Evolution Strategy Based Automatic Tuning of Neural Machine Translation Systems

Hao Qin[1], *Takahiro Shinozaki*[1], *Kevin Duh*[2]

[1]Tokyo Institute of Technology, Japan
[2]Johns Hopkins University, USA

qin.h.aa@m.titech.ac.jp, shinot@ict.e.titech.ac.jp, kevinduh@cs.jhu.edu

Abstract

Neural machine translation (NMT) systems have demonstrated promising results in recent years. However, non-trivial amounts of manual effort are required for tuning network architectures, training configurations, and pre-processing settings such as byte pair encoding (BPE). In this study, we propose an evolution strategy based automatic tuning method for NMT. In particular, we apply the covariance matrix adaptation-evolution strategy (CMA-ES), and investigate a Pareto-based multi-objective CMA-ES to optimize the translation performance and computational time jointly. Experimental results show that the proposed method automatically finds NMT systems that outperform the initial manual setting.

1. Introduction

Neural machine translation (NMT) is a new approach to translation and has shown promising results in recent years. Many active ongoing research are focused on developing new architectures and training methods. When developing a machine translation system based on neural network structure, the major design question is how to set the meta-parameter values of the network structure and training configuration, so that the system performs well in terms of translation performance and computational cost. For network structure design, important meta-parameters include what kind of network should be applied, the number of layers, the number of units per layer, and unit type. With the increase of layers, the problem becomes more complex. For training configurations, important meta-parameters include learning algorithm, learning rate, dropout ratio and so on. All of these meta-parameters interact with each other and affect the performance of the whole system in a subtle way, thus they need to be tuned simultaneously.

Usually, these meta-parameters are tuned by human experts based on their experience. Such work requires much effort. In some ways such bottleneck may limit the wider adoption of NMT, or lead us into locally-optimal design decisions. Meanwhile, more powerful computing resource are available for academic and public use. Our motivation is to replace tedious manual tuning work with automatic computation conducted by computers.[1] As neural network training process is conducted off-line and a well-trained model can be used repeatedly, it is meaningful to allocate more computational resources to meta-parameter tuning as it can alleviate manual work.

Grid search is a simple method for meta-parameter optimization. However, as the number of meta-parameters increases, it becomes less tractable. This is because the number of lattice points increases in an exponential way with the increase of the number of meta-parameters. For example, if there are ten meta-parameters to be tuned and we only try five values for each parameter, it requires more than 750 billion (5^{10}) evaluations. In the case of NMT, training and evaluating one instance requires significant computational resource and time. Thus grid search is not a feasible method even using the fastest super computer. A black box meta-parameter optimization framework that can intelligently search a proper solution is needed.

Related work has proposed many meta-heuristic optimization methods such as genetic algorithms (GA) [1], evolutionary strategies (ES) [2], and Bayesian optimization (BO) [3]. They have all demonstrated success in many practical problems. In this study, we focus on an ES method called covariance matrix adaptation-evolution strategy (CMA-ES) [4] and its multi-objective extension [5, 6]. CMA-ES has been shown to be a practical and simple-to-implement algorithm that finds good solutions under few instance evaluations. To the best of our knowledge, this is the first work on applying CMA-ES to NMT.

Experiments are implemented with the Nematus machine translation toolkit [7]. Both single-objective optimization based on BLEU and multi-objective optimization based on BLEU and validation time are investigated, where validation time represent the time cost of generating translations on a validation data set. We show that CMA-ES can automatically find NMT models that improves upon the initial setting. Further, we analyze the factors that affect translation performance and computational time cost.

In the following, we introduce CMA-ES and its multi-

[1]We use the term "tuning" to refer to "hyperparameter search" in neural networks; note this differs from the fine-tuning in neural networks and the development set tuning in statistical MT system building.

objective extension in Section 2, then describe the machine translation system used in this work in Section 3. Experiment setup will be described in Section 5. Experiment results and analysis are in Section 6 and Section 7, followed by related work and conclusion.

2. CMA-ES META-PARAMETER OPTIMIZATION

2.1. CMA-ES

CMA-ES is a population-based meta-heuristics optimization method. Like GA, it encodes potential solutions as genes. The differences between GA and CMA-ES are that CMA-ES uses a fixed length vector $\boldsymbol{x}$ of real values as a gene, and uses a full covariance Gaussian distribution as gene distribution instead of directly representing them by a set of genes. In CMA-ES, it is assumed that the value of the objective function $f(\boldsymbol{x})$ can be evaluated, but the availability of an analytical form of the objective function $f(\boldsymbol{x})$ and its differentiability are not needed. Figure 1 shows the basic process of using CMA-ES.

In our experiment, the objective function $f(\boldsymbol{x})$ represents the performance of the machine translation system trained with a gene $\boldsymbol{x}$ encoding a set of meta-parameters. The meta-parameters include model structure and training configurations. Specifically, mean and covariance parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ of a Gaussian distribution for $\boldsymbol{x}$ is estimated by CMA-ES so that the distribution is concentrated in a region where $f(\boldsymbol{x})$ has a high value[2] by maximizing expectation $\mathbb{E}\left[f(\boldsymbol{x})|\boldsymbol{\theta}\right]$ as shown in Eq. (1).

$$\begin{aligned}\widehat{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{x}|\widehat{\boldsymbol{\theta}}) \ \text{ s.t. } \ \widehat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}\left[f(\boldsymbol{x})|\boldsymbol{\theta}\right] \\ &= \arg\max_{\boldsymbol{\theta}} \int f(\boldsymbol{x})\mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta})d\boldsymbol{x}.\end{aligned} \tag{1}$$

In order to solve the problem efficiently, the natural gradient based gradient ascent is used. The expectation can be approximately computed by Monte Carlo sampling with the function evaluation $y_k = f(\boldsymbol{x}_k)$ as shown in Eq. (2).

$$\tilde{\nabla}_{\boldsymbol{\theta}}\mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}] \approx \frac{1}{K}\sum_{k=1}^{K} y_k \boldsymbol{F}_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}} \log\mathcal{N}(\boldsymbol{x}_k|\boldsymbol{\theta}), \tag{2}$$

where $\boldsymbol{x}_k$ is a sample drawn from the previously estimated distribution $\mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_{n-1})$, and $\boldsymbol{F}$ is the Fisher information matrix.

Analytical forms of the updates of $\widehat{\boldsymbol{\mu}}_n$ and $\widehat{\boldsymbol{\Sigma}}_n$ are obtained by substituting the concrete Gaussian form into Eq.(2), leading to:

$$\begin{cases}\widehat{\boldsymbol{\mu}}_n = \widehat{\boldsymbol{\mu}}_{n-1} + \epsilon_{\boldsymbol{\mu}} \sum_{k=1}^{K} w(y_k)(\boldsymbol{x}_k - \widehat{\boldsymbol{\mu}}_{n-1}) \\ \widehat{\boldsymbol{\Sigma}}_n = \widehat{\boldsymbol{\Sigma}}_{n-1} + \epsilon_{\boldsymbol{\Sigma}} \sum_{k=1}^{K} w(y_k) \\ \qquad \cdot\big((\boldsymbol{x}_k - \widehat{\boldsymbol{\mu}}_{n-1})(\boldsymbol{x}_k - \widehat{\boldsymbol{\mu}}_{n-1})^{\intercal} - \widehat{\boldsymbol{\Sigma}}_{n-1}\big)\end{cases} \tag{3}$$

[2]Importantly, it is worth emphasizing that CMA-ES is a black-box method that makes no assumption on the relationship between gene value and system performance. The search distribution used to sample next generation genes is Gaussian, but $f(\boldsymbol{x})$ is not assumed to be Gaussian.

where $^{\intercal}$ is the matrix transpose. Note that as in [8], y_k in Eq.(2) is approximated in Eq.(3) as a weight function $w(y_k)$, which is defined as :

$$w(y_k) = \frac{\max\{0, \log(K/2+1) - \log(\mathrm{R}(y_k))\}}{\sum_{k'=1}^{K}\max\{0, \log(K/2+1) - \log(\mathrm{R}(y_{k'}))\}} - \frac{1}{K}, \tag{4}$$

and $\mathrm{R}(y_k)$ is a ranking function that returns the descending order of y_k among $y_{1:K}$. That is, $\mathrm{R}(y_k) = 1$ for the highest y_k, and $\mathrm{R}(y_k) = K$ for the smallest y_k. This equation only considers the order of y, which makes the updates less sensitive to the choice of evaluation measurements. As the correspondence to GA, the set of sampled genes $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_K\}$ represents a population of a generation, and an iteration of the gradient ascent corresponds to a generation.

2.2. Multi-objective CMA-ES using the Pareto frontier

In addition to the accuracy of translation, objectives such as time cost are also important in practice. Suppose we want to maximize J objectives $F(\boldsymbol{x}) \triangleq [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_J(\boldsymbol{x})]$ jointly with respect to $\boldsymbol{x}$. To handle the situation where the objectives may conflict with each other, we adopt Pareto optimality [9, 10]. We say $\boldsymbol{x}_k$ *dominates* $\boldsymbol{x}_{k'}$ if $f_j(\boldsymbol{x}_k) \geq f_j(\boldsymbol{x}_{k'})\ \forall\ j = 1, .., J$ and $f_j(\boldsymbol{x}_k) > f_j(\boldsymbol{x}_{k'})$ for at least one objective j, and write $F(\boldsymbol{x}_k) \triangleright F(\boldsymbol{x}_{k'})$. When given a set of candidate solutions, $\boldsymbol{x}_k$ is *Pareto-optimal* iff no other $\boldsymbol{x}_{k'}$ exists such that $F(\boldsymbol{x}_{k'}) \triangleright F(\boldsymbol{x}_k)$. There are several Pareto-optimal solutions given a set of candidates. The subset of all Pareto-optimal solutions is known as the Pareto frontier. Compared to combining multiple objectives into a single objective via an weighted linear combination, the Pareto definition has an advantage that weights need not be specified and it is more general.

CMA-ES can be extended to optimize multiple objectives by modifying the rank function $\mathrm{R}(y_k)$ used in Eq.(4). Given a set of solutions $\{\boldsymbol{x}_k\}$, we first assign rank = 1 to those on the Pareto frontier. Then, we exclude these rank 1 solutions and compute the Pareto frontier again for the remaining solutions, assigning them rank 2. This process is iterated until no $\{\boldsymbol{x}_k\}$ remain, and we obtain a ranking of all solutions according to multiple objectives in the end. Figure 2 shows the intuition behind multi-objective optimization in our work, where BLEU score and negative validation time are used as the objectives. We expect superior individuals with higher BLEU score and lower translation time than the initial one are obtained by the automatic optimization by CMA-ES.

3. Neural Machine translation

3.1. Encode-Decoder Model

The neural machine translation (NMT) system we used in this experiment is based on an attentional encoder-decoder architecture as implemented in Nematus [7]. This is very similar to the structure proposed by [11].

The NMT model is part of the family of models using

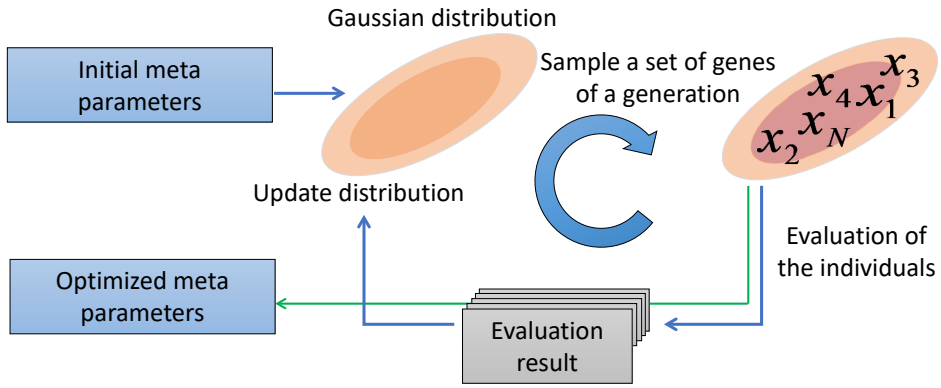


Figure 1: Automatic system tuning process using CMA-ES.

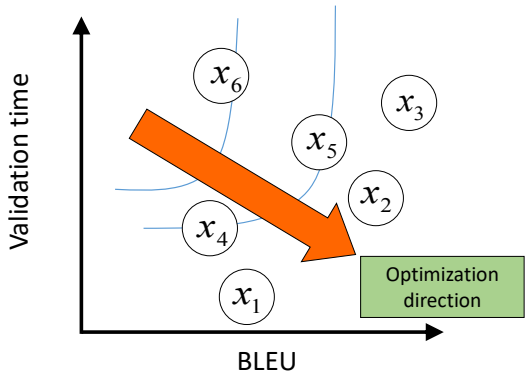


Figure 2: Pareto based optimization for two objectives.

encoder-decoder with recurrent neural networks. The encoder is implemented as a bidirectional neural network with gated recurrent unit [12]. First, it reads the input sentence, which is a sequence of m words $x = x_1, ..., x_m$. The forward RNN reads the input sequence from x_1 to x_m and the reverse RNN reads the sequence from x_m to x_1. The hidden states of the two RNN at each time-step are concatenated to form the encoding, or annotation vectors $h_1, ..., h_m$.

The decoder is trained to predict the target word sequence $y = (y_1, ..., y_n)$, and is also implemented as a recurrent neural network. The decoder predicts each word y_i based on a recurrent state s_i, previous word y_{i-1} and a context vector c_i. The context vector c_i is computed as a weighted sum of annotations $c_i = \sum_{j=1,...,m} \alpha_{ij} h_j$, where the weight α_{ij} is based on a single-layer feedforward neural network. The weights can be viewed as "attention" on the input. During training, we use the previous word y_{i-1} according to the target reference; during evaluation or test, we use the word previously predicted by the RNN decoder as y_{i-1} and run a beam search to generate the translation (beam is 5 in our case).

3.2. Subword Preprocessing

We follow the work of [13] in subword preprocessing, which uses byte pair encoding (BPE) to split words into subword units. The motivation is to reduce the number of distinct vocabulary items in the Encoder-Decoder model. Large vocabulary may lead to slower models and sparser statistics.

We briefly describe the BPE preprocessing procedure here: First the symbol vocabulary is initialized with all characters in the training set. The frequency of each symbol pair is calculated, and we iteratively merge the most frequent pair to create a new symbol. In other words, each merge operation produces a new vocabulary item that represents a character n-gram. Very frequent character n-grams, such as frequent words, eventually become a single symbol. The final vocabulary size of BPE equals to the size of the initial character set, plus the number of BPE merge operations.

While BPE is a simple preprocessing method to handle the large vocabulary problem in NMT, the optimal number of BPE merge operations in unclear. Intuitively, a larger vocabulary size should lead to better translation accuracy, assuming sufficient data to estimate the model parameters. The effect on translation time is uncertain: on one hand, smaller vocabulary implies a faster softmax operation in the RNN decoder, but also a longer sequence to process. Finally, the impacts of BPE vocabulary size may be different for source and target.

4. EXPERIMENTAL SETUP

4.1. Data

We performed two sets of experiments: single-objective experiment and multi-objective experiment. In the single-objective experiment, we optimize translation accuracy, which is measured by BLEU on the validation (development) set. In the multi-objective one, we optimize translation accuracy and computational cost jointly. The computational cost is measured by the translation time (seconds) on validation set. We use the data from Kyoto Free Translation Task version 1.4 (KFTT)[3] for both experiments. KFTT contains Wikipedia articles about Kyoto tourism and traditional Japanese culture, religion, and history. These articles are originally in Japanese and are manually translated

[3] http://www.phontron.com/kftt/

Table 1: Data statistics

	Articles	Sentence pairs	Japanese words	English words
Train	14126	330k	6.2M	5.9M
Dev	15	1166	27.8k	24.3k
Test	15	1160	29.6k	26.7k

into English by NICT.[4] The English side is preprocessed (i.e. tokenized, lowercased, filtered to exclude sentences more than 40 words) using standard machine translation tools from Moses, and the Japanese side is word-segmented using Kytea[5]. Both sides are then broken in subword units independently using BPE, where the exact BPE meta-parameter (number of merge operations) is automatically tuned via CMA-ES.

The bitext is separated into training, validation (dev), and test sets. The training set is used for training the NMT models, development set used for measuring BLEU and computation time, the objectives to be optimized. The test set is only used for reporting final results. We focus on the Japanese-English direction, and the baseline results using Giza++/Moses PBMT on the KFTT leaderboard is 15.41 BLEU for dev and 17.68 BLEU for test. There is a stronger result of 16.93 BLEU for dev and 19.35 BLEU for test on the leaderboard. It is a Moses PBMT system that utilizes pre-ordering (permuting Japanese words into English SVO order prior to training and translation), which have demonstrated substantial gains in Japanese-English tasks [14, 15]. We compare with the standard 15.41 BLEU baseline using bitext in their original word order, and leave pre-ordering's effect on NMT to future work. Table 1 summarizes the data used for experiments.

4.2. Meta-parameters

Table 2 shows meta-parameters that are subject to tuning by CMA-ES. All Nematus meta-parameters that are not shown in the table are set to their defaults. The meta-parameters we tune for can be divided into model architecture (e.g. size of embedding, LSTM unit) and training configuration (learning rate, drop out). Their initial values were manually tuned slightly to achieve a reasonable starting BLEU of 16.48 on the dev set and 15.13 on the test set. The corresponding computation times for decoding the dev and test sets are 248 and 230 seconds, respectively.

Our goal in the experiment is to run evolution and observe whether these initial values and corresponding BLEU/time can be automatically improved without manual effort. If evolution can search through a large range for meta-parameters, we can expect a highly optimized system. That is the generalization of this black-box approach to automatic optimization. The generality can also help us investigate the association of some meta-parameters with the machine translation system's performance. The same initial values are used for both single-objective and multiple-objective experiments.

In order to apply CMA-ES, we first need to encode the meta-parameters into a fixed-dimensional gene vector. Depending on the domain and possible values of each meta-parameter, a mapping from a real number to a desired domain is needed to translate the gene value to the actual configuration. For the meta-parameters BPE merge operations (bpe_op_src, bpe_op_trg), word embedding dimension (dim_word) and LSTM dimensions (dim_lstm), we used $int(exp(x))$ since they may be large positive integers and $exp(x)$ can represent a large number with a small exponent. For the other meta-parameters such as dropout, alignment regularization, and learning rate, which are positive but small, we used $abs()$ to ensure they are positive as the genes sampled from Gaussian distribution might be negative. There were 10 meta-parameters to tune so the dimension of gene vector was 10, for both single and multi-objective experiment.

4.3. Details of the CMA-ES Setup

Experiments were performed using the TSUBAME 2.5 supercomputer that equips with NVIDIA K20X GPGPU's[6]. We have conducted 10 CMA-ES generations for single-objective experiment and 5 generations for multi-objective experiment. Each generation consisted of 30 individuals for both single and multi-objective optimization. In the single-objective experiment, the training time was limited to a maximum of 48 hours for each generation; in multi-objective experiment, the training time was a maximum of 36 hours. We limited the maximum training time for computational reasons: we found that sometimes the training process of an model may take a week until convergence, but in practice the BLEU scores are not very different from the model at 36+ hours. (See Figure 3 for an example). We think that in CMA-ES, high precision estimates of the final BLEU or time values at convergence are not necessary. It is more efficient to run more generations of CMA-ES, as opposed to spending a long time to obtain the most precise estimate of a gene's BLEU/time rank. The 36 or 48 hours limit on training time is a practical tradeoff.

The experimental process is shown in Figure 4. After sampling genes from the Gaussian search distribution, genes will be converted into meta-parameter configurations. Then the model will be trained using the training set for up to 36 or 48 hours. We call each set of configurations an individual or gene, interchangeably. All individuals of one generation are executed in parallel. After training, the models are used to translate the dev set, and BLEU scores and computation time scores are collected. We rank all individuals based on

[4] http://alaginrc.nict.go.jp/WikiCorpus/index_E.html

[5] http://www.phontron.com/kytea/

[6] http://www.gsic.titech.ac.jp/en

Table 2: Meta-parameters tuned in this study. The initial values are the baseline settings obtained by manual tuning, and is the first individual seeded in CMA-ES. Example results of single and multiple objective evolution are shown: (a) is the individual with maximum dev BLEU of single-objective evolution, achieved at generation 8; see Figure 5. (b) is the individual with minimum computation time in multi-objective evolution's final generation, (c) is the individual with the maximum dev BLEU in multi-objective evolution, achieved at generation 3, and (d) is another individual on the Pareto frontier, achieved at generation 2. Note that (b), (c), and (d) are three of the five points on the Pareto frontier in Figure 6. All of them are considered "optimal" in the multi-objective sense and the single model to deploy in practice should be the human designer's decision.

Meta-parameter	Initial value	(a) Single objective	(b) Multiple objective	(c) Multiple objective	(d) Multiple objective
# BPE merge operations on Source (bpe_op_src)	5000	5250	5345	5011	5102
# BPE merge operations on Target (bpe_op_trg)	5000	6617	4622	5706	5877
# dimension of word embedding (dim_word)	100	121	333	99	104
# of LSTM units (dim_lstm)	400	496	123	459	430
alignment regularization (alpha_c)	0	0.188	0.158	0.249	0.043
learning rate	0.0001	0.100	0.213	0.295	0.083
dropout prob. of embedding (dropout_embedding)	0.2	0.148	0.017	0.147	0.070
dropout prob. of LSTM hidden unit (dropout_hidden)	0.2	0.152	0.036	0.099	0.103
dropout prob. of source words (dropout_source)	0.1	0.026	0.044	0.117	0.013
dropout prob. of target words (dropout_target)	0.1	0.204	0.094	0.019	0.102
dev BLEU	16.48	18.83	17.42	18.04	18.02
dev computation time	248	264	222	269	241

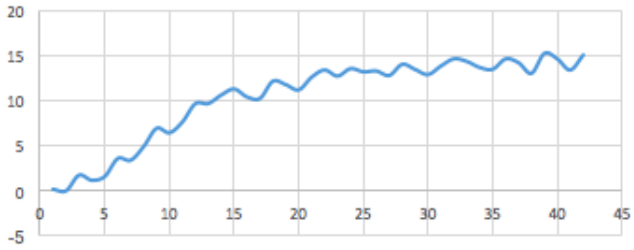


Figure 3: BLEU (y-axis) by number of epoch (x-axis) for an example model/gene.

their scores and update the distribution, via CMA-ES update equations. We then sample new genes and the whole process is repeated for a number of generations until our budget constraint (e.g. 10 generations for single-objective experiment).

5. RESULTS

5.1. Single-objective evolution

For the single-objective evolution experiment, we evaluated a total of $10 \times 30 = 300$ models. For visualization purposes, we choose those individuals with the highest dev BLEU in each generation and plot them on Figure 5. The figure shows how development set BLEU and validation time varies with the number of generation in single-objective evolution that optimizes for development set BLEU.

We observe a general trend of increasing BLEU as evolution progresses. For example, the 8-th generation achieves 18.83 BLEU, the highest among all results, and significantly improves from the baseline of 16.48. There is no guarantee that the improvements are monotonic, however; for example, note that an individual in generation 7 achieves lower BLEU compared to that of generation 6. There is also a slight increase in computation time during the evolution process, which is expected since our single-objective CMA-ES does not account for that objective.

To summarize, the best individual of CMA-ES, achieved at generation 8, has a dev BLEU of 18.83. This outperforms the dev BLEU of our NMT baseline initial setting (16.48) and the KFTT Moses baseline (15.41). In terms of BLEU on the *test set*, this model achieves 16.45, which is an improvement over the NMT baseline initial setting (15.13). So we conclude that CMA-ES has demonstrated its ability to improve upon manually-tuned results. This is done at the expense of considerable computational resources, but the process is entirely automatic and required no human intervention.[7] Meta-parameters of this model is shown in Table 2, column (a).

5.2. Multi-objective evolution

Figure 6 shows a visualization of our multi-objective evolution results, where we evaluated a total of $5 \times 30 = 150$ models. The Pareto optimal models of each generation are plotted. Note that there is a general trend toward individuals moving towards the lower-right hand side of the plot. If we compute the Pareto frontier on all points aggregated in Figure 6, we will obtain 5 points: (18.04 BLEU, 269 seconds), (18.02, 241), (17.42, 222), (16.82, 209), (16.66, 206). The first three of these are shown as examples (b), (c), (d) in Table

[7]However, note that our best NMT test BLEU is still lower than the KFTT Moses baseline test BLEU (17.68). Further work is needed to examine the differences between NMT and PBMT on this dataset.

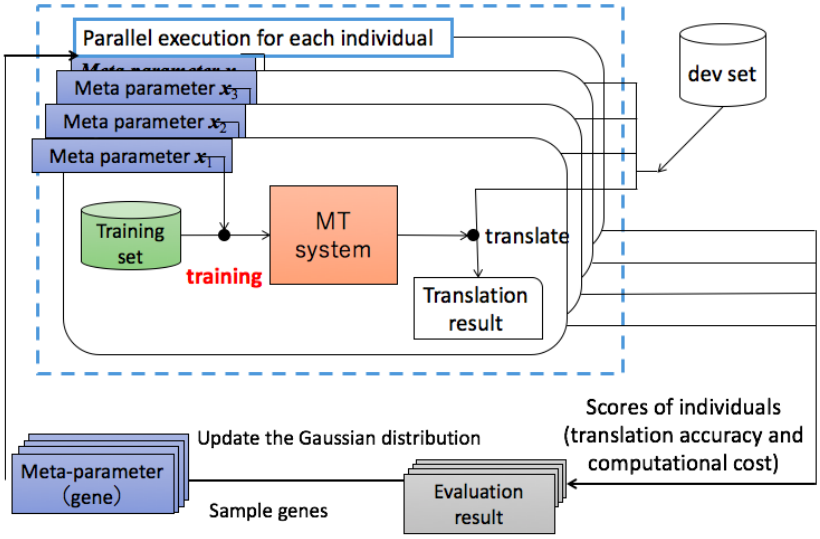


Figure 4: Experimental process of applying CMA-ES to automatically tune NMT models.

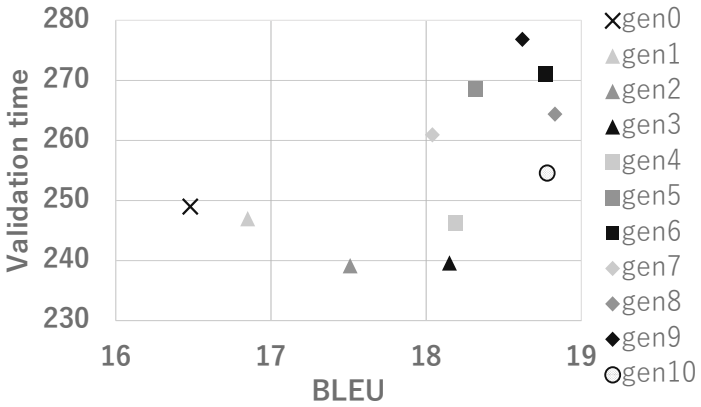


Figure 5: Single-objective evolution results, from generation (gen) 1 to 10. The baseline model with initial value settings is labeled as gen0 and indicated by a cross (x). Note the general improvement of BLEU from the early generations (gen1-3, labeled as triangles) to the later ones (rhombus and circle).

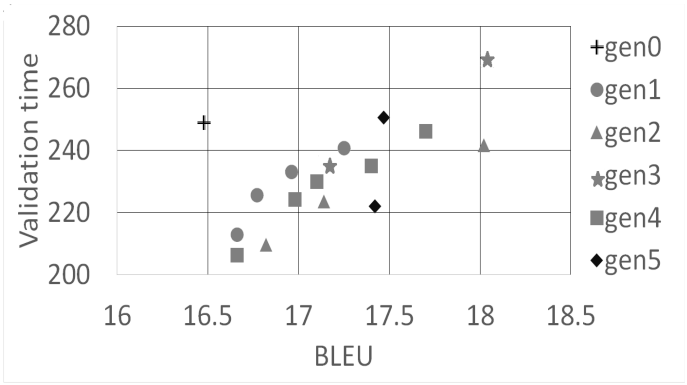


Figure 6: Multi-objective evolution results. The initial model (gen0) is labeled (+), followed by generation 1 models (circles), generation 2 models (triangles), etc.

2. The meta-parameter settings of these Pareto-optimal models are quite distinct. For instance, example (b) has small LSTM units while examples (c) and (d) have larger LSTM units but smaller word embedding dimensions. The target vocabulary (bpe_op_trg) of example (b) is smaller than the initial setting, while those of (c) and (d) are larger; all have larger source target vocabulary.

All the Pareto points in the multi-objective evolution results outperform the baseline initial setting in terms of dev BLEU; some of them outperform the baseline in both BLEU and computation time. Therefore we conclude that the Pareto extension to CMA-ES is achieving its expected effect. There are no improvements in terms of BLEU over the single-objective CMA-ES setting, however. One reason might be that the computation resources used for the multi-objective experiment is less than that of the single-objective experiment. In any case, ideally multi-objective optimization will subsume the single-objective case, and we plan to investigate this further in future work.

6. Analysis

While the results are promising, we want to analyze the statistics of our experiments in order to improve the efficiency of CMA-ES for future work. Figure 7 plots the distribution of various meta-parameters computed across the 300 and 150 models in single- and multi-objective experiments. We note the distribution of word and LSTM dimensions has much wider variance in the multi-objective case compared to the single-objective case (Figure 7 (d) vs (c)), which is expected. Interesting, the range of BPE merge operations (and

thus, the final vocabulary size) is relatively small for both cases (Figure 7 (b) vs (a)). We hypothesize there needs to be some more aggressive (or diverse) sampling in order to fully explore the meta-parameter space. We also think our mapping function that converts real numbers from the CMA-ES Gaussian sample to training configurations may require some re-design: for example, the range of $int(exp())$ may be too narrow, and the use of $abs()$ may induce symmetric properties and confound positive and negative values.

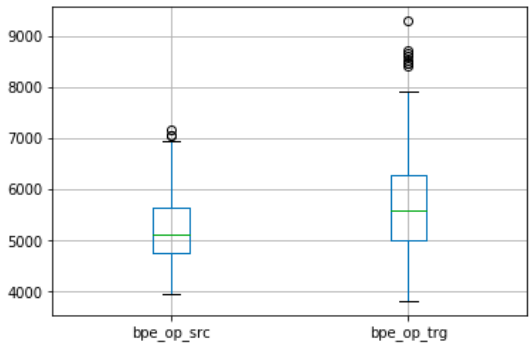
7. Conclusion & Related Work

We demonstrate that an evolution strategy like CMA-ES can be use to automate the tuning of neural network based machine translation system. We start with an initial manually-tuned NMT baseline on KFTT, and show that our our single-objective and multi-objective CMA-ES method can create models that perform better in BLEU and/or computation time.

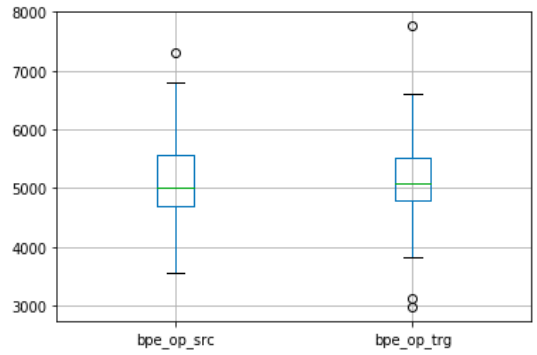
There is a large literature on blackbox optimization, with many successes in practical problems that are difficult to characterize. The main approaches include evolutionary methods (GA or ES) [1, 2] and Bayesian optimization [3, 16]. Recently, in the context of automatic tuning of neural network systems, reinforcement learning [17] and a bandit learning [18] approaches have been proposed. Each approach has its strengths: Evolutionary strategies are efficiently parallelizable. Bayesian optimization models uncertainty in a principled fashion. Reinforcement learning captures sequential dependencies among hyperparameters. Bandit learning provides a framework for trading-off computational resources. In future work, it will be interesting to compare these different approaches on a wider array of datasets.

Pareto optimality has been applied to statistical MT in the context of optimizing multiple evaluation metrics such as BLEU and TER [19, 20]. We are not aware of previous work that performs multi-objective optimization on BLEU and computation time.

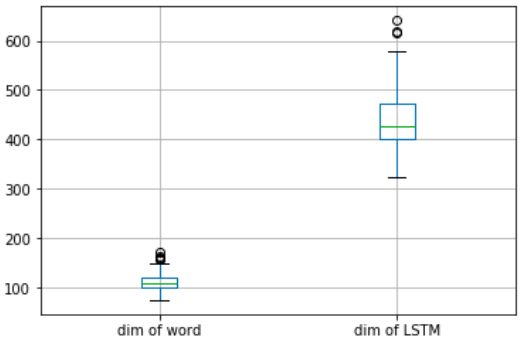
For automatic tuning of neural networks, evolutionary strategies have demonstrated strong results in image classification [21], acoustic modeling [6], and language modeling [22], among others. In NMT, a grid search of meta-parameters is performed in [23]. They used a total of more than 250,000 GPU hours to explore common variations in NMT architectures. Their conclusions include: (a) deep encoders are more difficult to optimize than decoders, (b) dense residual connections are good, (c) LSTMs outperform GRUs. Our work investigates different meta-parameters; it will be interesting to validate their findings with CMA-ES.



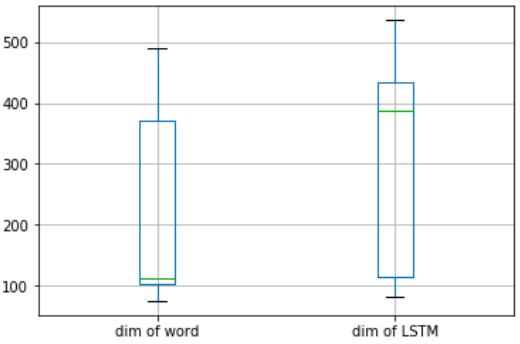
(a) Single objective, BPE



(b) Multi objective, BPE



(c) Single objective, dimensions



(d) Multi objective, dimensions

Figure 7: Boxplot showing the distributions of meta-parameters searched by single-objective and multi-objective CMA-ES.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Numbers JP26280055 and JP17K20001.

8. References

[1] L. Davis, Ed., *Handbook of genetic algorithms*. Van Nostrand Reinhold New York, 1991, vol. 115.

[2] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 949–980, 2014.

[3] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithm s," in *Advances in Neural Information Processing Systems 25*, 2012.

[4] N. Hansen, "The CMA evolution strategy: a comparing review," in *Towards a new evolutionary computation*. Springer, 2006, pp. 75–102.

[5] S. Rostami and A. Shenfield, "CMA-PAES: Pareto archived evolution strategy using covariance matrix adaptation for multi-objective optimisation," in *2012 12th UK Workshop on Computational Intelligence (UKCI)*, Sept 2012, pp. 1–8.

[6] T. Moriya, T. Tanaka, T. Shinozaki, S. Watanabe, and K. Duh, "Automation of system building for state-of-the-art large vocabulary speech recognition using evolution strategy," in *Proceedings of the IEEE 2015 Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2015.

[7] R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A. V. Miceli Barone, J. Mokry, and M. Nadejde, "Nematus: a toolkit for neural machine translation," in *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain: Association for Computational Linguistics, April 2017, pp. 65–68. [Online]. Available: http://aclweb.org/anthology/E17-3017

[8] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[9] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer, 1998.

[10] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, 2004.

[11] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[12] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translations," in *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2014*, ser. cs.CL, no. 1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[13] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 1715–1725. [Online]. Available: http://www.aclweb.org/anthology/P16-1162.pdf

[14] H. Isozaki, K. Sudoh, H. Tsukada, and K. Duh, "Head finalization: A simple reordering rule for sov languages," in *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 244–251. [Online]. Available: http://www.aclweb.org/anthology/W10-1736

[15] G. Neubig, T. Watanabe, and S. Mori, "Inducing a discriminative parser to optimize machine translation reordering," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea: Association for Computational Linguistics, July 2012, pp. 843–853. [Online]. Available: http://www.aclweb.org/anthology/D12-1077

[16] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, p. 28, 12/2015 2016.

[17] B. Zoph and Q. Le, "Neural architecture search with reinforcement learning," in *Proceedings of the International Conference on Representation Learning (ICLR)*, 2017.

[18] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[19] K. Duh, K. Sudoh, X. Wu, H. Tsukada, and M. Nagata, "Learning to translate with multiple objectives," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2012.

[20] B. Sankaran, A. Sarkar, and K. Duh, "Multi-metric optimization using ensemble tuning," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 947–957. [Online]. Available: http://www.aclweb.org/anthology/N13-1115

[21] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. [Online]. Available: http://arxiv.org/abs/1703.01041

[22] T. Tanaka, T. Moriya, T. Shinozaki, S. Watanabe, T. Hori, and K. Duh, "Automated structure discovery and parameter tuning of neural network language model based on evolution strategy," in *Proceedings of the 2016 IEEE Workshop on Spoken Language Technology*, 2016.

[23] D. Britz, A. Goldie, M.-T. Luong, and Q. V. Le, "Massive exploration of neural machine translation architectures," *CoRR*, vol. abs/1703.03906, 2017.