A workflow for scalable benchmarking and hyperparameter tuning of transfer learning methods - Exposé

Alpay Yilmaz

November 16, 2022

1 Introduction

Machine learning is increasingly important to peoples lives [1]. Especially in the field of image classification, neural networks have shown impressive results [2]. Furthermore transfer learning becomes increasingly relevant in the many field with scarce training data because of its ability to transfer information from a many sample source domain to a few sample target domain. [3].

Machine learning methods are complex and contain numerous different parameters (hyperparameters, sample sizes, validation sample sizes, ...). Transfer learning multiplies the number of these parameters and further increases the complexity of machine learning [3]. The high number of configurable parameters can be overwhelming for researchers, cause mistakes in the research, and increase the time and human effort required for solving machine learning tasks.

Currently, there is no standardized approach to systematically evaluate and compare transfer learning methods. In the literature, different methods tend to be evaluated on different datasets, different sample sizes, different source and target data, different underlying deep learning architectures (encoder, e.g., resnet, vgg, ...), and with varying compute-budgets for hyperparameter tuning [4]. Thus, it is exceedingly difficult to meaningfully compare transfer learning methods and to make informed decisions on which method is most appropriate for a given data type and sample size. The authors of [4] created a unified re-implementation and evaluation platform for certain contrastive semi-supervised learning methods, however this platform is too specific for usage outside the intended domain.

2 Goals of this Thesis

The overarching question of this thesis will be if and how we can realize a unified and scalable evaluation workflow for transfer learning methods, with the required hyperparameter tuning procedures included in the workflow. Our workflow should systematically benchmark multiple transfer learning methods on different datasets, with different transfer tasks, sample sizes, and deep learning architectures. We define scalable as a measure of how easy we can grow or shrink an application by using computational resources such as threads, clusters and gpus. Our method should be able run in a distributed manner within a cluster to accelerate the time consuming work of hyperparameter optimization.

3 Approach

Our proposed solution for the reduction of complexity and effort for systematically evaluating transfer learning methods is to develop a flexible workfow for an existing worflow managment system (e.g., snakemake, nextflow). This means that the researchers will instead of writing a program/script for their machine learning task and subsequent evaluation just write a configuration file with all necessary parameters, reducing the effort and time for the researchers. In a broader sense we want to create a platform capable of the same ideas of unified implementation and evaluation like the authors of [4], but generalized to transfer learning.

Nextflow [5] is a workflow manager that enables the development of portable and reproducible workflows. By creating a Nextflow script, we can use supported workload managers, which include SLURM.

Specifically, we will attempt to create a Nextflow script capable of systematic benchmarking of transfer learning methods with integrated hyperparameter optimization, which the researchers can execute on any number of environments supported by Nextflow. This should provide users with the ability to apply the suggestion from the authors of [4] for realistic evaluation of machine learning in a fast and easy manner.

3.1 CTLR

CTLR (Charite Transfer Learning Repository) is a developing transfer learning framework used for image classification. It consists of multiple components representing different aspects of TL, such as the model architecture or the data sets. Its purpose is to simplify and accelerate transfer learning development. This framework will provide the transfer learning methods and utility functions to be used in our newly designed workflow.

3.2 Hyperparameter Optimization

Hyperparameter optimization (HPO) is part of automated machine learning, which has been developed to not rely on experts experience when creating a model at the cost of additional computational resources [1]. Hyperparameter refers to parameters that can not be updated during training. They often determine the structure of the model [1]. HPO serves four purposes [1]:

- Reduce work of artificial intelligence experts
- Lower threshold of expert knowledge for research
- Improve accuracy and efficiency of neural network training [6]
- Make the training results more reproducible [7]

Currently, CTLR does not support hyperparameter tuning. This means that the first task should be an implementation of at least one hyperparameter tuning method into CTLR and provide a working example.

3.2.1 Grid Search [1]

Grid search is a basic method for HPO. It performs exhaustive search on the hyperparameter set specified by the users. For grid search to be effective, users need preliminary knowledge for the generation of all candidates. The trial runs are independent from each other, which makes them easy to run in parallel, but the computation resources increase exponentially when more hyperparameter need to be tuned.

It is the most widely used HPO method [7].

3.2.2 Random Search [1]

Random search is a basic improvement on grid search. It indicates randomized search over hyperparameters from certain distributions over possible parameter values until its predetermined budget is depleted or desired accuracy is reached. It is also able to run in parallel.

Random search is proven to create better results than grid search [8].

3.2.3 Bayesian Optimization [1]

Bayesian optimization is a typical method for almost all types of global optimization. It is a sequential model-based method aimed to become more accurate with more data [9]. Bayesian optimization tries to find the global optimum with the minimum number of trials by balancing exploration and exploitation [10] to avoid trapping into a local optimum.

It outperforms random and grid search. Furthermore it does not require users to possess preliminary knowledge of distribution of hyperparameters and posterior probability.

Bayesian optimization is more computationally efficient with fewer attempts, however parallel execution is difficult because of the dependency on previous attempts. One possible method of running Bayesian optimization in parallel is to increase the created models per step [11]. This would allow for

distributed training but create a loss of efficiency and for a gain in quickness[11]. Ideally, our program will allow the user to set the number of models created per step.

3.3 Parallel or Sequential Execution

When performing hyperparameter optimization, dependent on the method, users can tune their models in parallel (Grid and Random Search) or sequential (Bayesian) to find an optimum. Because we focus on transfer learning, we also have to consider if we want to separately find the optimum for the source model first and then for the target model or if we only want an optimum for the target model. Ideally, our program will provide options for both variants. All parameters listed in the previous section allow for distributed trials.

3.4 Distributed Execution

For distributed execution we will use a Nextflow executor for a cluster (e.g. SLURM). A Nextflow executor is the component that determines the system where a process is run [5]. By creating and using a Nextflow script, we should be able to make effective use of a cluster for distributed execution. The Nextflow script has to be designed and implemented in a way that makes distributed execution on a cluster possible.

4 Evaluation

For evaluation we want to check multiple goals for our application. First we want to confirm that our program can create transfer learning workflows (without hyperparameter optimization) comparable to ones made by experts. Then, we will include hyperparameter tuning in the workflow. Finally, we will extend the workflow to different task specifications (i.e., dataset, target task, sample size). We will compare the expert TL workflows with our generated workflows based on classification metrics usually used for machine learning (precision, accuracy, F1 score). For evaluation of the scalabilty of our workflows we will test the execution on a cluster with different number of threads, nodes, cores, gpus and hyperparameters. We will evaluate the results of the different test runs by comparing different metrics, such as execution time, resource utilization, etc..

Generally, we will investigate three representative transfer learning methods. The first method starts with pre-training a model on an auxilliary task, then freezing the lower layers and finetuning on the target task. The second method starts with pre-training in a self-supervised fashion [12], then freezing the lower layers and finetuning on the target task. The third method skips the pre-training step, and directly combines source and target task data in a semi-supervised fashion [13].

The datasets consists of MRI scans of brains. Pre-training auxilliary task (method 1) is prediction of sex. Self-supervised auxilliary task (method 2) and semi-supervised auxilliary task (method 3) will be invariance to data augmentation (warping). For fine-tuning we use the same dataset with the target task being age prediction. The prediction of age based on brain images is a highly researched topic. It can help detect degenerative diseases by comparing the predicted age of a patients brain to the real age [14]. If the real age is older than the predicted age, it is possible that the patient suffers from a degenerative disease [14].

References

[1] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

[2] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[3] Mohsen Kaboli. A review of transfer learning algorithms. 2017.

[4] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *Advances in neural information processing systems*, 31, 2018.

[5] Nextflow. https://www.nextflow.io/.

[6] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

[7] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[8] Why is random search better than grid search for machine learning. https://analyticsindiamag.com/why-is-random-search-better-than-grid-search-for-machine-learning/.

[9] Bayes rule applied. https://towardsdatascience.com/bayes-rule-applied-75965e4482ff.

[10] David Silver. Lecture 9: Exploration and exploitation. *Computer Science Department, University of London*, 2014.

[11] From sequential to parallel, a story about bayesian hyperparameter optimization. shorturl.at/BDEIO.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[13] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[14] Tobias Kaufmann, Dennis van der Meer, Nhat Trung Doan, Emanuel Schwarz, Martina J Lund, Ingrid Agartz, Dag Alnæs, Deanna M Barch, Ramona Baur-Streubel, Alessandro Bertolino, et al. Common brain disorders are associated with heritable patterns of apparent aging of the brain. *Nature neuroscience*, 22(10):1617–1623, 2019.