Learning Task Sampling Policy for Multitask Learning

Dhanasekar Sundararaman[1*], **Henry Tsai**[2], **Kuang-Huei Lee**[2],
Iulia Turc[2], **Lawrence Carin**[1]
[1] Duke University
[2] Google Research
dhanasekar.sundararaman@duke.edu

Abstract

It has been shown that training multi-task models with auxiliary tasks can improve the target tasks quality through cross-task transfer. However, the importance of each auxiliary task to the primary task is likely not known a priori. While the importance weights of auxiliary tasks can be manually tuned, it becomes practically infeasible with the number of tasks scaling up. To address this, we propose a search method that automatically assigns importance weights. We formulate it as a reinforcement learning problem and learn a task sampling schedule based on evaluation accuracy of the multi-task model. Our empirical evaluation on XNLI and GLUE shows that our method outperforms uniform sampling and the corresponding single-task baseline.

1 Introduction

Multi-task learning (Caruana, 1997) has been shown to improve performance of multiple related tasks through cross-task knowledge transfer using just one model, which also drastically improves parameter efficiency (Hashimoto et al., 2016; Kaiser et al., 2017). In the case where the objective is improving specific target tasks, the use of auxiliaries has demonstrated substantial benefits, for example, in sequence-to-sequence learning (Luong et al., 2015), and question answering (McCann et al., 2018).

In this paper, we follow this line and consider the goal of training a multi-task model to be the maximizing performance of one target task. While there are numerous efforts that attempted to increase the complexity of multi-task models to match the performance through architectural addition, not a lot of emphasis has been placed on harnessing the potential of a model through exploration of task sampling weights, as we view weighted combination of different tasks as a sampling problem.

In our work, we formulate this exploration problem as learning a policy to assign task sampling weights deferentially over time. For example, an optimal policy could be putting more weights on auxiliary tasks with rich data at the beginning of training and shifting to the target tasks near the end for finetuning. While manual hyper-parameter fine-tuning should help find a good policy, it becomes challenging as the number of combinations increases exponentially with respect to the number of tasks. To solve this problem, we propose a reinforcement-learning-based search method that automatically finds the optimal policy to maximize the target task accuracy. Different from previous dynamic re-weighting works (Guo et al., 2019; Wang et al., 2020), our policy is static and agnostic to model training artifacts such as which training examples are selected and network parameters, and thus the policy can also be viewed as a schedule. This property enables reuse of a fixed sampling schedule once search is done. When we need to retrain model due to system change or minor underlying model change, it allows us to directly retrain without performing policy search again.

We conduct experiments by targeting at improving one language's performance in the XNLI (Conneau et al., 2018) dataset or one task's performance in the GLUE dataset. We empirically demonstrate that multi-task models trained with the proposed method outperform multi-task models learned with various rule-based sampling as well as corresponding single-task baselines. We also conduct necessary ablation study to validate our approach. Our main contributions are as follows. First, we are the first work to formulate multi-task exploration as a static sampling policy/schedule learning problem to the best of our knowledge. Then, we propose a simple and effective policy optimization algorithm for learning task sampling policy.

[*] Work done during an internship at Google Research

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 4410–4415
November 7–11, 2021.

2 Related Works

Existing works (Collobert and Weston, 2008; Luong et al., 2015) show that, by doing proper architecture changes, we can utilize cross-task transfer of multi-task learning and have multi-task models outperform single-task baselines. However, for different multi-task settings, the optimal architectures may be different and difficult to know a priori. It leads us to prefer an automatic searching approach, e.g. RL-based architecture search methods (Ma et al., 2019) were proposed to learn how to share under different settings.

While there are many successes in searching architecture for multi-task learning, how to select tasks to train together is underexplored. The current approaches are somehow arbitrary. With a goal of optimizing the performance for one target task, some works (Bingel and Søgaard, 2017; Platanios et al., 2019; Vu et al., 2020) exhaustively explored the relatedness of tasks in a multi-task training setting by manually picking pair of tasks and found that the multi-task performance gains significantly. This becomes practically infeasible as the number of tasks grow making the possible combinations of tasks exponential. Up-sampling strategy (Johnson et al., 2017; Conneau et al., 2018), such as uniform sampling that balances high-resource and low-resource tasks, is another dimension that has been explored. Finally, the sampling strategy can also change over time. Curriculum learning (Platanios et al., 2019) can be used to choose to learn hard tasks first and easy tasks later. Domain adaption by continuous pre-training on the target domain (Gururangan et al., 2020) can be seen as a schedule that first pre-train on a general domain and then on a specific domain.

Instead of manually trying all the task combination options, reward-based learning methods have been proposed by Guo et al. (2019); Wang et al. (2020) to solve the problem to maximize task transfer. However, such methods, which implicitly or explicitly depend on artifacts during training such as evaluation accuracy of the current time stamp or example-level weight, suffer from instability of rewards and randomness of states changes. As a result, it can be difficult to re-train to ensure to get good model quality.

Unlike previous works, we focus on learning a *static sampling schedule*, which means our schedule, once learned, should be fixed before any future model training. Future model training should not involve policy optimization. That makes it easy to re-train the models when underlying system changes or share the sampling schedule with other teams who do not have policy-optimization expertise in an industrial setting.

3 Methods

In Sec. 3.1, we introduce our algorithms for learning task sampling policy. In Sec. 3.2, we discuss our exploration strategies.

3.1 Learning Task Sampling Policy

We consider the task sampling policy to be discrete and stochastic. At time step t, the policy outputs a distribution $\pi(\cdot|s_t)$ over the actions (s_t is the state that the policy depends on). Here we define an action to be a choice of task to sample data from, and we sample a batch at each time step.

Our objective is to maximize the expected reward R which we define as model's evaluation accuracy on the target task:

$$\mathbb{E}_\pi \left[\frac{1}{K} \sum_{k=0}^{K} \sum_{t=0}^{T} R(s_t^{(k)}, a_t^{(k)}) \right], \quad (1)$$

where K is the batch size, and k marks the k-th sample.

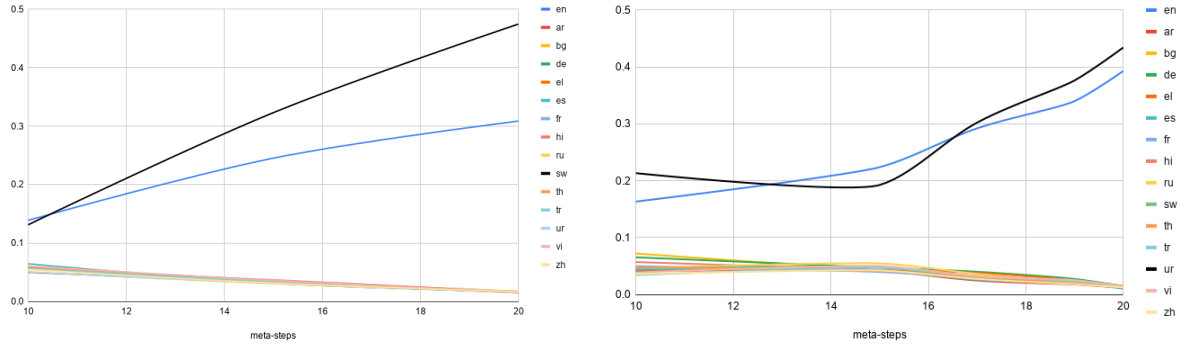
We parameterize the policy π with θ, and learn it with REINFORCE policy gradient (Sutton et al., 1999) (we ignore batch in notation):

$$\mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T} R(s_t, a_t) \nabla_\theta \log \pi_\theta(s_t, a_t) \right]. \quad (2)$$

Our goal is to learn a static sampling schedule that does not depend on complex dynamics of model parameter changes during training but only relies on timestamp, which makes s_t only contains a timestamp. In practice, running evaluation on validation set to obtain reward signal at each training step could be costly. Therefore, we define a meta-step to group N continuous training steps by creating a mapping $M(t)$ that maps a step t to a meta-step, and only run evaluation at each meta-step.

In this work, our policy $\pi_{\theta_{M(t)}}$ is simply parameterized as a vector of logits that defines a Boltzmann distribution $\text{Softmax}(\theta_{M(t)})$ at each time step. The logits $\theta_{M(t)}$ are trainable weights and uniformly initialized

We consider two different types of policy:



(a) Target language: Swahili

(b) Target language: Urdu

Figure 1: The sampling schedule learned for different target languages across meta-steps. The black line is the target language's translate-train data and the blue line is English. We can see the learned sampling policy change as the target languages change.

1. Time-variant policy: The policy at time step t is defined as $\pi_{\theta_{M(t)}}$. For all time steps in a meta-step, we use the same sampling policy.

2. Time-invariant policy: We only use one task sampling policy across all time steps within one run of model training, and thus $\pi_{\theta_{M(t)}}$ reduces to π_θ.

During training, the model is updated at each step using sampled examples by gradient descent. At each meta-step, we run evaluation on the full validation set to obtain accuracy, and use it as a reward to update θ or $\theta_{M(t)}$ through gradient ascent as in Eq. (2). This process ends until we reach the maximum number of training steps. While we can optionally repeat the training process with a randomly initialized model to further optimize the policy, practically we are able to find reasonably good sampling policy in one training run in all of our experiments.

After the searching is done, we re-train with the updated policy to generate the final optimized model.

3.2 Exploration Strategy

To facilitate exploration, the first strategy we consider is ϵ-exploration (Tokic and Palm, 2011), which refers to sampling actions from a uniform distribution with probability ϵ and sampling from policy with probability $1 - \epsilon$ during training. The second strategy we consider is a heuristic guided exploration that simply samples data from the target task.

We use a combination of these two. Let n be the number of tasks, T_p be the primary task, $\{T_1, ..., T_n\}$ be the task set we consider, and $Unif\{T_1, ..., T_n\}$ be a discrete uniform distribution over actions/tasks. To draw the k-th sample in a batch at time step t, we sample a $p \in \mathbb{R}$ from a continuous uniform distribution $Unif(0, 1)$, and then sample an action (choice of task) as follows.

$$a_t^{(k)} \sim \begin{cases} Unif\{T_1, ..., T_n\}, & \text{if } p < \epsilon \\ \{T_p\}, & \text{if } \epsilon <= p < \gamma \\ \pi(\cdot|s_t^{(k)}), & \text{otherwise} \end{cases}$$

Then we sample from the data associated with the task.

4 Experiments

We use the public BERT-base[1] (Devlin et al., 2018) checkpoints as our base model to run multi-task fine-tuning experiments. We consider two scenarios for knowledge transfer: cross-lingual transfer between languages of the same task and cross-task transfer between tasks of the same language.

For the cross-lingual transfer study, we use the Cross-lingual Natural Language Inference (XNLI) corpus (Conneau et al., 2018). The data set asks whether a premise sentence entails, contradicts, or is neutral toward a hypothesis sentence. Crowd-sourced English data are translated to ten other languages by professional translators and used for evaluation, while the English MultiNLI (Williams et al., 2018) training data and its translation to other languages, *translate-train* (Hu et al., 2020), are used for training. For the cross-task transfer study, we cover multiple tasks from the GLUE benchmark (Wang et al., 2019).

[1] https://github.com/google-research/bert

Language	English	Target Language	Multi-task uniform	Multi-task time-variant	Multi-task time-invariant
Urdu	0.5693	0.6616	0.6618	0.6651	0.7032
Swahili	0.5094	0.6785	0.657	0.6877	0.7045
Thai	0.5386	0.6853	0.6847	0.7096	0.6853
Hindi	0.5903	0.6897	0.6811	0.6823	0.7127

Table 1: Performance improvements on the XNLI test dataset by our proposed methods. Our proposed methods, both the time-varying version and the time-invariant version, outperforms various commonly used data re-weighting baselines.

Auxiliary	Tasks	Single Task	Multi-task uniform	Multi-task proportional	Multi-task time variant	Multi-task time invariant
QNLI	RTE	0.675	0.7369	0.6679	0.7112	0.7473
QNLI	MRPC	0.8284	0.8652	0.7279	0.8653	0.8627
MNLI	RTE	0.675	0.6787	0.7401	0.7148	0.8014
MNLI	MRPC	0.8284	0.8358	0.7696	0.8603	0.8603

Table 2: Performance improvements on low-resources GLUE tasks, RTE and MPRC, when using high-resource GLUE tasks. Our proposed methods, both the time-varying version and the time-invariant version, outperforms various commonly used data re-weighting baselines.

For all of our experiments, we compare two of our proposed methods to three baselines: uniform-sampling that uses a uniform weight to sample across all tasks or languages, proportional sampling, and the target task baseline that only uses the labeled data from the target task for training.

4.1 Training Settings

For all of our experiments, the classification model is simply a linear projection on top of BERT's classification ([CLS]) token. One meta-step contains 1000 steps. During fine-tuning, we use batch size 64 and ADAM optimizer (Kingma and Ba, 2017) with learning rate $2e^{-5}$. For each REINFORCE update, we use learning rate 0.001 and run stochastic gradient descent for 100 steps. We train all models on TPU v3 with 8 cores. All code is implemented in Tensorflow.

4.2 Experimental Results

Multilingual BERT Based on the performance of multilingual BERT on XNLI, four languages consisting of Urdu, Swahili, Thai, and Hindi which are either low-resource or had low performance were picked. Single-task baseline denotes the classification performance of multilingual BERT on these languages with their respective translate-train data.

Table 1 shows the accuracy comparisons of the proposed methods on the languages picked from the XNLI dataset. Note that uniform sampling is the same as proportional sampling, because the data set size is the same for all languages. From this table, we see that multi-tasking with naive uniform sampling performs worse than even target language baselines. Our methods, on the other hand, can utilize the auxiliary tasks to achieve quality improvements over single-task baselines. Interestingly, time-invariant policies in general have similar quality to time-varying policies. This may indicate time-varying schedule used by previous work (Guo et al., 2019) may not be necessary.

The sampling schedule is shown in Figure 1 with meta-steps on the X-axis and weights of languages on the Y-axis. We see the model outperforms the baselines by properly combining English supervised data or the target language's translate-train data. We also observe that a good sampling schedule can vary from task to task. It validates our motivation to learn the sampling schedule instead of using a rule-based one.

English BERT For GLUE, we select two high-resource tasks MNLI and QNLI with more than 100k labeled data each as auxiliary tasks. We target cross-task transfer to low-resource tasks: MPRC and RTE.

The results are shown in Table 2. Unlike the results on XNLI, uniform sampling, which naively up-samples the low-resource tasks, generally gives better accuracy than single-task baseline as shown in previous work (Liu et al., 2019). We show that the sampling policy learned by our algorithm consistently outperforms all rule-based alternatives.

5 Conclusions

We propose a simple and effective RL-based algorithm to learn a static sampling schedule for multi-

task learning, with the goal to improve the performance of one target task. We apply the algorithms to model fine-tuning and show that the proposed approach outperform popular rule-based baselines.

6 Ethical Considerations

Compute: For the entire work in this paper including running all the experiments, we have only used TPUs. TPUs have been shown to be significantly faster than GPUs for tensor based operations. The dataset used in the experiments were all precomputed thus saving further computation in the form of pre-processing.

References

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*.

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

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2019. Autosem: Automatic task selection and mixing in multi-task learning. *arXiv preprint arXiv:1904.04153*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation.

Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *arXiv preprint arXiv:1706.05137*.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding.

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

Jiaqi Ma, Zhe Zhao, Jilin Chen, Ang Li, Lichan Hong, and Ed H. Chi. 2019. Snr: Sub-network routing for flexible parameter sharing in multi-task learning. In *AAAI*, pages 216–223.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. 2019. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1162–1172, Minneapolis, Minnesota. Association for Computational Linguistics.

Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPs*, volume 99, pages 1057–1063. Citeseer.

Michel Tokic and Günther Palm. 2011. Value-difference based exploration: adaptive control between epsilon-greedy and softmax. In *Annual Conference on Artificial Intelligence*, pages 335–346. Springer.

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. 2020. Exploring and predicting transferability across nlp tasks.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020. Balancing training for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8526–8537, Online. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.