Adaptive Scheduling for Multi-Task Learning

Sébastien Jean[*]
Department of Computer Science
New York University
sebastien@cs.nyu.edu

Orhan Firat
Google AI
orhanf@google.com

Melvin Johnson
Google AI
melvinp@google.com

Abstract

To train neural machine translation models simultaneously on multiple tasks (languages), it is common to sample each task uniformly or in proportion to dataset sizes. As these methods offer little control over performance trade-offs, we explore different task scheduling approaches. We first consider existing non-adaptive techniques, then move on to adaptive schedules that over-sample tasks with poorer results compared to their respective baseline. As explicit schedules can be inefficient, especially if one task is highly over-sampled, we also consider implicit schedules, learning to scale learning rates or gradients of individual tasks instead. These techniques allow training multilingual models that perform better for low-resource language pairs (tasks with small amount of data), while minimizing negative effects on high-resource tasks.

1 Introduction

Multiple tasks may often benefit from others by leveraging more available data. For natural language tasks, a simple approach is to pre-train embeddings [15, 16] or a language model [17, 3] over a large corpus. The learnt representations may then be used for upstream tasks such as part-of-speech tagging or parsing, for which there is less annotated data. Alternatively, multiple tasks may be trained simultaneously with either a single model or by sharing some model components. In addition to potentially benefit from multiple data sources, this approach also reduces the memory use. However, multi-task models of similar size as single-task baselines often under-perform because of their limited capacity. The underlying multi-task model learns to improve on harder tasks, but may hit a plateau, while simpler (or data poor) tasks can be over-trained (over-fitted). Regardless of data complexity, some tasks may be forgotten if the schedule is improper, also known as *catastrophic forgetting* [5].

In this paper, we consider multilingual neural machine translation (NMT), where both of the above pathological learning behaviors are observed, sub-optimal accuracy on high-resource, and forgetting on low-resource language pairs. Multilingual NMT models are generally trained by mixing language pairs in a predetermined fashion, such as sampling from each task uniformly [4] or in proportion to dataset sizes [13]. While results are generally acceptable with a fixed schedule, it leaves little control over the performance of each task. We instead consider adaptive schedules that modify the importance of each task based on their validation set performance. The task schedule may be modified explicitly by controlling the probability of each task being sampled. Alternatively, the schedule may be fixed, with the impact of each task controlled by scaling the gradients or the learning rates. In this case, we highlight important subtleties that arise with adaptive learning rate optimizers such as Adam [9]. Our proposed approach improves the low-resource pair accuracy while keeping the high resource accuracy intact within the same multi-task model.

[*]Work done while at Google AI.

Continual Learning Workshop at the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

2 Explicit schedules

A common approach for multi-task learning is to train on each task uniformly [4]. Alternatively, each task may be sampled following a fixed non-uniform schedule, often favoring either a specific task of interest or tasks with larger amounts of data [13, 10]. Kipperwasser and Ballesteros [10] also propose variable schedules that increasingly favor some tasks over time. As all these schedules are pre-defined (as a function of the training step or amount of available training data), they offer limited control over the performance of all tasks. As such, we consider adaptive schedules that vary based on the validation performance of each task during training.

To do so, we assume that the baseline validation performance of each task, if trained individually, is known in advance[2]. When training a multi-task model, validation scores are continually recorded in order to adjust task sampling probabilities. The unnormalized score w_i of task i is given by

$$w_i = 1 \Bigg/ \left(\min\left(1, \frac{s_i}{b_i}\right)^{\alpha} + \epsilon \right) \tag{1}$$

where s_i is the latest validation BLEU score and b_i is the (approximate) baseline performance. Tasks that perform poorly relative to their baseline will be over-sampled, and vice-versa for language pairs with good performance. The hyper-parameter α controls how agressive oversampling is, while ϵ prevents numerical errors and slightly smooths out the distribution. Final probabilities are simply obtained by dividing the raw scores by their sum.

3 Implicit schedules

Explicit schedules may possibly be too restrictive in some circumstances, such as models trained on a very high number of tasks, or when one task is sampled much more often than others. Instead of explicitly varying task schedules, a similar impact may be achieved through learning rate or gradient manipulation. For example, the GradNorm [2] algorithm scales task gradients based on the magnitude of the gradients as well as on the training losses.

As the training loss is not always a good proxy for validation and test performance, especially compared to a single-task baseline, we continue using validation set performance to guide gradient scaling factors. Here, instead of the previous weighting schemes, we consider one that satisfies the following desiderata. In addition to favoring tasks with low relative validation performance, we specify that task weights are close to uniform early on, when performance is still low on all tasks. We also as set a minimum task weight to avoid catastrophic forgetting.

Task weights $w_i, i = 1, ..., N$, follow

$$w_i = 1 + (\text{sign}\,(\overline{S} - S_i)) \min\left(\gamma, \left(\max_j S_j\right)^{\alpha} \left|S_i - \overline{S}\right|^{\beta}\right), \tag{2}$$

where $S_i = \frac{s_i}{b_i}$ and $\overline{S}$ is the average relative score $(\sum_{j=1}^{N} S_j)/N$. γ sets the floor to prevent catastrophic forgetting, α adjusts how quickly and strongly the schedule may deviate from uniform, while a small β emphasizes deviations from the mean score. With two tasks, the task weights already sum up to two, as in GradNorm [2]. With more tasks, the weights may be adjusted so their their sum matches the number of tasks.

3.1 Optimization details

Scaling either the gradients g_t or the per-task learning rates α is equivalent with standard stochastic gradient descent, but not with adaptive optimizers such as Adam [9], whose update rule is given in Eq. 3.

$$\begin{aligned} \hat{m}_t &= (\beta_1 m_{t-1} + (1-\beta_1) g_t)/(1-\beta_1^t) \\ \hat{v}_t &= (\beta_2 v_{t-1} + (1-\beta_2) g_t^2)/(1-\beta_2^t) \\ \theta_t &= \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) \end{aligned} \tag{3}$$

[2]Baseline scores can be obtained from already trained single task models, or can be set to an expected value to be reached by the multi-task model.

Moreover, sharing or not the optimizer accumulators (eg. running average of 1st and 2nd moment $\hat{m}_t$ and $\hat{v}_t$ of the gradients) is also impactful. Using separate optimizers and simultaneously scaling the gradients of individual tasks is ineffective. Indeed, Adam is scale-insensitive because the updates are divided by the square root of the second moment estimate $\hat{v}_t$. The opposite scenario, a shared optimizer across tasks with scaled learning rates, is also problematic as the momentum effect ($\hat{m}_t$) will blur all tasks together at every update. All experiments we present use distinct optimizers, with scaled learning rates. The converse, a shared optimizer with scaled gradients, could also potentially be employed.

4 Experiments

4.1 Data

We extract data from the WMT'14 English-French (En-Fr) and English-German (En-De) datasets. To create a larger discrepancy between the tasks, so that there is a clear dataset size imbalance, the En-De data is artificially restricted to only 1 million parallel sentences, while the full En-Fr dataset, comprising almost 40 million parallel sentences, is used entirely. Words are split into subwords units with a joint vocabulary of 32K tokens.[3] BLEU scores are computed on the tokenized output with *multi-bleu.perl* from Moses [11].

4.2 Models

All baselines are Transformer models in their base configuration [19], using 6 encoder and decoder layers, with model and hidden dimensions of 512 and 2048 respectively, and 8 heads for all attention layers. For initial multi-task experiments, all model parameters were shared [7], but performance was down by multiple BLEU points compared to the baselines. As the source language pair is the same for both tasks, in subsequent experiments, only the encoder is shared [4]. For En-Fr, 10% dropout is applied as in [19]. After observing severe overfitting on En-De in early experiments, the rate is increased to 25% for this lower-resource task. All models are trained on 16 GPUs, using Adam optimizer with a learning rate schedule (inverse square root [19]) and warmup.

4.3 Results

The main results are summarized in Table 1. Considering the amount of training data, we trained single task baselines for 400K and 600K steps for En-De and En-Fr respectively, where multi-task models are trained for 900K steps after training. All reported scores are the average of the last 20 checkpoints. Within each general schedule type, model selection was performed by maximizing the average development BLEU score between the two tasks.

With uniform sampling, results improve by more than 1 BLEU point on En-De, but there is a significant degradation on En-Fr. Sampling En-Fr with a 75% probability gives similar results on En-De, but the En-Fr performance is now comparable to the baseline. Explicit adaptive scheduling behaves similarly on En-De and somewhat trails the En-Fr baseline.

Method	Task 1 (En-De)		Task 2 (En-Fr)	
	Dev	Test	Dev	Test
En-De Baseline	23.58	24.90	-	-
En-Fr Baseline	-	-	**34.71**	40.80
Explicit - Constant (50% En-Fr)	**24.80**	26.14	34.25	39.98
Explicit - Constant (75% En-Fr)	24.53	26.16	34.56	**41.00**
Explicit - Validation based	24.67	26.35	34.55	40.70
Implicit - GradNorm	24.69	**26.42**	34.33	40.28
Implicit - Validation based	24.32	25.58	34.67	40.89

Table 1: Comparison of scheduling methods, measured by BLEU scores. Best results in bold.

[3] Joint vocabulary is extracted from the full En-De and En-Fr datasets.

For implicit schedules, GradNorm performs reasonably strongly on En-De, but suffers on En-Fr, although slightly less than with uniform sampling. Implicit validation-based scheduling still improves upon the En-De baseline, but less than the other approaches. On En-Fr, this approach performs about as well as the baseline and the multilingual model with a fixed 75% En-Fr sampling probability.

Overall, adaptive approaches satisfy our desiderata, satisfactory performance on both tasks, but an hyper-parameter search over constant schedules led to slightly better results. One main appeal of adaptive models is their potential ability to scale much better to a very large number of tasks, where a large hyper-parameter search would prove prohibitively expensive.

Additional results are presented in the appendix.

5 Discussion and other related work

To train multi-task vision models, Liu et al. [12] propose a similar *dynamic weight average* approach. Task weights are controlled by the ratio between a recent training loss and the loss at a previous time step, so that tasks that progress faster will be downweighted, while straggling ones will be upweighted. This approach contrasts with the curriculum learning framework proposed by Matiisen et al. [14], where tasks with faster progress are preferred. Loss progress, and well as a few other signals, were also employed by Graves et al. [6], which formulated curriculum learning as a multi-armed bandit problem. One advantage of using progress as a signal is that the final baseline losses are not needed. *Dynamic weight average* could also be adapted to employ a validation metric as opposed to the training loss. Alternatively, uncertainty may be used to adjust multi-task weights [8].

Sener and Volkun [18] discuss multi-task learning as a multi-objective optimization. Their objective tries to achieve Pareto optimality, so that a solution to a multi-task problem cannot improve on one task without hurting another. Their approach is learning-based, and contrarily to ours, doesn't require a somewhat ad-hoc mapping between task performance (or progress) and task weights. However, Pareto optimality of the training losses does not guarantee Pareto optimality of the evaluation metrics. Xu et al. present *AutoLoss* [20], which uses reinforcement learning to train a controller that determines the optimization schedule. In particular, they apply their framework to (single language pair) NMT with auxiliary tasks.

With implicit scheduling approaches, the effective learning rates are still dominated by the underlying predefined learning rate schedule. For single tasks, hypergradient descent [1] adjusts the global learning rate by considering the direction of the gradient and of the previous update. This technique could likely be adapted for multi-task learning, as long as the tasks are sampled randomly.

Tangentially, adaptive approaches may behave poorly if validation performance varies much faster than the rate at which it is computed. Figure 6 (appendix) illustrates a scenario, with an alternative parameter sharing scheme, where BLEU scores and task probabilities oscillate wildly. As one task is favored, the other is catastrophically forgotten. When new validation scores are computed, the sampling weights change drastically, and the first task now begins to be forgotten.

6 Conclusion

We have presented adaptive schedules for multilingual machine translation, where task weights are controlled by validation BLEU scores. The schedules may either be explicit, directly changing how task are sampled, or implicit by adjusting the optimization process. Compared to single-task baselines, performance improved on the low-resource En-De task and was comparable on high-resource En-Fr task.

For future work, in order to increase the utility of adaptive schedulers, it would be beneficial to explore their use on a much larger number of simultaneous tasks. In this scenario, they may prove more useful as hyper-parameter search over fixed schedules would become cumbersome.

References

[1] Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*, 2017.

[2] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257*, 2017.

[3] 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.

[4] Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1723–1732, 2015.

[5] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128 – 135, 1999.

[6] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*, 2017.

[7] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association of Computational Linguistics*, 5(1):339–351, 2017.

[8] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics.

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

[10] Eliyahu Kiperwasser and Miguel Ballesteros. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240, 2018.

[11] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics, 2007.

[12] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. *arXiv preprint arXiv:1803.10704*, 2018.

[13] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[14] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *arXiv preprint arXiv:1707.00183*, 2017.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[16] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[17] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

[18] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *arXiv preprint arXiv:1810.04650*, 2018.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[20] Haowen Xu, Hao Zhang, Zhiting Hu, Xiaodan Liang, Ruslan Salakhutdinov, and Eric Xing. Autoloss: Learning discrete schedules for alternate optimization. *arXiv preprint arXiv:1810.02442*, 2018.

A Impact of hyper-parameters

In this appendix, we present the impact of various hyper-parameters for the different schedule types.

Figure 1 illustrates the effect of sampling ratios in explicit constant scheduling. We vary the sampling ratio for a task from 10% to 90% and evaluated the development and test BLEU scores by using this fixed schedule throughout the training. Considering the disproportional dataset sizes between two tasks (1/40), oversampling high-resource task yields better overall performance for both tasks. While a uniform sampling ratio favors the low-resource task (50%-50%), more balanced results are obtained with a 75% - 25% split favoring the high-resource task.

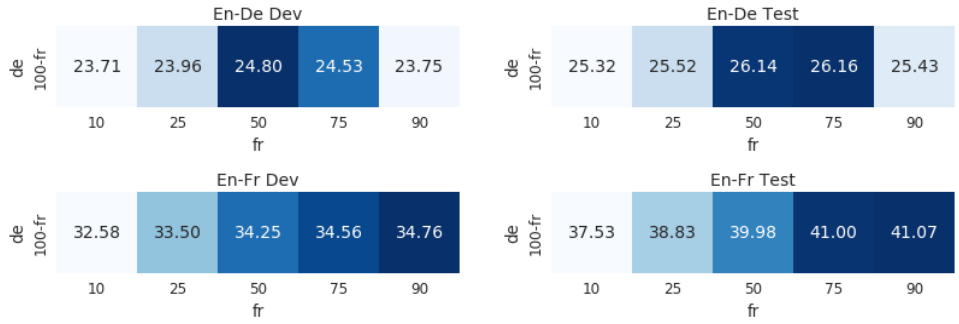

Figure 1: BLEU Score Results for Explicit Constant Schedules. Higher scores are color coded with darker colors and indicate better accuracy.

Explicit Dev-Based schedule results are illustrated in Figure 2 below, where we explored varying α and ϵ parameters, to control oversampling and forgetting.

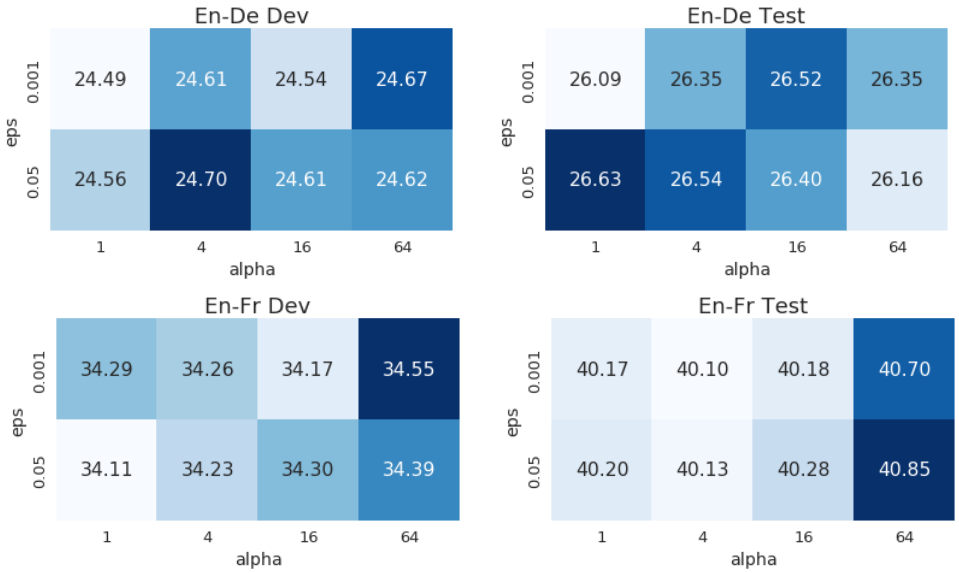

Figure 2: Explicit Dev-Based Schedules

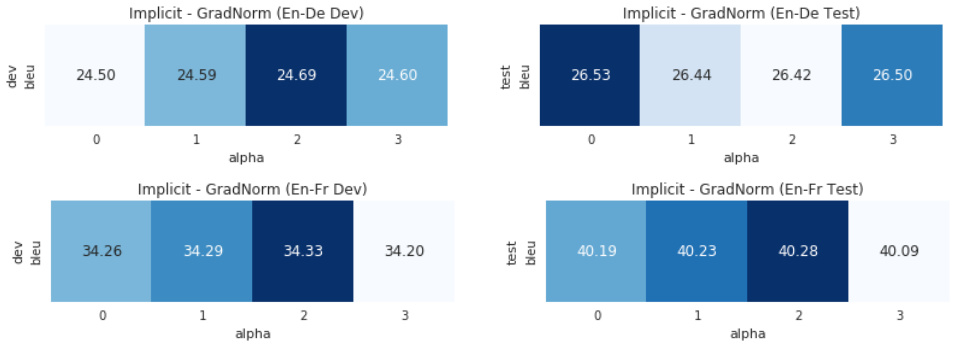

Figure 3: Implicit GradNorm Schedules

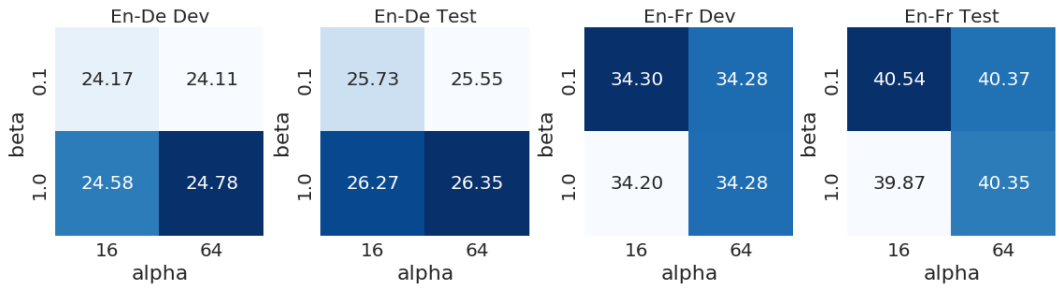

Figure 4: Implicit Dev-Based Schedules

B Implicit validation-based scheduling progress

We here present how the task weights, learning rates and validation BLEU scores are modified over time with an implicit schedule. For the implicit schedule hyper-parameters, we set $\alpha = 16$, $\beta = 0.1$, $\gamma = 0.05$ with baselines b_i being 24 and 35 for En-De and En-Fr respectively. For the best performing model, we used inverse-square root learning rate schedule [19] with a learning rate of 1.5 and 40K warm-up steps.

Task weights are adaptively changed by the scheduler during training (Figure 5 top-left), and predicted weights are used to adjust the learning rates for each task (Figure 5 top-right). Following Eq. 2, computed relative scores for each task, S_j, are illustrated in Figure 5 bottom-left. Finally, progression of the validation set BLEU scores with their corresponding baselines (as solid horizontal lines) are given in in Figure 5 bottom-right.

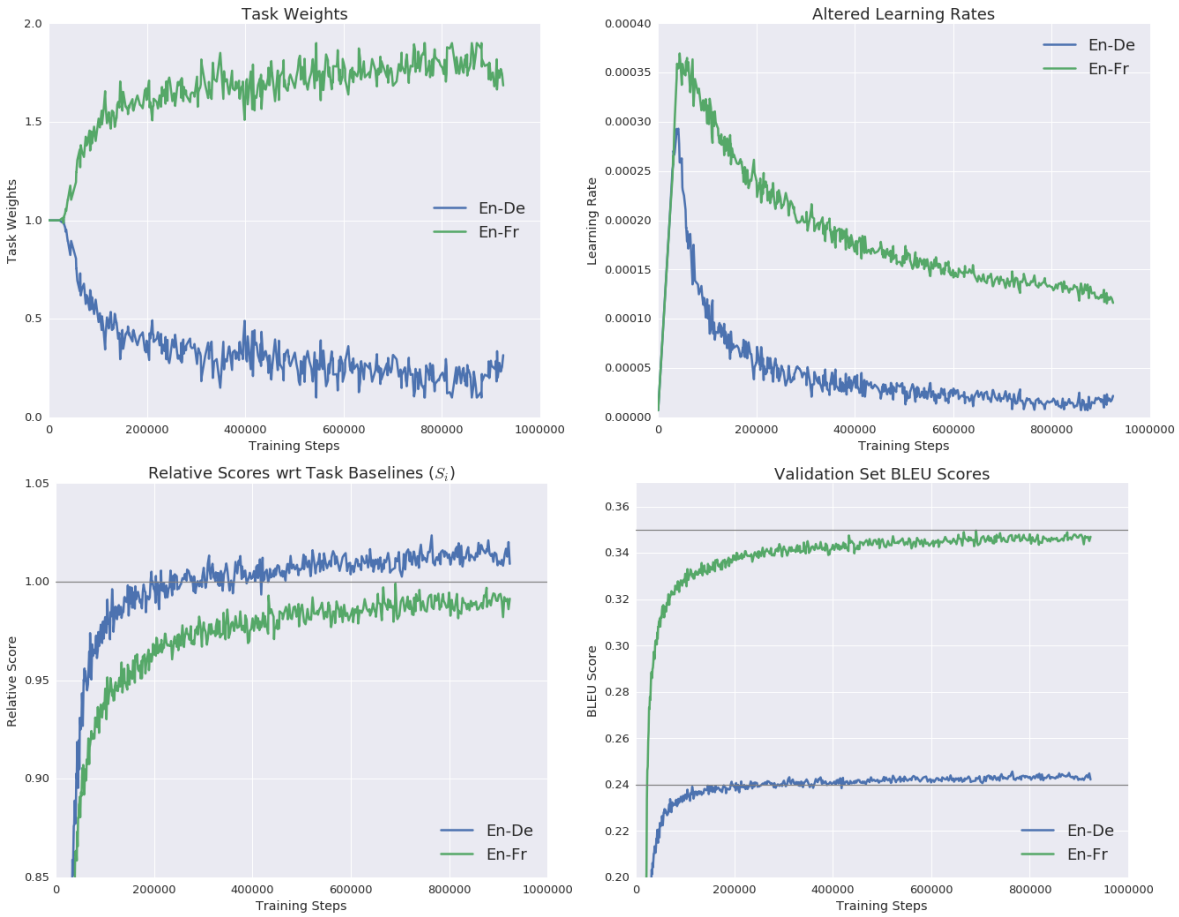

Figure 5: Implicit Validation-Based Scheduling Progress.

C Possible training instabilities

This appendix presents a failed experiment with wildly varying oscillations. All encoder parameters were tied, as well as the first four layers of the decoder and the softmax. An explicit schedule was employed.

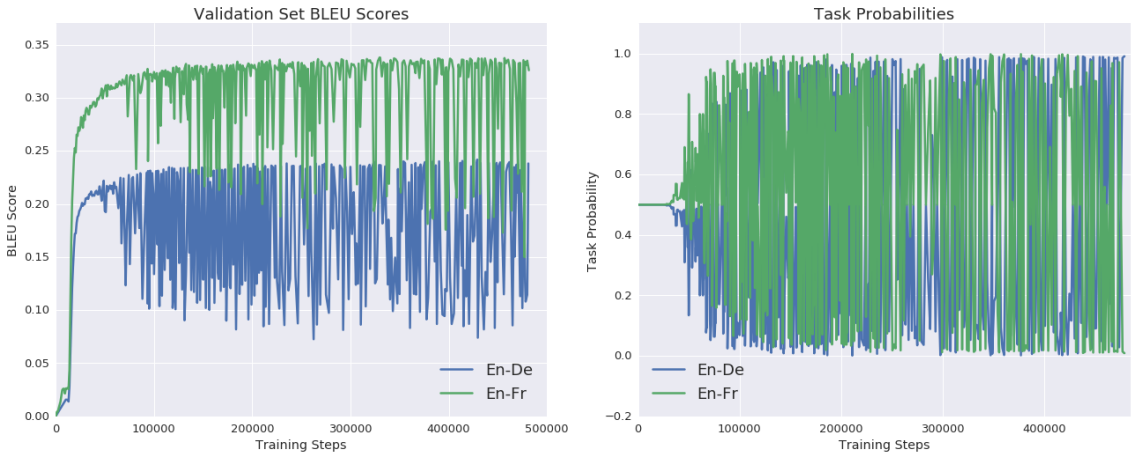

Figure 6: Wild oscillations