A De Novo Divide-and-Merge Paradigm for Acoustic Model Optimization in Automatic Speech Recognition

Conghui Tan[1*], **Di Jiang**[1], **Jinhua Peng**[1], **Xueyang Wu**[2], **Qian Xu**[1] and **Qiang Yang**[1,2]
[1]WeBank Co., Ltd., Shenzhen, China
[2]The Hong Kong University of Science and Technology, Hong Kong
{martintan, dijiang, kinvapeng, qianxu}@webank.com, {xwuba, qyang}@cse.ust.hk

Abstract

Due to the rising awareness of privacy protection and the voluminous scale of speech data, it is becoming infeasible for Automatic Speech Recognition (ASR) system developers to train the acoustic model with complete data as before. In this paper, we propose a novel Divide-and-Merge paradigm to solve salient problems plaguing the ASR field. In the Divide phase, multiple acoustic models are trained based upon different subsets of the complete speech data, while in the Merge phase two novel algorithms are utilized to generate a high-quality acoustic model based upon those trained on data subsets. We first propose the Genetic Merge Algorithm (GMA), which is a highly specialized algorithm for optimizing acoustic models but suffers from low efficiency. We further propose the SGD-Based Optimizational Merge Algorithm (SOMA), which effectively alleviates the efficiency bottleneck of GMA and maintains superior performance. Extensive experiments on public data show that the proposed methods can significantly outperform the state-of-the-art.

1 Introduction

Automatic Speech Recognition (ASR) has already become an indispensable part of modern intelligence systems such as voice assistant and client service robot. An effective ASR system relies on a robust acoustic model that is trained over a huge amount of speech data collected from a wide range of domains. However, in real-life scenarios, training acoustic model with complete data is increasingly infeasible due to the following three reasons:

- **R1:** Speech data from different domains are owned by distinct curators, who are typically unwilling to share these data with others due to privacy concerns.
- **R2:** Speech data of multiple curators may be distributed across different computing centers. Traditional distributed computing paradigms such as ParameterServer require frequent information exchange between different computing nodes and is hardly feasible if the data are remotely distributed.
- **R3:** The speech data from a single curator can be voluminous. Traditional optimizing methods like Stochastic Gradient Descent (SGD) are considered to be slow, even for their parallel or asynchronous versions. Instead, one would like to process data in a fully parallel way.

To tackle the aforementioned issues, we propose to optimize the acoustic model in a two-stage fashion: we first train multiple models on different parts of the data independently, and then merge them into a single one. Actually, such a technique has already been applied in some existing applications. For example, [McMahan *et al.*, 2016] proposed to use this technique to resolve the privacy and communication problems (i.e., **R1** and **R2**) in federated learning; and Kaldi [Povey *et al.*, 2011], one of the most widely-used speech recognition toolkits, has adopted it as the default training scheme in order to deal with the efficiency challenge (i.e., **R3**) [Povey *et al.*, 2014]. However, in the phase of merging models, most of the existing work still relies on the simplistic technique of averaging over all the models. Although fairly good performance can be achieved, whether there exists better strategies for model merging is still an open problem.

In this paper, we propose a new Divide-and-Merge paradigm. Different from existing similar approaches, we merge the models in a more data-efficient way, where the model quality is greatly improved with a very limited number of data. In detail, we first propose a merge method based on genetic algorithm named GMA, which is capable of yielding models of great performance. However, its practicality is heavily limited by its poor efficiency. To further tackle this issue, we convert the model merging problem into a mathematical optimization problem via a novel formulation, and develop a new optimization method based on SGD to solve it, which leads to a new method called SOMA. Experiments suggest SOMA can produce models comparable to GMA, but with much lesser computation cost.

Without loss of generality, we focus on the scenario of merging several DNN acoustic models with homogenous structures into a single one and this practice can be straightforwardly applied in both the state-of-the-art ASR systems based upon DNN-HMM or End-to-End architectures.

[*]Contact Author

2 Related Work

2.1 Automatic Speech Recognition

Automatic speech recognition has been studied for a long time. ASR systems typically consist of two components: acoustic model (AM) and language model (LM), where the former one is in charge of capturing the relationship between acoustic inputs and phones, while the latter is used for modeling the probability of possible word sequences.

Traditionally, acoustic model is based on Gaussian mixture model (GMM) and hidden markov model (HMM). However, since the seminal work [Hinton *et al.*, 2012], the deep learning based AMs has become the mainstream in both industrial applications and academic research. Some of the research focus on the end-to-end DNN acoustic model, such as connectionist temporal classification (CTC, [Graves *et al.*, 2006]). However, to the best of our knowledge, the AM with the combination of DNN and classical HMM model still retains the state-of-the-art performance at present.

As for the LM, DNN-based LMs have also drawn a lot of attention. Related works include [Bengio *et al.*, 2003; Morin and Bengio, 2005; Mikolov *et al.*, 2010]. Especially, models with attention mechanism such as BERT [Devlin *et al.*, 2018] has brought breakthrough in performance over previous methods. However, in terms the applications in ASR systems, the traditional back-off n-gram model is still the most common choice due to its simplicity and robustness.

2.2 Acoustic Model Optimization

As the standard method for training deep neural networks, SGD and its variants such as Adam [Kingma and Ba, 2014] are still the first choice for acoustic model optimization. But some efforts are still made to explore other optimization technique. For example, [Cui and Picheny, 2019] proposed to use a combinational optimization strategy with SGD and evolutionary learning. As for the distributed computation setting, asynchronous SGD under the ParameterServer framework is widely adopted [Zhang *et al.*, 2013]. However, asynchronous SGD heavily suffers from the communication bottleneck on the central server. To resolve this issue, [Povey *et al.*, 2014] proposed to train acoustic models on subsets of data totally independently, and merge these models periodically.

On the other hand, we note that there are also some works concerning acoustic model combination (e.g., [Kumar and Gong, 2019; Meinedo and Neto, 2000; Xiong *et al.*, 2018]). We have to clarify its difference with our setting: model combination does not merge the models into a single one in the training phase. However, when doing prediction, all the models are respectively evaluated and their outputs are combined to produce the final results. Hence, it is more related to ensemble learning [Zhou, 2012]. Though model combination can deal with heterogeneous models, it heavily increases the burden of prediction. Moreover, is not suitable for the applications like distributed training where the merged model still needs to be improved iteratively.

3 Problem Setup

Assume we have n acoustic models $\{M_{S,1}, M_{S_2}, \ldots, M_{S,n}\}$ with homogenous structures but different parameters since they are trained on different data. We call these n acoustic models as the *source models*. Our goal is to merge them into one *target model* M_T, which posseses the same structure as as source models but has better performance.

For each acoustic model M_i, we assume it has L DNN layers, and its parameter of l-th layer is denoted as W_i^l $(1 \leq l \leq L)$. W_i^l includes all types of trainable parameters on that layer, such as weight and bias. For notational simplicity , we use the operation on model M_i to denote the operation on all its parameters W_i^l with $l = 1, \ldots, L$. For example, $(M_i + M_j)/2$ is the model generated by averaging all the corresponding parameters of M_i and M_j.

In order to select the best M_T, we need some data to evaluate the quality of M_T, and we refer those data as the validation data. In contrast, the data used for training $\{M_{S,1}, M_{S,2}, \ldots, M_{S,n}\}$ are called as the training data. Though extra validation data is required for our scheme, it will be shown later that the model quality can be greatly improved with very few validation data, which implies it is possible to obtain acoustic models of similar performance with less training data. To measure the performance of acoustic models, we utilize the widely used metric Word Error Rate (WER), which is defined as the minimum edit distance between the ASR hypothesis and the ground truth over the number of words in ground truth.

4 Genetic Merge Algorithm

Since we aim at discovering a better target acoustic model from a set of source models, a relatively straightforward approach is to consider the source models as the initial population and apply the genetic algorithm [Holland and others, 1992]. Genetic algorithm is a class of heuristic search algorithm inspired by biological evolution. It optimizes a group of candidates by repeatedly generate new individuals via operations like mutation and crossover and then offer the ones with large fitness the right to produce offsprings, just like how biological evolution works. Genetic algorithm has two important factors that determines its performance: the scheme of generating offsprings and the strategy of selecting the fittest individuals. In the following, we propose the Genetic Merge Algorithm (GMA), which is caliberated for the scenario of acoustic model optimization in ASR systems.

In GMA, the scheme for generating offsprings includes four different operators. The first three are classical in GA: reproduction, mutation and crossover. Specifically, we choose to use single point mutation operator and the one-point crossover operator respectively. Moreover, inspired by the phenomenon discovered in [McMahan *et al.*, 2016] that directly averaging two neural network models with same initialization but trained on different data can lead to a better one, we propose a new operator called linear interpolation operator. In detail, these four operators works as follows:

- *Reproduction* directly copies the existing models into next generation.
- *Mutation* randomly changes one bit in the binary expression of the parameters for the selected model.
- *Crossover* takes two parent models as the input. It randomly draws an integer l $(1 \leq l < L)$, and the first l

layers of these two models are swapped. For example, if M_1 and M_2 are two input parent models, then two generated offsprings are:

$$M_{\text{new},1} = \{W_1^1, \dots, W_1^l, W_2^{l+1}, \dots, W_2^L\},$$
$$M_{\text{new},2} = \{W_2^1, \dots, W_2^l, W_1^{l+1}, \dots, W_1^L\}.$$

- *Linear interpolation* linearly combines all the parameters of two parent models in a weighted way to generate one new model, i.e.,

$$M_{\text{new}} = \lambda M_1 + (1-\lambda) M_2,$$

where λ is an interpolation coefficient randomly sampled from $(0, 1)$. Obviously, this operator is an extension of simple average.

We utilize WER as the measurement of the fitness and lower WER implies better fitness. For each generation, we evaluate the WER of each acoustic model on the validation set and choose the top-K with the lowest WERs as the parents of the next generation. It is worth noting that larger K brings more diversity into the population and usually leads to better searching result, but it also increases computation cost. Furthermore, inspired by the fact that the simple average of all the sources models is already a good choice for M_T again, we also include the averaged model $\sum_{i=1}^n M_{S,i}/n$ into the initial population. Such initialization provides a better start for GMA and reduces the time needed for converging. Besides, it ensures that the final acoustic model generated by GMA is always better than simple average. The workflow of GMA is presented in Algorithm 1. GMA requires three additional hyperparameters: p_1, p_2 and p_3, which are the probabilities that mutation, crossover and linear interpolation operators are applied to generate offsprings respectively.

Algorithm 1 Genetic Merge Algorithm (GMA)

```
input : source models $M_{S,1}, M_{S,2}, \dots, M_{S,n}$
Initialize $P = \{M_{S,1}, \dots, M_{S,n}, \sum_{i=1}^n M_{S,i}/n\}$
while not converged do
    $P' = P$                                  // Reproduction
    foreach $M_i$ in $P$ do
        With probability $p_1$ let
          $P' = P' \cup \text{Mutation}(M_i)$
    end
    Randomly shuffle $P$
    foreach adjacent models $M_i, M_{i+1}$ in $P$ do
        With probability $p_2$ let
          $P' = P' \cup \text{Crossover}(M_i, M_{i+1})$
        With probability $p_3$ let
          $P' = P' \cup \text{LinearInterpolation}(M_i, M_{i+1})$
    end
    Compute the WERs of the models in $P'$ on
      validation set
    Let $P$ be the set of top-$K$ models in $P'$
end
output: $M_T$ =model in $P$ with lowest WER
```

5 SGD-Based Optimizational Merge Algorithm

With its superior performance in generating high-quality acoustic models, GMA suffers from extremely low efficiency. Processing a few acoustic models with GMA on a small validation data (e.g., 5 source on the validation set containing 10 hours of speech data) already requires several days, making it hardly applicable real-life applications.

To tackle this issue, we propose the SGD-Based Optimizational Merge Algorithm (SOMA), which enjoys similar performance as GMA but much more efficient. The major challenge of developing this method is how to convert our acoustic model merging problem into a mathematical optimization problem that SGD can be applied.

The key observation is that any model M generated by GMA can be layer-wisely presented by the following formula:

$$W^l = \sum_{i=1}^n \theta_i^l W_{S,i}^l + \Delta W^l \qquad \text{s.t. } \theta_i^l \geq 0,\ \sum_{i=1}^n \theta_i^l = 1 \tag{1}$$

for all its layers W^l $(1 \leq l \leq L)$. Here the summation term $\sum_{i=1}^n \theta_i^l W_{S,i}^l$ corresponds to the linear interpolation and crossover operators, while the extra variable ΔW^l catches the change introduced by mutation. This fact is rigorously justified by the following proposition:

Proposition 1. *M in the form of (1) covers any model generated by GMA. Besides, term ΔW^l is brought by the mutation operation. In other words, it always holds that $\Delta W^l = 0$ if mutation operator is not applied.*

Proof. We prove this proposition by induction.

For any source model $M_{S,i}$, it is obvious that (1) can recover it by setting $\Delta W^l = 0$, $\theta_i^l = 1$ and all other $\theta_j^l = 0$ with $j \neq i$. For the simple average $\sum_{i=1}^n M_{S,i}/n$, we can choose $\theta_i^l = 1/n$ for all i and also $\Delta W^l = 0$ for all layers. Hence, all the models in the initial generation of GMA can be presented by (1) with $\Delta W^l = 0$.

Assume all the models in one generation satisfies equation (1), then we proceed to show the proposition also holds for the next generation. For model generated by the mutation operator, it can be formulated into (1) by changing the quantity ΔW^l of its parent while inheriting the other terms. Since the crossover operator just swaps some layers of two models, and the internal structures of each layer is still preserved, while the constraints in (1) are imposed layer-wisely, the resulting models should still also stick to the pattern of (1) once their parents do, but with the corresponding θ_i^l and ΔW^l swapped. As for the linear interpolation operator, assume M_1 and M_2 are two input parents, then their layers can be written as:

$$W_1^l = \sum_{i=1}^n \alpha_i^l W_{S,i}^l + \Delta W_1^l,$$
$$W_2^l = \sum_{i=1}^n \beta_i^l W_{S,i}^l + \Delta W_2^l,$$

where α_i^l and β_i^l are two groups of realization for θ_i^l and satisfy the constraints for θ_i^l in (1). Thus, the parameter of the linearly interpolated model should be:

$$\begin{aligned}
&W_{\text{new}}^l \\
=&\lambda W_1^l + (1-\lambda)W_2^l \\
=&\lambda\big(\sum_{i=1}^n \alpha_i^l W_{S,i}^l + \Delta W_1^l\big) \\
&+ (1-\lambda)\big(\sum_{i=1}^n \beta_i^l W_{S,i}^l + \Delta W_2^l\big) \\
=&\sum_{i=1}^n \underbrace{\big(\lambda\alpha_i^l + (1-\lambda)\beta_i^l\big)}_{\theta_i^l} W_{S,i}^l + \underbrace{\lambda\Delta W_1^l + (1-\lambda)\Delta W_2^l}_{\Delta W_{\text{new}}^l}.
\end{aligned}$$

In the next, we need to show θ_i^l satisfies the constraints in (1). The non-negativity of θ_i^l is obvious, considering that $\alpha_i^l \geq 0$, $\beta_i^l \geq 0$ and $\theta \in (0,1)$. Besides,

$$\begin{aligned}
\sum_{i=1}^n \theta_i^l =& \sum_{i=1}^n \big(\lambda\alpha_i^l + (1-\lambda)\beta_i^l\big) \\
=&\lambda\sum_{i=1}^n \alpha_i^l + (1-\lambda)\sum_{i=1}^n \beta_i^l \\
=&\lambda + (1-\lambda) \\
=&1.
\end{aligned}$$

Therefore, all the offsprings produced by any operator still follows the pattern in (1). Finally, from the above argument, it can be seen that ΔW^l of the generated model of both crossover and linear interpolation operators must be 0 once the corresponding terms of their parents are 0. Along with the fact that $\Delta W^l = 0$ for all the models in the initial generation, we can conclude that ΔW^l is introduced by mutation if it is non-zero. $\square$

Now, we have already defined the pattern how the target model should follow. Then, we can formulate our optimization problem as:

$$\begin{array}{ll}
\min_{W^l,\theta_i^l,\Delta W^l} & \ell(M) \\
\text{s.t.} & W^l = \sum_{i=1}^n \theta_i^l W_{S,i}^l + \Delta W^l \\
& \theta_i^l \geq 0,\ \sum_{i=1}^n \theta_i^l = 1,
\end{array} \tag{2}$$

where M is the model consisting of parameters $\{W^1, \ldots, W^L\}$, and $\ell(M)$ is the loss function of model M on the validation data. Any common training criterion for DNN-based acoustic model can be used as the loss function here, such as maximum mutual information (MMI, [Bahl *et al.*, 1986]).

Finally, due to the different nature between SGD and genetic algorithm, it is much easier for SGD to overfit the validation data when solving (2). This is because ΔW^l can be arbitrary in our current formulation, and it is possible that ΔW^l becomes large enough to dominate the other terms. To avoid such problem, we impose an extra restriction on ΔW^l that its magnitude can not exceed the whole parameter W^l up to a constant factor $\rho \geq 0$, e.g., $\rho = 0.01$. As a result, our formal formulation for model merging turns into:

$$\begin{array}{ll}
\min_{W^l,\theta_i^l,\Delta W^l} & \ell(M) \\
\text{s.t.} & W^l = \sum_{i=1}^n \theta_i^l W_{S,i}^l + \Delta W^l \\
& \theta_i^l \geq 0,\ \sum_{i=1}^n \theta_i^l = 1 \\
& \|\Delta W^l\| \leq \rho\|W^l\|.
\end{array} \tag{3}$$

5.1 Solving Optimization Problem

Though we have already formulated our problem into an optimization problem, it is still unclear how to solve it, because this problem seems complicated by having many constraints. In this subsection, we develop a new approach to solve it.

To solve (3), one of our basic strategy is that we will not directly update variable W^l by SGD. Instead, we only update θ_i^l and ΔW^l, while the value of W^l is inferred from these two group of variables according to the constraint

$$W^l = \sum_{i=1}^n \theta_i^l W_{S,i}^l + \Delta W^l$$

at the beginning of each iteration. And the latest estimation of W^l will work as the bridge for updating θ_i^l and ΔW^l.

To do updates for θ_i^l and ΔW^l, we first need to compute the gradients of them. According to the chain rule, the gradient of each θ_i^l can be derived as:

$$\frac{\partial \ell}{\partial \theta_i^l} = \frac{\partial \ell}{\partial W^l} \cdot \frac{\partial W^l}{\partial \theta_i^l} = \frac{\partial \ell}{\partial W^l} \cdot W_{S,i}^l,$$

where $\cdot$ standards for the dot product of matrices. Therefore, we just need to compute the stochastic gradient of model $M = \{W^1, \ldots, W^L\}$ by back-propagation as normal, then the computation of the derivative of θ_i^l becomes trivial from the above equation. As for ΔW^l, by chain rule again, we can show its gradient is exactly the same as the gradient of W^l.

After conducting one step of SGD, we further need to do projection operations to ensures the other two constraints are still satisfied. The constraint

$$\theta_i^l \geq 0 \quad \text{and} \quad \sum_{i=1}^n \theta_i^l = 1, \tag{4}$$

is the so-called simplex constraint, which is well-studied in optimization literature and efficient methods for dealing with it are already known [Chen and Ye, 2011; Condat, 2016]. Hence, we will not dive into the detail how this projection should be done, but simply denote the projection operator of it as $\Pi(\cdot)$. While for

$$\left\|\Delta W^l\right\| \leq \rho \left\|W^l\right\|, \tag{5}$$

we actually just need to scale ΔW^l to make it smaller, once we find ΔW^l violates this constraint. Besides, it is trivial to show that the optimal scaling factor should be:

$$\gamma_l = \frac{\rho\|W^l\|}{\|\Delta W^l\|}.$$

Now we have completed all building blocks of our algorithm. The full algorithm is summarized in Algorithm 2. Here $\eta > 0$ is the step size hyperparameter for SGD. In the initialization stage, we just set $\theta_i^l = 1/n$ and $\Delta W^l = 0$, which implies we choose the model obtained by simple average as the initial model.

Dataset	Name	Training		Validation		Test	
		no. wav	duration (h)	no. wav	duration (h)	no. wav	duration (h)
SLR18	THCHS-30	7984	20.4	2657	6.7	2747	7.0
SLR33	Aishell	120418	151.2	14331	18.1	7176	10.0
SLR38	Free ST Chinese Mandarin Corpus	61698	65.9	20395	21.8	20507	22.0
SLR47	Primewords Chinese Corpus Set 1	30366	59.6	10092	19.8	10212	20.1
SLR62	aidatatang_200zh	164905	139.9	24216	20.2	48144	40.2
Sum		385371	437.1	71691	86.7	88786	99.3

Table 1: Statistics of the datasets

Algorithm 2 SGD-Based Optimizational Merging Algorithm (SOMA)

```
input : source models $M_{S,1}, M_{S,2}, \ldots, M_{S,n}$
Initialize $\theta_i^l = 1/n$ and $\Delta W^l = 0$ for all $i$ and $l$
while not converged do
    // The following operations are done for all $i$ and $l$
    Let $W^l = \sum_{i=1}^{n} \theta_i^l W_{S,i}^l + \Delta W^l$
    Draw a batch of samples from validation data, and compute $\frac{\partial \ell}{\partial W^l}$ by back-propagation
    Update by SGD:
        $\theta_i^l = \theta_i^l - \eta \frac{\partial \ell}{\partial W^l} \cdot W_{S,i}$
        $\Delta W^l = \Delta W^l - \eta \frac{\partial \ell}{\partial W^l}$
    Let $\theta_i^l = \Pi(\theta_i^l)$    // $\Pi(\cdot)$ is the projection operator for (4)
    if (5) not hold then
        Let $\Delta W^l = \Delta W^l \cdot \frac{\rho \|W^l\|}{\|\Delta W^l\|}$
    end
end
output: $M_T$ with parameters
        $W_T^l = \sum_{i=1}^{n} \theta_i^l W_{S,i}^l + \Delta W^l$
```

6 Experiments

6.1 Experimental Setup

In order to ensure the reproducibility of the experiments, we conduct all experiments on public datasets. Specifically, we collected five speech dataset from the OpenSLR[1] website, which are SLR18, SLR33, SLR38, SLR48 and SLR62. All of them contain Chinese speech records in wav format with a sampling rate of 16kHz. Each dataset includes training, validation and test sets [2]. The detailed statistics of all the datasets are presented in Table 1.

As a testbed, we develop a full-fledged ASR system through the open-source toolkit Kaldi [Povey *et al.*, 2011]. Its built-in "Chain" model is used as the acoustic model of the ASR system. The DNN component of the "Chain" model is implemented by Time Delay Neural Network (TDNN) [Waibel *et al.*, 1989] and the other components of the model such as HMM are pre-trained. The backoff n-gram model with $n = 3$ is used as the language model, which is trained with the SRILM toolkit [Stolcke, 2002]. The whole system is deployed on a machine with CentOS, Intel Xeon CPU of 72 cores, NVIDIA Tesla K80 GPU and 314GB memory.

Five TDNNs (i.e., the DNN components of the corresponding "Chain" models) with the same initialization are trained on the training sets of the five datasets respectively. The five TDNNs play the roles of the source models. Considering the slow speed of GMA, we sample a subset from the collection of the validation data of the five datasets with a proportion 10% as the validation set. Both model merging methods will work on this set to optimize the target model by default. All the reported WERs in our experiments are computed on the test set unless otherwise specified.

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

[2]For dataset which does not have splits in advance, we randomly split it into training, validation and test sets with proportions 60%: 20%: 20%

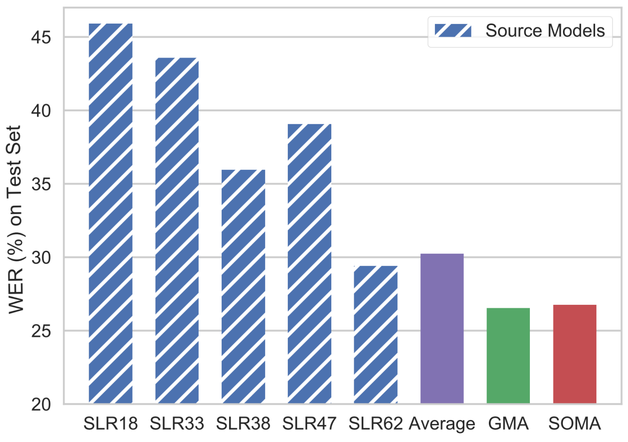

Figure 1: WERs of the sources models and target models generated by different methods.

6.2 Effectiveness Evaluation

We first compare the performance of models generated by different methods: direct average, GMA and SOMA. Both GMA and SOMA have been run for enough time until they converge. In detail, SOMA has been run for 10 iterations, where we define one iteration as one passes through the sampled validation data. While for GMA, it has been run for 100 generations with a population size $K = 15$, which results in

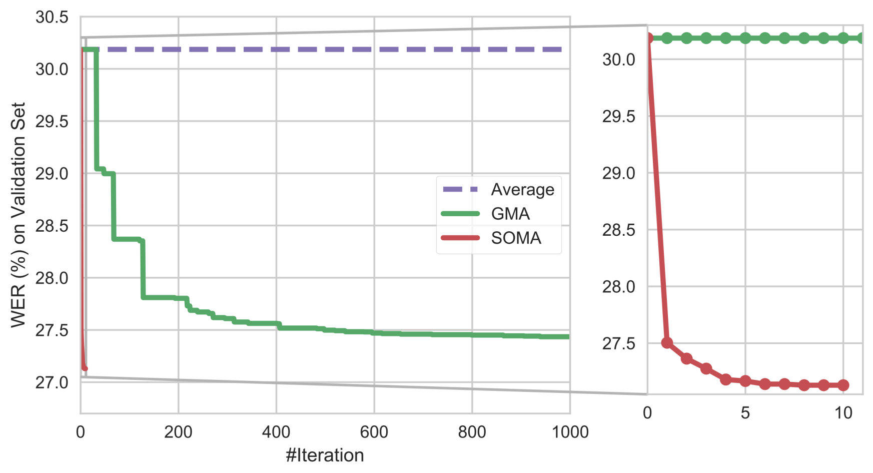

Figure 2: Convergence curves of GMA and SOMA.

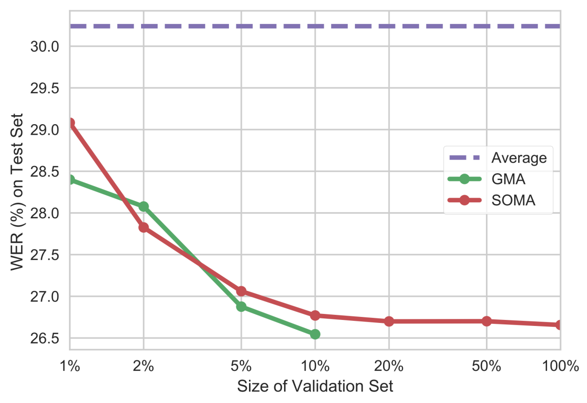

Figure 3: WERs of the target models optimized on different size of validation set. Log-scale is adopted for x-axis.

more than 1000 iterations since each generated model needs to be evaluated on the validation set once.

The WERs of all the models, including the sources models, are reported in Figure 1. Due to the different sizes and qualities of the training data, the performances the source models vary a lot, which brings challenges into model merging. Though direct average achieves a WER lower than most the source models, it is still slightly worse than the best one. As a comparison, GMA and SOMA obviously outperform all of them. Between them, GMA works better than SOMA, but their difference (0.2%) is quite limited.

6.3 Efficiency Evaluation

Though GMA and SOMA have close performance in terms of the generated model quality, they differ substantially in their efficiency. To demonstrate this, we draw the convergence curves of both methods in Figure 2, where the WERs of the best-so-far generated models over different iterations are reported. It can be observed that SOMA converges quickly, so that the WER is greatly reduced after just one iteration, and it converges in less than 10 iterations. However, GMA fails to generate a better model than the direct average within the first 30 iterations. It improves the models in a very slow way. Even after 1000 iterations, it still falls behind SOMA. Therefore, we can conclude that GMA is impractical on large datasets due to its poor efficiency.

6.4 Variation of Validation Data Size

Considering both GMA and SOMA requires an extra set of validation data for model merging compared to direct average, in this part we will vary the size of the validation set, and see what will happen to them. We randomly sample subsets of different sizes from the complete validation set with proportions $\{1\%, 2\%, 5\%, 10\%, 20\%, 50\%, 100\%\}$, and run GMA and SOMA on them, and finally evaluate the WERs on the test set. Due to the slow speed of GMA, it is not tested on the validation subsets larger than 10%.

Overall, we can observe that larger validation set yields better target models for both GMA and SOMA. However, if the data size is already large enough, say 10% of the whole validation set, further increasing data volume does not bring too much help for SOMA. Moreover, we can see that both GMA and SOMA can beat direct average with a very limited number of validation data such as proportion 1%. Note that 1% of the validation data only corresponds to approximately 0.2% of the training data, but can already help reduce test WER for more than one percent comparing to simple average. This fact provides a strong reason for why to use our methods.

7 Conclusion

In this paper, we propose a novel Divide-and-Merge paradigm for optimizing the acoustic model in ASR. In the Divide phase, we train multiple acoustic models independently on distinct parts of data. In the Merge phase, instead of applying the simplistic averaging scheme for merging acoustic models, we propose two novel algorithms with significantly better performance: GMA and SOMA, where the former one is based on genetic algorithm and the latter one adopts SGD with a novel mathematical formulation. Experiments show that both of them can greatly improve acoustic model quality with very limited amount of validation data. Besides, SOMA demonstrates superior efficiency and can be easily applied to large-scale speech data.

References

[Bahl *et al.*, 1986] Lalit R Bahl, Peter F Brown, Peter V De Souza, and Robert L Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *proc. icassp*, volume 86, pages 49–52, 1986.

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[Chen and Ye, 2011] Yunmei Chen and Xiaojing Ye. Projection onto a simplex. *arXiv preprint arXiv:1101.6081*, 2011.

[Condat, 2016] Laurent Condat. Fast projection onto the simplex and the simplex and the ℓ_1 ball. *Mathematical Programming*, 158(1-2):575–585, 2016.

[Cui and Picheny, 2019] Xiaodong Cui and Michael Picheny. Acoustic model optimization based on evolutionary stochastic gradient descent with anchors for automatic speech recognition. *arXiv preprint arXiv:1907.04882*, 2019.

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

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

[Hinton *et al.*, 2012] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[Holland and others, 1992] John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[Jiang *et al.*, 2019] Di Jiang, Yuanfeng Song, Yongxin Tong, Xueyang Wu, Weiwei Zhao, Qian Xu, and Qiang Yang. Federated topic modeling. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1071–1080, 2019.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kumar and Gong, 2019] Kshitiz Kumar and Yifan Gong. Static and dynamic state predictions for acoustic model combination. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2782–2786. IEEE, 2019.

[McMahan *et al.*, 2016] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[Meinedo and Neto, 2000] Hugo Meinedo and Joao P Neto. Combination of acoustic models in continuous speech recognition hybrid systems. In *Sixth International Conference on Spoken Language Processing*, 2000.

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

[Morin and Bengio, 2005] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer, 2005.

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

[Povey *et al.*, 2014] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. Parallel training of deep neural networks with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455*, 2014.

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

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

[Xiong *et al.*, 2018] Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke. The microsoft 2017 conversational speech recognition system. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5934–5938. IEEE, 2018.

[Zhang *et al.*, 2013] Shanshan Zhang, Ce Zhang, Zhao You, Rong Zheng, and Bo Xu. Asynchronous stochastic gradient descent for dnn training. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6660–6663. IEEE, 2013.

[Zhou, 2012] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.