Multi-objective Hyperparameter Search for Fast and Accurate Neural Machine Translation

PI: Kevin Duh, Johns Hopkins University

September 15, 2017

Neural machine translation (NMT) systems are difficult to train. Although they achieve impressive results, non-trivial amounts of effort are required for selecting appropriate hyperparameters, such as number of layers, vocabulary size, non-linear unit type, embedding dimensionality, optimization algorithm, and learning rate.[1] This trial-and-error process is necessary for each task, domain, or language-pair. Further, the rapid development of new neural network architectures implies that this hyperparameter search process will only become more expensive. **The goal of the proposed research is to explore automatic methods for hyperparameter search, with a focus on multi-objective algorithms.** We are interested in deploying models that are not only accurate, but also fast and small.

The hyperparameter search algorithm is given (1) training/validation bitexts, and (2) a budget indicating the maximum number of compute hours available for hyperparameter search. The desired output is a list of hyperparameter settings, together with the resulting NMT models and associated metrics. For example:

Hyperparameter settings	Accuracy (BLEU)	Model Size (MB)	Inference Speed (#words/second)	Pareto optimal?
(a) GRU, 4-layer, 60k vocabulary, ...	0.30	128	1200	**yes**
(b) LSTM, 2-layer, 90k vocabulary, ...	0.25	300	1000	no
(c) LSTM, 8-layer, 70k vocabulary, ...	**0.32**	900	900	**yes**
(d) 8-bit quantized version of model (a)	0.28	**32**	**1300**	**yes**

Table 1: Example output of multi-objective hyperparameter search. The goal is to find hyperparameter settings that lead to the accurate, small, and fast models under a budget constraint. The best result for each metric is boldfaced. Note that model (a), which does not have the best result in any metric, may represent a good tradeoff in practice. Pareto-optimal solutions are candidates to consider for deployment.

The multi-objective aspect of hyperparameter search is important in practice. In the fictitious example above (Table 1), we note that model (c) is best in terms of accuracy and model (d) is best in terms of size and speed. These are definitely candidates for deployment. Model (a) is also a good candidate: although it is not the best in any metric, it represents a well-rounded tradeoff that is reasonably accurate, reasonably small, and reasonably fast. On the other hand, model (b) can be ignored because it is worse than model (a) in every metric. This can be formalized by the concept of *Pareto optimality* [1, 2].

Definition: Let $\boldsymbol{x}$ be a hyperparameter setting. Assume that we wish to maximize J objectives $F(\boldsymbol{x}) \triangleq [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_J(\boldsymbol{x})]$ with respect to $\boldsymbol{x}$ jointly.[2] 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, then we say that $\boldsymbol{x}_k$ *dominates* $\boldsymbol{x}_{k'}$ and write $F(\boldsymbol{x}_k) \triangleright F(\boldsymbol{x}_{k'})$. Given a set of candidate solutions, $\boldsymbol{x}_k$ is *Pareto-optimal* iff no other $\boldsymbol{x}_{k'}$ exists such that $F(\boldsymbol{x}_{k'}) \triangleright F(\boldsymbol{x}_k)$.

For any multi-objective problem, there are generally multiple Pareto-optimal solutions (known as the *Pareto Frontier*). Our goal is to devise algorithms that will efficiently find solutions on the Pareto Frontier. Given these solutions (i.e. NMT models and their associated hyperparameters), the practitioner can then decide manually which model to deploy.

[1] For example, Amazon's feature-rich NMT toolkit Sockeye has at least 40 command-line options related to model architecture and 30 options related to the training algorithm (as of Sept. 13, 2017).

[2] Objectives that we wish to minimize (e.g. model size) can be reformulated by multiplying by -1 so that it can be maximized.

Concretely, we propose:

1. To develop new multi-objective hyperparameter search algorithms, geared towards NMT specifically and deep learning generally.
2. To provide best practices in NMT hyperparameter settings and better understanding of deep learning models via extensive benchmark results.

1 First Challenge: Methods for Practical Hyperparameter Search

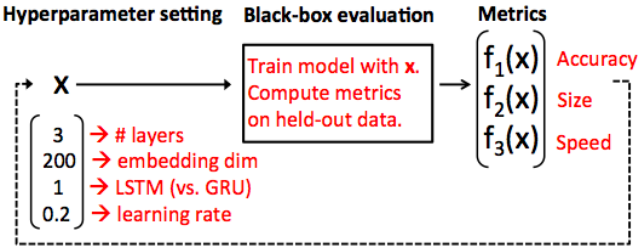

Figure 1: Hyperparameter search viewed as a black-box optimization problem.

Hyperparameter search can be formulated as a **black-box optimization problem** (Figure 1). Suppose we have d hyperparameters to search over. The hyperparameter setting for training a model is encoded as a vector $\boldsymbol{x} \in \mathcal{R}^d$.[3] In the single-objective case, the goal is to find the $\boldsymbol{x}$ that maximizes $f_1(\boldsymbol{x})$. In the multi-objective case, the goal is to find the set of $\boldsymbol{x}$ which form the Pareto frontier of $F(\boldsymbol{x}) \triangleq [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x})]$.

Given a particular $\boldsymbol{x_k}$, the black-box instantiates a neural network model according to the hyperparameter setting, runs training to convergence, and reports the metrics (accuracy, model size, inference speed) computed on a held-out set. The algorithm then decides what hyperparameter $\boldsymbol{x_{k+1}}$ to search next, using information from past $\boldsymbol{x_k}$ and $F(\boldsymbol{x_k})$. This is difficult because there is no gradient information, though several classes of algorithms have been developed (mainly for the single-objective case):

- Bayesian optimization [3, 4]: directly model the function $f_1(\cdot)$ with a Gaussian Process. Point estimates $f_1(\boldsymbol{x_k})$ reduce the uncertainty in certain regions of $\mathcal{R}^d$, and $\boldsymbol{x_{k+1}}$ that maximizes *expected improvement* are searched next.
- Evolutionary methods [5, 6, 7]: use a *population* of $\{\boldsymbol{x_k}\}$, and let $f_1(\boldsymbol{x_k})$ represent each individual solution's *fitness* in the Darwinian sense. Solutions with higher fitness have a higher likelihood of producing offspring $\{\boldsymbol{x_{k+1}}\}$ in the next generation.
- Reinforcement learning [8]: views each of the d hyperparameter choices as actions. The goal is to find a sequence of d actions that leads to high reward $f_1(\boldsymbol{x_k})$.

Each algorithm class has its strengths. Bayesian optimization models uncertainty. Evolutionary methods are efficiently parallelizable. Reinforcement learning captures sequential dependencies among hyperparameters. We do not subscribe to any particular philosophy and will explore whatever seems promising.

The main challenge common to all these algorithms is that the black-box evaluation is expensive. This is the case whether we focus on single or multiple objectives, since the bottleneck is in the time and computational resources needed to train a neural network model to convergence. It may take days or weeks on a GPU node simply to compute one instance of $F(\boldsymbol{x_k})$. If we cannot obtain sufficient number of instances of $\{\boldsymbol{x_k}, F(\boldsymbol{x_k})\}$ under the budget constraint, no hyperparameter search method will be effective in practice. To address this, we propose to *approximate* the black-box evaluation. This will necessarily involve *peeking into the black-box* and devising methods that are tailored to the NMT models.

[3] Discrete hyperparameters (e.g. unit type like LSTM or GRU) may be converted arbitrarily to integer values; some hyperparameters may be depended on others (e.g. learning rate and optimization algorithm type). In general, the representation of hyperparameter settings in a vector in $\mathcal{R}^d$ is an art (or hack).

Proposed Approach: We propose methods to approximate the black-box evaluation step in hyperparameter search (Figure 1). In other words, we seek an approximation $\hat{F}(\boldsymbol{x_k})$ such that $\hat{F}(\boldsymbol{x_k}) \approx F(\boldsymbol{x_k})$. A simple approximation method is to terminate the model training process immediately when the learning curve begins to plateau: the resulting model will likely give accuracies, model sizes, and inference speeds that are similar to the fully-trained version.

Hyperparameter search can be made robust to imprecisions in $\hat{F}(\boldsymbol{x_k})$. The hope is that cheaper computation of $\hat{F}(\boldsymbol{x})$ in the inner-loop can free up computational resources, allowing for more iterations of hyperparameter search in the outer-loop and leading to better solutions in the long run.

We will explore a combination of multiple approximation strategies, including:

1. **Bandit learning:** Suppose we employ a hyperparameter search algorithm that evaluates multiple black-boxes in parallel (e.g. evolutionary method, grid search). Each model $\boldsymbol{x_k}$ can be viewed as an arm in a multi-armed bandit problem, each with unknown reward $F(\boldsymbol{x_k})$. The more we pull on an arm (i.e. the longer we let training run for a particular hyperparameter setting), the more precise our estimate $\hat{F}(\boldsymbol{x_k})$ will be. This formulation enables us to prematurely terminate the training process of any model before convergence, thereby dynamically adjusting the computational resources allocated for each black-box. We will extend the single-objective bandit method in [9] to multiple objectives.

2. **Data and vocabulary selection:** One common strategy for *manual tuning* of hyperparameters is to first perform hyperparameter search on smaller problems, then re-evaluate promising solutions on the original problem. We will devise data selection methods that speed up the training of NMT models. One idea is to use submodularity optimization to find data subsets that reduce the vocabulary size [10], then perform a first-pass hyperparameter search using smaller-vocabulary models. Once promising hyperparameters are found, we run full hyperparameter search allowing for larger vocabulary sizes.

3. **Hyperparameter clustering:** Models with similar architectures can share parameters and training statistics. For example, the parameters of a 2-layer LSTM can be exploited to initialize a 4-layer LSTM, speeding up the training process. Sharing can also be possible across similar learning problems or datasets, i.e. *collaborative hyperparameter tuning* [11]. We will explore ways to cluster the hyperparameters $\boldsymbol{x_k}$, manually or automatically, and integrate sharing into the overall algorithm.

We will re-design multi-objective hyperparameter search algorithms with black-box approximation in mind. The challenge is to devise an effective integration of the various approximation ideas above, which interact with the outer-loop in different ways, into a practical yet effective overall algorithm.

Previous Work: We have had success in extending an evolutionary method called CMA-ES [12] to the multi-objective case. The idea is to incorporate Pareto optimality into the fitness function. This led to improved acoustic models [13] and language models [14] for speech recognition.

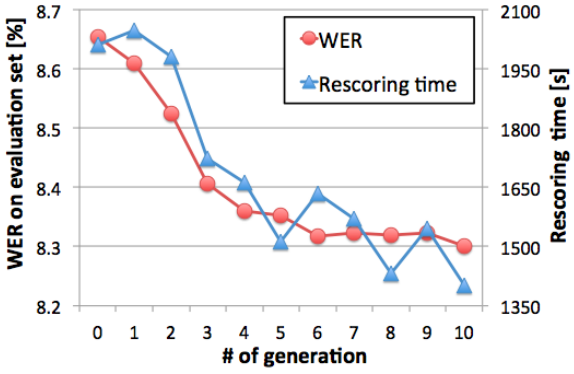

Figure 2: Multi-objective search in action.

Figure 2 demonstrates our algorithm's search process on the task of building neural language models for speech recognition. We have 37 hyperparameters, e.g. choice of feed-forward vs. recurrent nets, #layers, vocabulary size, n-gram interpolation weight. The initial point is a manually-tuned baseline, with a word-error-rate of 8.65% and an inference speed that rescores the evaluation set in 2010 seconds. Both metrics improve at each generation of the evolutionary method (population size is 30); we observe a 4% improvement in accuracy and 30% improvement in speed at the end.

So, we know that multi-objective hyperparameter search can be very useful. The challenge is the computational bottleneck of black-box evaluations. Our goal here is to enable extensive large-scale hyperparameter searches on more tasks, in particular in NMT.

2 Second Challenge: Extensive NMT Benchmarking

We will implement our hyperparameter search algorithm on top of Sockeye, Amazon's feature-rich NMT toolkit based on MxNet (`https://github.com/awslabs/sockeye`). This is an attractive toolkit because it includes many of the recent NMT innovations. We will run hyperparameter search on a wide range of datasets, collect the Pareto-optimal models, and perform data visualization on the resulting metrics.

This will provide a better understanding of NMT models in general. Questions include: What are best practices for choosing among recurrent [15], convolutional [16], or transformer [17] architectures? Do desirable hyperparameter settings transfer across language-pairs? What are the performance bounds we can expect from the best models in low-resource and high-resource settings? What are the tradeoffs between accuracy, size, and speed? We aim to provide comprehensive answers, adding to the initial observations by [18] using single-objective grid search.

In addition, we aim to demonstrate that automatic hyperparameter search can lead to new state-of-the-art results on benchmarks like WMT. This is one measure of success for the project.

The main challenge here is software engineering: both the hyperparameter search and the various black-box approximate methods need to be implemented such that jobs can be scheduled and run efficiently at scale. We will also need to build data visualization tools to fully glean insights from the massive amount of log data generated by hyperparameter search.

3 Budget, Research Plan, Expected Results

A support of $80,000 is appreciated. This will cover partial funding for one Ph.D. student (full funding of tuition+stipend is $86k/year at JHU) and $1,500 in travel expenses, which will be allocated for visits to Amazon (Pittsburgh, Seattle, or Berlin). If possible, I would love to engage with researchers and engineers from e.g. the groups of **Alex Klementiev**, **Alon Lavie**, and **Alex Smola**, to learn more about technical details of Sockeye/MxNet and best practices of large-scale machine learning on AWS. Also, I will continue to have discussions with my long-time collaborators Prof. Takahiro Shinozaki (Tokyo Tech) and Prof. Shinji Watanabe (JHU), who are experts in hyperparameter search in speech recognition.

I also request $20,000 in AWS credits, which I plan to use fully for the experiments. My plan is to implement a software package that runs hyperparameter search and black-box evaluation efficiently on both AWS and my local JHU SGE Linux cluster. The software will be open-sourced, and the hope is that more researchers will contributed to this practical and important research problem.

The research plan is:

Winter 2018	Implement multi-objective hyperparameter search on top of Sockeye NMT
Spring 2018	Participate in WMT benchmark evaluation
Summer 2018	Devise more approximation methods; run large-scale experiments on AWS
Fall 2018	Release code and models; research publication

Finally, expected results include: (1) practical multi-objective hyperparameter search algorithms, (2) the associated open-source software built on top of Sockeye, (3) state-of-the-art Pareto-optimal NMT models, and (4) better understanding of the deep learning models for NMT.

4 Biography & CV

I am assistant research professor at JHU Computer Science and senior research scientist at the HLT Center of Excellence. Previously, I was assistant professor at NAIST in Japan. I received my PhD from the University of Washington in 2009. I was program chair of EMNLP 2016.

Full CV: `http://cs.jhu.edu/~kevinduh/cv.pdf`.

References

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

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

[3] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka," *J. Mach. Learn. Res.*, vol. 18, pp. 826–830, Jan. 2017.

[4] 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, p. 28, 12/2015 2016.

[5] 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.

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

[7] 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.

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

[9] 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.

[10] K. Kirchhoff and J. Bilmes, "Submodularity for data selection in machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 131–141, Association for Computational Linguistics, October 2014.

[11] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag, "Collaborative hyperparameter tuning," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. II–199–II–207, JMLR.org, 2013.

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

[13] 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.

[14] 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.

[15] 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.

[16] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional Sequence to Sequence Learning," *ArXiv e-prints*, May 2017.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

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