RESCIENCE C

Replication / ML Reproducibility Challenge 2022

[Re] Easy Bayesian Transfer Learning with Informative Priors

Martin Špendl[1] and Klementina Pirc[1]

[1]University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia

Edited by
Koustuv Sinha,
Sharath Chandra Raparthy

Reviewed by
Anonymous Reviewers

Received
04 February 2022

Published
23 May 2022

DOI
10.5281/zenodo.6574623

Reproducibility Summary

Scope of Reproducibility — In this work, we study the reproducibility of the paper: *Pre-Train Your Loss: Easy Bayesian Transfer Learning with Informative Priors*. The paper proposes a three-step pipeline for replacing standard transfer learning with a pre-trained prior. The first step is training a prior, the second is re-scaling of a prior, and the third is inference. The authors claim that increasing the rank and the scaling factor improves performance on the downstream task. They also argue that using Bayesian learning with informative prior leads to a more data-efficient and improved performance compared to standard SGD transfer learning or using non-informative prior. We reproduce the main claims on one of the four data sets in the paper.

Methodology — We used a combination of the authors' and our code. The authors provided a training pipeline for the user but not the code to fully reproduce the paper. We modified the training pipeline to suit our needs and created a testing pipeline to evaluate the models. We reproduced the results for the Oxford-102-Flowers data set on an Nvidia RTX 3070 GPU using approximately 310 GPU hours for the main results.

Results — Our results confirm most of the claims tested, although we could not achieve the exact same accuracy due to missing hyper-parameters. We reproduced the trend in how scaling the prior impacts the performance and how a learned prior outperforms a non-learned prior. On contrary, we could not reproduce the effect of rank in low-rank covariance approximation on model performance, as well as the beneficial boost in performance of Bayesian learning compared to the standard SGD.

What was easy — The authors' implementation provides various training and logging parameters. It is also helpful that the authors provided both the learned priors and scripts for the download, split and pre-processing of the data sets used in the study.

What was difficult — Setting the environment for the used packages to work correctly was difficult. Although many parameters are available for running the pipeline, their descriptions are misguiding, therefore a lot of time went into clarifying the parameter function and debugging different settings. The training also took a while, especially when training 5 models per data point.

Copyright © 2022 M. Špendl and K. Pirc, released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to Martin Špendl (martin.spendl@fri.uni-lj.si)
The authors have declared that no competing interests exist.
Code is available at https://github.com/MartinSpendl/ML-reproducibility-challenge-23 – DOI 10.5281/zenodo.7949229. – SWH swh:1:dir:a67381741fc59350e147883d925d6246f60bb3ef.
Open peer review is available at https://openreview.net/forum?id=JpaQ8GFOVu.

Communication with original authors — We contacted the authors via e-mail about their pipeline and their use of hyper-parameters but did not hear back.

1 Introduction

Transfer learning is a popular approach in deep learning, allowing model parameters from a trained model to be reused as an initialisation for a different task. This is especially useful when we have little data or computing resources available. Bayesian inference offers additional benefits with prior distributions encapsulating the uncertainty of pre-trained weights.
The authors suggest a pipeline for Bayesian transfer learning consisting of three steps:

1. Learning the prior on a source task.
2. Re-scaling the learned prior to express uncertainty.
3. Using the prior with Bayesian inference on a downstream task.

To learn a prior, we fit a probability distribution to the parameters extracted from the model trained on the source task, meaning we extricate the knowledge about the source task and can then use it as an informative prior on the downstream tasks, which saves us time and resources. The source and the downstream task have related yet different data distributions; thus, it is important to re-scale the prior. With such an approach, we are less restrictive and explore a wider parameter space. Finally, we use the informative prior with Bayesian inference to learn a posterior distribution of the model parameters on the downstream task and draw samples to initialise the model. The authors state their pipeline outperforms the standard transfer learning and is simple to apply due to combining easy-to-use existing components.

2 Scope of reproducibility

The paper suggests the pipeline for Bayesian transfer learning consisting of three main parts: (a) learning the prior, (b) re-scaling the prior and (c) Bayesian inference. We aimed to confirm the most important claim from each of those steps.
The first step is using a SWAG method [2] to construct a prior. We did not construct a prior using SWAG, as this was done using a method from another article. However, we used a pre-trained prior made public by the authors. With regard to this step, the authors claim that:

- **Claim 1**: by increasing the low-rank component from zero, the model's performance improves until it saturates at a relatively small number.

They back this claim by increasing the low-rank component of the learned prior from 1 to 10 and comparing the model performance of the ResNet50 model on the CIFAR-10 data set. We tested this claim on the Oxford-102-Flowers data set using the same model and prior but only using the low-rank span from zero to five as the authors provided only a covariance matrix of rank five.
The second step is re-scaling the prior with a variance and covariance scaling factor. Based on the scaling factor value, the re-scaling introduces the uncertainty in the prior; the larger the factor, the more uncertain it is. The authors claim that to optimise performance; we need to make a prior more diffuse, thus increasing the scaling factor.

- **Claim 2**: using a non-re-scaled prior can provide a worse likelihood than using Bayesian methods from scratch. Using a more diffused prior optimises performance to a certain point. Increasing a scaling factor further decreases the effect, making the prior near-uniform.

They back this claim by comparing the performance of the ResNet50 model on the CIFAR-10 data set using different scaling factors on a logarithmic scale. We tested this

claim on the Oxford-102-Flowers data set using *torchvision* and *SimCLR (self-supervised, SSL)* prior, provided by the authors.
The third step is Bayesian inference, which uses a learned and re-scaled prior from the previous steps. The authors claim that:

- **Claim 3:** modifying loss surface on a downstream task improves performance,
- **Claim 4:** Bayesian learning provides a particular performance boost with informative priors,
- **Claim 5:** informative priors lead to more data-efficient performance.

They back the claims by comparing the performance of different combinations of inference and priors on 4 different data sets with varying sizes. We chose the Oxford-102-flowers data set and *SimCLR* prior to reproduce the results.

3 Methodology

The authors released PyTorch pre-trained priors and the code for using priors for downstream inference. However, the code itself is insufficient to reproduce the paper's results. We used the authors' code as a framework for learning but implemented the evaluation part ourselves. The code was hard to navigate due to ambiguous parameters and a few hard-to-spot bugs.

3.1 Model descriptions

Replicated results are based on the ResNet50 model architecture with 2.5M trainable parameters. Throughout, we used *SimCLR* (self-supervised) pre-trained prior as our prior. We only used *torchvision* prior for the comparison of the scaling effect. We use Stochastic Gradient Langevin Dynamics (SGLD) with Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) for Bayesian inference and Stochastic Gradient Descent (SGD) for standard learning.

3.2 Data sets

Oxford 102 Flower Data set — The authors provided a python script that downloaded the data set and split it into train, test and validation sets based on the included files. Throughout the study, we use the Oxford-102-Flowers data set [3]. The set consists of 8189 flower images distributed into 102 classes, where each class has 10 training and 10 validation images, while other images are used for testing. The data set can be downloaded from https://s3.amazonaws.com/fast-ai-imageclas/oxford-102-flowers.tgz. Train, test and validation sets contain all the classes and have 1020, 6149 and 1020 images, respectively. We used the authors' code for pre-processing, specified by the data set's authors.
The authors did not specify the procedure for downsampling the data set. We took one image for each class for data sets of sizes 5, 10, 50 and 100 while sampling images uniformly for larger data set sizes.

3.3 Hyper-parameters

The authors performed a hyper-parameter search for models prior to training but did not report them. We extracted most of the parameters used in the final model from the available code and set the rest to the middle value among the tested ones. The hyper-parameters were the same for all trained models except for the one that was being varied in the experiment (rank and scaling factor).

Hyper-parameter	*batch size*	*learning rate*	*no. cycles*	*weight decay*
Value	16	0.01	4	10^{-4}

Table 1. Hyper-parameters used in training of models.

When training models on data sets with 5 and 10 images, we adjusted the batch size to 5 and 10, respectively. The number of epochs varied based on the size of the data set and the batch size.

3.4 Experimental setup and code

We modified and debugged the code from the authors. In every setting, we trained 5 models using the same parameters to evaluate the SE of the performance. This procedure was the same as in the original paper. Models were trained for 30000 training steps, resulting in 483 epochs for the full Oxford-102-Flowers data set. When training on smaller data sets (5 and 10), we trained models for up to 30000 epochs. The evaluation criteria of performance for all models was accuracy.

The importance of rank in low-rank approximation — We used Bayesian learning to train the model on the full training data set with re-scaled prior. We re-scaled the prior's variance and covariance matrix with a scaling factor of 10^5 and evaluated the performance of models with their covariance rank from 1 through 5. For the rank 0 approximation, we used a zero covariance matrix.

Re-scaling the prior — We trained the model with the rank of the covariance matrix set to 5 and 10 for *SimCLR* and *torchvision* priors, respectively. When scaling, we scaled both the variance and the covariance matrix. We evaluated the performance of the models with their scaling factor set from 10^0 through 10^9 by increasing the exponent by one.

Evaluating the performance — We trained 2 models using Bayesian learning and 2 models using SGD. One Bayesian and one SGD model were trained using a learned prior with a rank five covariance matrix and scaling factor of 10^9. The other two were trained using a non-learned prior using a normal zero-mean prior. The rank and the scaling of the prior were not specified by the authors, but we made an educated guess based on results from the first two steps and the default parameters of the pipeline.

3.5 Computational requirements

We fine-tuned the models using RTX 3070 GPU with 8Gb of memory. Training a model on the Oxford-102-Flowers data set with batch size 16 took around 1 hour, thus taking approximately 5 hours to reproduce one data point with the standard error. Training on smaller data sets took up to 4 hours. The total GPU time to reproduce results was approximately 310 hours (Table 2).

Results	Claims	approx. GPU hours
Rank influence	1	30
Scale influence	2	80
Inference comparison	3,4,5	200
Total		~ 310

Table 2. Approximate GPU hours to reproduce results of this paper.

4 Results

We confirm the paper's main claims by replicating some of the results; however, we were not able to achieve the exact same accuracy. Firstly, the rank of a low-rank component of the covariance matrix did not significantly affect the models' performance when using the Oxford-102-Flowers data set and the ResNet50 architecture. Secondly, scaling either of the learned priors with an increasingly larger scaling factor increased the models' performance. Lastly, we observe that Bayesian learning with a learned prior outperforms learning with a non-learned prior. However, we could not reproduce the result where the Bayesian model outperforms the standard SGD learning. Even though we could not fully reproduce the paper's results, we must acknowledge that some parameters were not specified, and our parameter choices might differ. We report results step-wise following the three steps in the article.

4.1 The influence of low-dimensional rank on performance

The authors claim that (**Claim 1**): with an increasing low-rank component from zero, the model's performance improves until it saturates at a relatively small number. They argue that increasing the rank of a low-rank component of a covariance matrix improves performance. However, the performance should stagnate after increasing the rank to some low value. They note that higher ranks add incrementally smaller corrections to the approximation and mostly add noise.
We observe the performance of the ResNet50 model with a learned *SimCLR (SSL)* prior to varying low-rank covariance approximations on the Oxford-102-Flowers data set (Figre). We used the scaling factor of 10^5 for variance and covariance scaling.

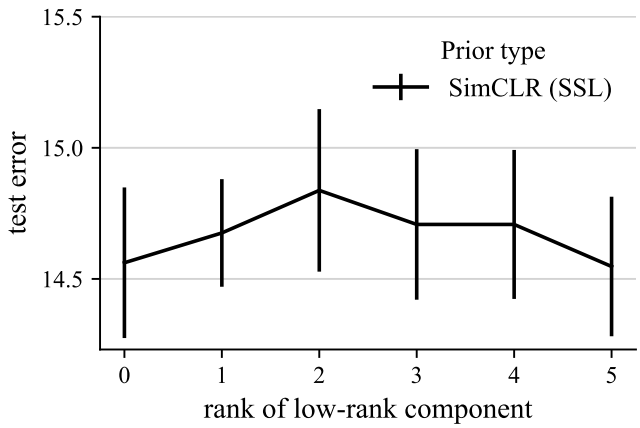

Figure 1. Model performance with varying ranks of a low-rank covariance matrix.

We observe that changing the rank of low-rank covariance approximation does not affect performance on the Oxford-102-Flowers data set. The results indicate that either the rank does not affect the performance or the scaling factor was too large for effect to be visible. The authors did not specify the scaling factor used; thus, we cannot compare the results fully.

4.2 The influence of prior scaling on performance

The authors claim that (**Claim 2**): using a non-re-scaled prior can provide a worse likelihood than using Bayesian methods from scratch. Using a more diffused prior optimises performance to a certain point. Increasing a scaling factor further decreases the effect, making the prior near-uniform.

We observe the performance of the ResNet50 model with a pre-trained prior with varying scaling factors on the Oxford-102-Flowers data set. We test the claim using *torchvision* and *SimCLR (SSL)* pre-learned prior. In experiments with varying scaling factors, a non-re-scaled prior is represented as scaling the variance and covariance by $10^0 = 1$. On the other hand, a non-informed prior is a pre-trained prior multiplied by a large scaling factor such as 10^9.

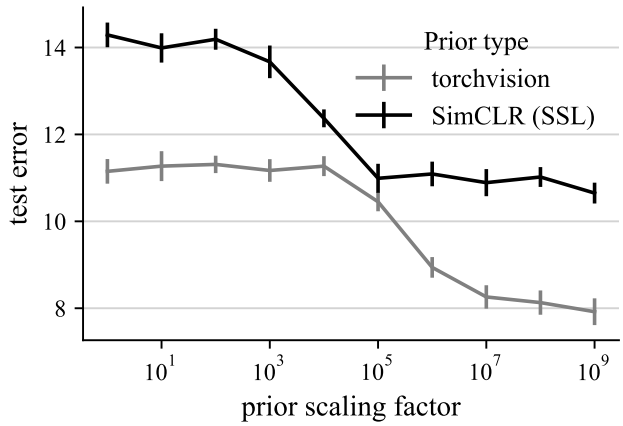

Figure 2. Model performance with varying scaling factors.

We observe that both the *torchvision* and the *SimCLR* prior produce a reduction in test error, thus increasing the performance. The decrease in test error using the *SimCLR* prior is evident from scaling by a factor of 10^2 through 10^5. The decrease in test loss is shifted towards higher scaling factors and spans from 10^4 through 10^8 using the *torchvision* prior. We note that these priors do not have the same rank with 5 and 10 for *SimCLR* and *torchvision*, respectively. Interestingly, the model performance is also better when using the *torchvision* prior.
We can confirm the first part of the authors' claim that using a non-re-scaled prior will negatively impact the performance compared to the re-scaled one. However, increasing the scaling factor towards the near-uniform prior does not decrease the performance compared to the prior multiplied by a smaller scaling factor.

4.3 Comparison of Bayesian and non-Bayesian learning

We observe the performance of the ResNet50 model with a combination of priors and inference techniques on the Oxford-102-Flowers data set. We use either Bayesian or SGD learning with the learned *SimCLR* or a Gaussian zero-mean prior.
We observe that models using the learned prior outperform models using non-learned prior. The increase in performance is mostly observed when training models on moderately sized data sets with only a few (if any) samples per target class. We also observe both Bayesian learning and standard SGD learning perform similarly on all data set sizes, with the SGD outperforming the Bayesian inference on larger data sets.
The authors claim that (**Claim 3**): modifying loss surface on a downstream task improves performance. We observe the same trend in model performance; thus, we can confirm the claim for this data set. They also claim that (**Claim 4**): Bayesian learning provides a particular performance boost with informative priors. The authors did not specify whether they were comparing Bayesian learning to SGD or learning with a non-learned prior. In the former, we can only confirm the claim for using smaller versions of the Oxford-102-Flowers, as with larger variations, the SGD performs equally or outperforms Bayesian learning. In the latter, we can fully confirm the claim as the performance increase is substantial, especially for moderately sized data sets. The last claim

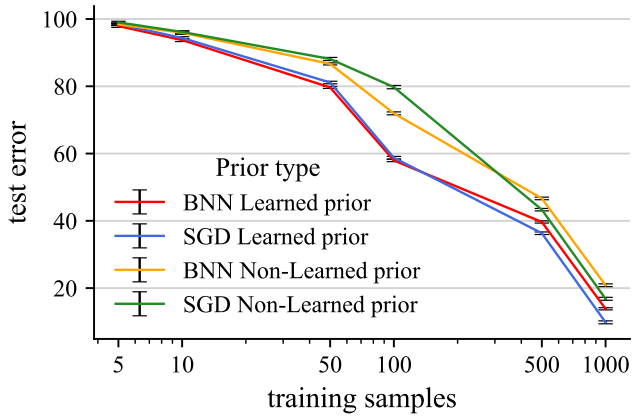

Figure 3. Comparison of model performance based on a prior selection.

is that (**Claim 5**): informative priors lead to more data-efficient performance. We can also confirm this claim, as we observe a larger difference between informative and non-informative model variants when the data set sizes are smaller.

5 Discussion

Our reproducibility report highlights the main claims of the paper. However, the reproducibility of the full paper was not achieved. There were a few unknown parameters regarding the model training and hyper-parameters for the models' optimal performance. Using the low-rank approximation of the covariance matrix does not improve performance on the Oxford-102-Flowers data set. The lack of increasing performance when using higher approximation ranks might be attributed to the prior or the task itself. We can only claim that Claim 1 should not be generalised without a proper evaluation of the task at hand.
Increasingly scaling a learned prior does have a beneficial effect on model performance; thus we can confirm Claim 2. However, we cannot claim that the increased performance is due to added uncertainty or scaling prior to near-uniform prior.
Finally, we can confirm Claims 3 and 5 about performance gains of Bayesian learning and informative priors. Even though we did not reproduce the exact values, the trend in overall performance among inference-prior combinations is the same. We cannot confirm Claim 4 about the performance boost of Bayesian learning because the SGD model performs better on the large-sized Oxford-102-Flowers data set.
There are many more experiments in this paper that we have not reproduced in this report. We would caution anyone who wishes to reproduce the results that the model performance will vary without an extensive and computationally demanding hyper-parameter search.

5.1 What was easy

The article was very well written, and its main claims were stated explicitly. The authors provided the code for the training pipeline, which is a great learning of individual parts of the proposed approach. Preparing the data and pre-processing were also well-written, and we could reuse those parts entirely.

5.2 What was difficult

We underestimated the computational time to reproduce the results, thus we would discourage reproducing the results without access to a computing cluster. Navigating through the code is fairly simple, but the lack of documentation requires more effort to understand individual parts. A bunch of TO-DO comments in the code reduce the confidence in the code to function properly. Because the training cycles are long and parameter testing is time-consuming, we would encourage those who wish to reproduce the results, to take the authors' code as a template and write the pipeline themselves.

5.3 Communication with original authors

We communicated with the authors via e-mail but did not receive any reply. Thus we would caution about generalising our results as we did not perform a greedy hyper-parameter search as did the authors.

References

1. R. Shwartz-Ziv, M. Goldblum, H. Souri, S. Kapoor, C. Zhu, Y. LeCun, and A. G. Wilson. **Pre-Train Your Loss: Easy Bayesian Transfer Learning with Informative Priors**. 2022. DOI: 10.48550/ARXIV.2205.10279. URL: https://arxiv.org/abs/2205.10279.
2. W. Maddox, T. Garipov, P. Izmailov, D. Vetrov, and A. G. Wilson. **A Simple Baseline for Bayesian Uncertainty in Deep Learning**. 2019. DOI: 10.48550/ARXIV.1902.02476. URL: https://arxiv.org/abs/1902.02476.
3. M.-E. Nilsback and A. Zisserman. "Automated Flower Classification over a Large Number of Classes." In: **2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing**. 2008, pp. 722–729. DOI: 10.1109/ICVGIP.2008.47.