. Furthermore, RMM that employs the feedback information to guide the search shows superior efficiency and can be applied to the real world with large-scale data.

References

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949. IEEE.

Lalit Bahl, Peter Brown, Peter De Souza, and Robert Mercer. 1986. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 49–52. IEEE.

Yoshua Bengio. 2009. Learning deep architectures for AI. *Found. Trends Mach. Learn.*, 2(1):1–127.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. 2018. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. 2011. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE.

Mark J. F. Gales and Steve J. Young. 2007. The application of hidden markov models in speech recognition. *Found. Trends Signal Process.*, 1(3):195–304.

J-L Gauvain and Chin-Hui Lee. 1994. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE transactions on speech and audio processing*, 2(2):291–298.

Joshua T Goodman. 2000. Putting it all together: language model combination. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, volume 3, pages 1647–1650. IEEE.

Alex Graves. 2012a. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Alex Graves. 2012b. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee.

Hitesh Gupta and Deepinder Singh Wadhwa. 2014. Speech feature extraction and recognition using genetic algorithm. *International Journal of Emerging Technology and Advanced Engineering*, 4(1):363–369.

Biing-Hwang Juang, Wu Hou, and Chin-Hui Lee. 1997. Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio processing*, 5(3):257–265.

Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014.

Christopher J Leggetter and Philip C Woodland. 1995. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer speech & language*, 9(2):171–185.

Jinyu Li, Rui Zhao, Eric Sun, Jeremy HM Wong, Amit Das, Zhong Meng, and Yifan Gong. 2020. High-accuracy and low-latency speech recognition with two-head contextual layer trajectory lstm model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7699–7703. IEEE.

Xunying Liu, Mark JF Gales, Jim L Hieronymus, and Philip C Woodland. 2010. Language model combination and adaptation usingweighted finite state transducers. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5390–5393. IEEE.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Rene Pickhardt, Thomas Gottron, Martin Körner, Paul Georg Wagner, Till Speicher, and Steffen Staab. 2014. A generalized language model as the combination of skipped n-grams and modified kneser-ney smoothing. *arXiv preprint arXiv:1404.3377*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, CONF. IEEE Signal Processing Society.

Lawrence R Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. 2013. Deep convolutional neural networks for lvcsr. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8614–8618. IEEE.

Jeffrey R Sampson. 1976. Adaptation in natural and artificial systems (john h. holland).

A Senior, V Vanhoucke, P Nguyen, T Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*.

Yujing Si, Ta Li, Shang Cai, Jielin Pan, and Yonghong Yan. 2012. Recurrent neural network language model in mandarin voice input system. In *2012 8th International Conference on Natural Computation*, pages 270–274. IEEE.

Yujing Si, Qingqing Zhang, Ta Li, Jielin Pan, and Yonghong Yan. 2013. Prefix tree based n-best list rescoring for recurrent neural network language model used in speech recognition system. In *Interspeech*, pages 3419–3423.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*.

Ching Y Suen. 1979. N-gram statistics for natural language understanding and text processing. *IEEE transactions on pattern analysis and machine intelligence*, (2):164–172.

Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. 2015. From feedforward to recurrent lstm neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):517–529.

Conghui Tan, Di Jiang, Jinhua Peng, Xueyang Wu, Qian Xu, and Qiang Yang. 2020. A de novo divide-and-merge paradigm for acoustic model optimization in automatic speech recognition. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3709–3715.

Julian R. Ullmann. 1977. A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words. *The Computer Journal*, 20(2):141–147.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. 1989. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Shiyu Zhou, Linhao Dong, Shuang Xu, and Bo Xu. 2018. Syllable-based sequence-to-sequence speech recognition with the transformer in mandarin chinese. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018*, pages 791–795.

Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. CRC press.

A More details about GMM

Genetic algorithm is an evolutionary algorithm, and its basic principle is to imitate the evolutionary laws of the nature. It starts from a population that represents the potential solution set of the problem, and each generation evolves to produce a better approximate solution according to their fitness in the problem domain. The key factor is to use the genetic operators to combine the crossover and mutation of the population to produce a new potential population. This process will cause the population to be like natural evolution, the offspring population is more adapted to the environment than the previous generation, and the optimal individual in the last generation population can be used as the approximate optimal solution to the problem.

The overall workflow of GMM is shown in Algorithm 1, where two additional hyper parameters $p1$ and $p2$ denote the probabilities of mutation and crossover, respectively.

Algorithm 1 Genetic Match-and-Merge Algorithm

```
Input:source paired models $M_1$,$M_2$,...,$M_n$
Initialize $P = \{M_1, M_2, ... ,M_n\}$,
$P_N = \{M_{N,1}, M_{N,2}, ..., M_{N,n}\}$, $P_R = \{M_{R,1}, M_{R,2}, ..., M_{R,n}\}$
1:  while not converged do
2:      for each $M_{N,i}$ in $P_N$ do
3:          With probability p1 let
            $P_N = P_N \cup$ Mutation($M_{N,i}$)
4:      end for
5:      for each $M_{R,i}$ in $P_R$ do
6:          With probability p1 let
            $P_R = P_R \cup$ Mutation($M_{R,i}$)
7:      end for
8:      Randomly shuffle $P$
9:      for each adjacent models $M_{N,i}$; $M_{N,i+1}$ in $P_N$ do
10:         With probability p2 let
            $P_N = P_N \cup$ Crossover($M_{N,i}$,$M_{N,i+1}$)
11:     end for
12:     for each adjacent models $M_{R,i}$; $M_{R,i+1}$ in $P_R$ do
13:         With probability p2 let
            $P_R = P_R \cup$ Crossover($M_{R,i}$,$M_{R,i+1}$)
14:     end for
15:     Reproduction($P_N \times P_R$)
16:     Let $P$ be the set of top-K pairs $(P_{N,T}, P_{R,T})$
17: end while
```

Although experiments show that the genetic strategy is an effective method, the limitation is that the results of each genetic operator are random, which leads to the generated offspring not necessarily achieving better quality. This inspires us to utilize the evolution results at each time step to pass the information to the next round of evolution, and guide each evolution towards a good quality direction as much as possible. Reinforcement learning regards the learning process as a trial evaluation where a agent selects an action for the environment. After the environment accepts the action, the state of the agent changes, and at the same time, a reward or punishment signalis generated and fed back to the agent. The agent selects the next action based on the reinforcement signal and the current state of the environment. The principle of selection is to increase the probability of receiving positive award. The selected action not only affects the immediate enhancement value, but also affects the state of the environment at the next moment and the final enhancement value. This reinforcement strategy can make up for the deficiencies of GMM, thereby improving time efficiency.

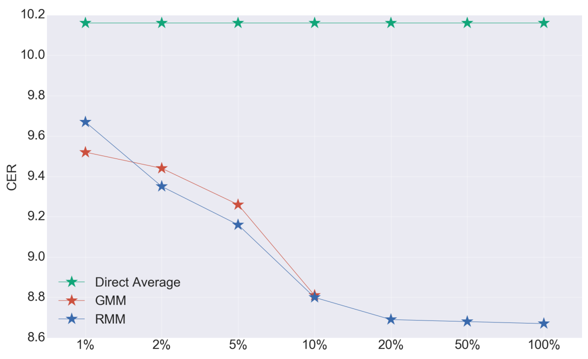

Figure 7: CERs of the target models optimized on different size of validation set.

B Dataset Collection

The detailed statistics of all datasets are shown in Table 1.

C Variation of Validation Data Size

Both GMM and RMM use an extra validation dataset for model merging. To understand the importance of validation data size on the final results, we vary the size of validation dataset and analyze the changes of CERs on the test dataset. We randomly take samples from the complete validation dataset of SLR33, SLR38, SLR47, SLR62 and SLR93 with different proportions at 1%; 2%; 5%;

Dataset	Name	Training		Validation		Test	
		no.wav	duration(h)	no.wav	duration(h)	no.wav	duration(h)
SLR18	THCHS-30	8032	20.5	2667	6.7	2689	6.8
SLR33	Aishell	120018	151.2	14331	18.1	7176	10.0
SLR38	Free ST Chinese Mandarin Corpus	61560	65.5	20520	22.0	20520	22.0
SLR47	Primewords Chinese Corpus Set 1	30232	59.2	10076	19.8	10076	19.8
SLR62	aidatatang 200zh	164905	139.9	24216	20.2	48144	40.2
SLR68	Chinese Read Speech Corpus	621665	727.0	12140	13.9	24279	27.9
SLR93	AISHELL-3	63262	61.1	12387	12.0	12386	12.0
Total	N/A	1069674	1224.4	96337	112.7	125270	138.7

Table 1: Statistics of the datasets

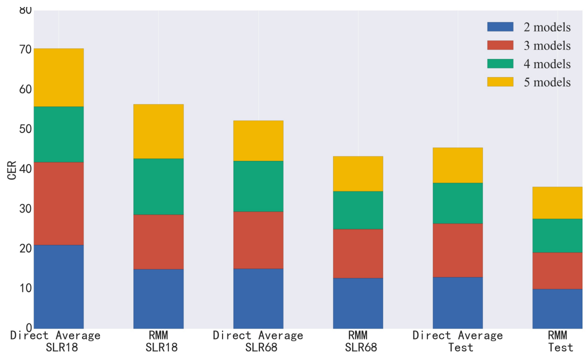

Figure 8: CERs of the target models optimized on different number of the source models.

10%; 20%; 50%; 100%. Then, we perform GMM and RMM based on the subsets and compare the results of these two algorithms with direct average. All Note that when the size of validation subsets is greater than 10%, we no longer test GMM because of its low efficiency. The evaluation results are reported in Figure 7, and we can find that both GMM and RMM benefit from larger validation dataset and can yield better pair of target models. Moreover, only a limited validation subset with 1% proportion can make both two algorithms surpass the direct average. Considering that 1% of the validation data only accounts for approximately 0.3% of the training data, our method is sufficiently practical for real world application, using only a small amount of data to significantly reduce test CER. However, further expansion of data volume does not bring greater improvement to the RMM since the size of validation set is already large enough.

D Influence of Source Model Number

We study the target model quality obtained by different numbers of source models, and all the source models are sequentially and separately added as input to the comparison methods for training. Note that genetic algorithm and transfer learning cannot be input sequentially, so they do not participate in this study. We ensure that the input of all methods is the same sequence, and the results are presented in Figure 8. We can find that the more the number of merged models, the more obvious the performance will be improved. Moreover, our method only needs a smaller number of models to merge a target model with a quality comparable to the target model merged with more models by the direct averaging method. However, it can be seen that not merging all models together can produce better results. The quality of the merging obviously depends on the quality of the individual source models. Because the direct averaging method is only a simple averaging, it has increased the error rate after some models are merged. As our method uses reinforcement learning to guide the search, the searcher assigns lower weights to models with lower quality and higher weights to models with better performance according to the reward.