UNDER REVIEW

RESCIENCE C

Replication / ML Reproducibility Challenge 2022

Reproducibility study of 'Proto2Proto: Can you recognize the car, the way I do?'

Anonymous[1, ID]
[1]

Edited by
(Editor)

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
03 February 2023

Published
–

DOI
–

Reproducibility summary

Scope of Reproducibility – This paper analyses the reproducibility of the study *Proto2Proto: Can you recognize the car, the way I do?* [1]. The main contributions and claims of the study are: 1) Using Proto2Proto, a shallower student model is more faithful to the teacher in terms of interpretability than a baseline student model while also showing the same or better accuracy; 2) Global Explanation loss forces student prototypes to be close to teacher prototypes; 3) Patch-Prototype Correspondence loss enforces the local representations of the student to be similar to those of the teacher; 4) The proposed evaluation metrics determine the faithfulness of the student to the teacher in terms of interpretability.

Methodology – A public code repository was available for the paper, which provided a working but incomplete and minimally documented codebase. With some modifications we were able to carry out the experiments that were best supported by the codebase. We spent a total of 60 computational GPU hours on reproduction.

Results – The results we were able to produce support claim 1, albeit weakly. Further results are in line with the paper, but we found them to go against claim 3. In addition, we carried out a theoretical analysis which provides support for claim 4. Finally, we were unable to carry out our intended experiment to verify claim 2.

What was easy – The original paper was clearly structured and understandable. The experiments for which configurations were provided were simple to conduct.

What was difficult – The public codebase contained minimal documentation. Moreover, the use of variable names did not correspond between the code and the paper. Furthermore, the codebase lacked elements vital to reproducing some experiments. Another significant constraint were the computational requirements needed to reproduce the original experiments. Finally, the code required to reproduce one of the visualizations was not provided.

Communication with original authors – We contacted the authors to ask for trained model weights and missing hyperparameters for several experiments. We did not receive a response.

Copyright © 2023 Anonymous, released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to ()
The authors have declared that no competing interests exists.

1 Introduction

Within the field of computer vision, deep models have achieved high accuracy in a variety of tasks. However, it remains generally difficult to understand their predictions because of their black-box nature. This interpretability issue is problematic when considering applications involving high-stakes decisions [2]. Prototypical deep learning methods have relatively high levels of interpretability, by comparing parts of the input image to prototypes learned during training. For example, a bird image classification decision could be explained as 'an image seems to contain a green kingfisher, because *this* part of the image looks like *this* prototypical feature of a green kingfisher' [3]. Prototypical methods introduced in recent years include ProtoPNet [4] and ProtoTree [5].
Knowledge distillation, a concept that has recently received increased attention, uses deep teacher models to train shallower student networks that aim to imitate the teacher and obtain competitive performance [6]. Although there has been a lot of focus on retaining or increasing accuracy between student and teacher models (for example [7], [8], [9]), interpretability is often disregarded. Keswani et al. [1] propose the Proto2Proto technique, which applies knowledge distillation to prototypical methods to transfer interpretability from teacher to student models without significantly compromising performance. This method assumes an interpretable teacher model and transfers both performance *and* interpretability to the student.
This paper aims to reproduce the experiments of the Proto2Proto study and verify the authors' main claims. We try to reinforce the claims with both qualitative analysis (showing images with their respective prototypes) and quantitative analysis (using various metrics).
The rest of this paper is structured in the following way. We first list the main claims of the Proto2Proto paper that we attempt to verify. In section 2, we summarize the necessary theoretical background to understand Proto2Proto. Section 3 describes the original experiments and explains how we attempted to reproduce the study. In section 4, we present the results of our reproduction and investigations. These results and the overall reproducibility of the experiments and claims are discussed in section 5.

1.1 Scope of reproducibility

The main claims of the Proto2Proto paper are:

1. Using Proto2Proto, a shallower student model is more faithful to the teacher in terms of interpretability than a baseline student model, while also showing the same or better accuracy.

2. Global Explanation Loss forces student prototypes to be close to teacher prototypes.

3. Patch-Prototype Correspondence Loss enforces the local representations of the student to be similar to those of the teacher.

4. The proposed evaluation metrics determine the faithfulness of the student to the teacher in terms of interpretability.

The fourth claim requires a theoretical discussion of the concept of interpretability and an analysis of the paper's method. To verify the other claims, it is necessary to reproduce the experiments and analyze the results.

2 Background

2.1 ProtoPNet

ProtoPNet is a prototypical model. These models use prototypes that provide a way of adding interpretability to image classification models. During training, the model learns a set of prototypes that represent the most relevant regions of the dataset. When an input image is fed into the network, it is first put through a backbone CNN, which results in a set of features. These features are used to extract a set of patches (parts of the image) that can be compared to the prototypes. These features are compared to every prototype by calculating the distance between the two. The similarity score is the inverse of this distance and the highest similarity score for each prototype is stored. These scores are fed into the decision module to compute the output logits.
ProtoPNet is inherently interpretable by displaying which local image patches contribute to its decision. Besides this local interpretability, prototypical models also provide global interpretability, because the learned prototypes show the regions that the model focuses on to make its decisions. An example of prototype activation can be seen in Figure 1.

2.2 Proto2Proto

The purpose of Proto2Proto is to transfer interpretability from a (prototypical) teacher model to a student model. To do this, the authors define two new losses that force the student model to agree with the teacher model on both prototypes and local representations. The *Global Explanation Loss* L_{global} is an average of the distance between the prototypes of the student and the teacher (any distance metric could be used, but both the original and our paper use Euclidean distance). The *Patch-Prototype Correspondence Loss* L_{ppc} is calculated by taking the difference between the student's and the teacher's feature maps of all the active patches, divided by the number of training images. Active patches are local patches whose distance to the closest prototype is below a threshold τ. Thus, τ is a hyperparameter that influences the amount of active patches.
The total loss function now becomes

$$L_{\text{total}} = L_{\text{model}} + \lambda_{\text{global}} L_{\text{global}} + \lambda_{\text{ppc}} L_{\text{ppc}}$$

where L_{model} is the loss of the prototypical method being used and λ_{global} and λ_{ppc} are hyperparameters that balance the newly introduced losses.
In addition to these two new losses, the authors also introduce three new metrics that evaluate how close the student and teacher models are in terms of interpretability. The *Average number of Active Patches* (AAP) is the average number of active patches that a model has per test image. It is related to local interpretability and should be as similar as possible between the teacher and student. The *Average Jaccard Similarity of Active Patches with Teacher* (AJS) gives the overlap of the student's and teacher's active patches. It is calculated for a pair of models and can never be higher than 1. AAP and AJS are used to determine the Patch-Prototype Correspondence Loss. Finally, the *Prototype Matching Score* (PMS) determines the Global Explanation Loss by comparing the prototypes of the teacher and the student. This requires a modified distance metric and matching algorithm because the mapping between student and teacher prototypes is not known.

3 Methodology

To verify the first three claims, we identified the following requirements:
Evaluating claim 1 requires training a teacher model, a baseline student trained without Proto2Proto and a student model trained using knowledge distillation with the Proto2Proto method. If the claim holds, the knowledge distillation model is expected to be evaluated

higher on both accuracy and the novel evaluation metrics than the baseline student model. Furthermore, we perform a qualitative analysis of the learned prototypes.
For claim 2, we need to confirm that L_{global} forces high PMS. As such we need to ablate L_{ppc} and confirm that PMS stays high.
For claim 3, we need to confirm that L_{ppc} forces high AJS and high similarity between the student's and the teacher's AAP. As such we need to ablate L_{global} and confirm that AJS stays high and the student's AAP remains close to the teacher's AAP.
Finally, claim 4 needs to be verified by conducting a theoretical analysis.
To conduct the experiments we used the public code repository of the authors[1]. We used YAML files containing arguments provided by the authors, and altered them where necessary.

3.1 Model descriptions

The authors use two existing prototypical methods: ProtoPNet [4] and ProtoTree [5]. Different CNN architectures were used to extract features from the input image. For teacher models, the ResNet-50 and VGG-19 architectures were used. The student models used the ResNet-18, ResNet-34, and VGG-11 architectures. All of these CNN models have been pre-trained on the ImageNet-1K dataset and the weights are downloaded from the PyTorch website.
The provided codebase lacked any reference to ProtoTree, thus we did not use this architecture in our experiments. We reproduced only the ResNet-50 to ResNet-18 experiment because this was the only configuration for which the authors provided the YAML files.

3.2 Datasets

The authors conducted experiments on two datasets. The first is the Stanford Cars Dataset (CARS)[2] [10]. It contains 16,185 images in 196 classes, which are split into a training set of 8144 and a test set of 8041 images. The training set is augmented by rotating, skewing, flipping, and distorting the images. The original paper created 40 variations of each picture. This was not feasible for our experiment since it increased train times beyond our computational capacity. Therefore, we lowered the number of variations to 4 per picture.
The second dataset that was used was the CUB-200-2011 [11] dataset. However, the authors did not provide documentation on the used parameters for the models trained on this dataset. We augmented the CUB dataset and trained it with the same argument settings. However, due to the lack of hyperparameters, we were unable to train the models on this dataset faithfully. As such, we only used the CARS dataset.

3.3 Hyperparameters

Training the models requires different hyperparameters, that can be separated into a few categories.

Prototypical method — In both ProtoPNet and Proto2Proto, the network learns 10 prototypes per image class. Other hyperparameters for the knowledge distillation experiment from ResNet-50 to ResNet-18 were provided by the Proto2Proto authors in YAML files. These files include learning rates for several parts of the model and information about the dataset. In the supplement, the authors state that '[They] follow the same hyper-par[a]meters as ProtoPNet [...]' [1, Suppl. p. 1]. However, when comparing the YAML file with the hyperparameters used to train the models in the original ProtoPNet paper [4, Suppl. p 24], we found that both deviate in the weight for the cluster cost (see A.1).

[1] https://github.com/archmaester/proto2proto
[2] https://ai.stanford.edu/~jkrause/cars/car_dataset.html

New hyperparameters for Proto2Proto — The loss function contains two novel hyperparameters: λ_{global} and λ_{ppc}. Both values are set to 10 in all experiments. The threshold value τ that determines the maximum distance between a prototype and its active patch is set to 100 during training. During testing, models are evaluated using different values of τ (see A.2) including infinity (i.e. no maximum distance), and the value yielding the highest accuracy is used. Prototypes have a dimension of 128 for the VGG model architectures and 256 for the ResNet architectures. The authors note that these values were found to be optimal, but do not explain why or how they arrived at them.

3.4 Experimental setup and code

To carry out the two experiments to reproduce the results of the paper and verify the authors' main claims, we modified the provided PyTorch code. Our code will be made publicly available[3]. The large majority of our code comes from the original authors[4] and the authors of ProtoPNet[5].
We trained the student, knowledge distillation, and teacher models with a batch size of 128. We trained the models for 35 epochs. All other hyperparameters were as provided in the YAML file of the Proto2Proto repository.
When reproducing Table 4 from the original paper in our second experiment, we did not include the rows that reused the teacher's decision module for the student, because the code contained no reference to this configuration. Furthermore, for the knowledge distillation student with both novel losses ablated, we reused the baseline student, because the Global Correspondence Loss and Patch-Prototype Loss were the only losses back-propagated in the first phase of loss calculation.

3.5 Evaluation metrics

To evaluate whether the models satisfy claim 1 and claim 3 we used the metrics accuracy, *Average number of Active Patches* (AAP) and *Average Jaccard Similarity of Active Patches with Teacher* (AJS).
Due to computational constraints, we were unable to evaluate the models using *Prototype Matching Score* (PMS), which prevented the reproduction of the experiment to verify claim 2.

3.6 Computational requirements

For training, we used a local device with an AMD Ryzen 7 3700x CPU with a RTX 2070 super GPU and 2 cloud devices which both used an NVIDIA T4 GPU from Google Colab. The computational requirements per experiment were 15 GPU hours on average using a batch size of 128 with 8 workers. The total computational requirements of all experiments were 60 GPU hours.
Our experiments only used 4 augmentations per image instead of 40. The average runtime of the model, when given the test set, is 25 minutes given a batch size of 128.

4 Results

4.1 Experiment 1

The results of the first experiment are shown in Table 1. In agreement with the original study, our knowledge distillation student model is closer to the teacher model in both accuracy and interpretability. These results support claim 1. Comparison of our results

[3] https://anonymous.4open.science/r/Fact-2164
[4] https://github.com/archmaester/proto2proto
[5] https://github.com/cfchen-duke/ProtoPNet

Datasets	Methods	Settings	AAP	AJS (↑)	Accuracy (↑)
CARS	ProtoPnet	ResNet-50 (Teacher)	35.53	1.0	77.32%
	ProtoPnet	ResNet-18 (Student)	40.55	0.71	70.39%
	Ours	ResNet-50 → ResNet-18 (KD)	39.31	0.73	72.94%
				(+0.02)	**(+2.55%)**

Table 1. Accuracy and interpretability of the Proto2Proto student with a ProtoPNet teacher and ResNet backbone. Interpretability is evaluated using the AAP and AJS interpretability metrics.

L_{ppnet}	L_{ppc}	L_{global}	AAP	AJS	Accuracy
✓			40.55	0.71	70.39%
✓		✓	17.78	0.73	51.22%
✓	✓		31.51	0.53	13.44%
✓	✓	✓	39.31	0.73	72.94%
Teacher			35.53	1.0	77.32%

Table 2. Performance of ResNet-18 student trained using a ResNet-50 teacher on different losses.

with Table 2 from Keswani et al. [1] shows that our trained models obtain lower accuracy and higher AJS.
Figure 1 shows the most highly activated prototypes for our trained models for a sample test image. Contrarily to Figure 1 in the original paper, we do not observe similarities between the most active prototypes of the teacher and the knowledge distillation student.

4.2 Experiment 2

Table 2 shows the results of the loss ablation experiment. When L_{global} is ablated, the distance between the student's and teacher's AAP become larger. Moreover, AJS decreases. Furthermore, when L_{ppc} is ablated, the AJS remains the same. These results are in line with the results from the original paper, but go against claim 3.

4.3 Theoretical analysis

To evaluate claim 4, it has to be made explicit what the authors mean by 'faithfulness of the student to the teacher in terms of interpretability'. Usage of the concept interpretability varies throughout the field [12]. Definitions are often left implicit, but one that has been given is 'the ability to explain or to provide the meaning in understandable terms to a human' [13, p. 93:5]. Prototypical methods satisfy this notion of interpretability. The goal, then, is not just to use an interpretable teacher model to train an interpretable student model, as the student model is already inherently interpretable due to its prototype layer.
Instead, the goal is that the student provides explanations that are similar to those of the teacher. When the authors talk about transferring interpretability from teacher to student, they do not refer to how interpretable the model is, but to the specific explanations that the model provides. This is why the proposed evaluation metrics are measures of proximity between the prototypes of both models. Under this understanding of interpretability transfer, claim 4 holds.

5 Discussion

The principal goal of this paper was to reproduce the experiments of the Proto2Proto study and to verify the authors' main claims. We were able to reproduce two experiments

Figure 1. Comparison of sample prototypes of test image between teacher model, baseline student model and Proto2Proto student model, following Figure 1 from the original paper. We do not observe noticeable similarities between any of the models' most active prototypes.

from the original setup, with some limitations.

5.1 Verifying claims

Claim 1 — The quantitative results we obtained are in line with claim 1. However, we were not able to conduct as many experiments as in the original research, so these results are weak. Moreover, our qualitative results do not show the same conclusion. The code required to obtain these results had to be modified, and we have reason to believe the final code is faulty: 10 out of 10 runs on the teacher model with different test images each gave an incorrect prediction, which seems unlikely given that the model got an accuracy score of 77%.

Claim 2 — We were unable to verify or falsify claim 2 because we could not use the provided code to calculate PMS. This code could be optimized in future work.

Claim 3 — Our quantitative results go against claim 3. Ablating the global loss leads to an increased difference in AAP and a lower AJS, suggesting that Patch-Prototype Correspondence Loss is not sufficient to force student prototypes to be close to the teacher.

Claim 4 — Our theoretical analysis of the paper supports claim 4. Future research could make the goal and interpretation of interpretability transfer more explicit.

5.2 What was easy

The original paper was clearly written and well-structured, which facilitated a quick understanding of the theoretical knowledge. There was a clear overview of the claims, which simplified the process of verifying the paper in an organized manner. The experiments for which YAML files were provided were simple to conduct.

5.3 What was difficult

Deficient documentation — Using the authors' code was more difficult than expected. There was minimal documentation: the code contains some comments, but functions and classes had no explanation. Furthermore, there was no overview of the folder structure. Use of variable names in the paper and in the codebase did not correspond (e.g. in the paper they refer to 'Patch-Prototype Correspondence Loss' and 'Global Explanation Loss' for the novel losses, while the codebase uses 'addOnLoss' and 'protoLoss', respectively). Additionally, the YAML file with arguments for the various scripts completely lacked documentation and not all variable names were transparent.

Incomplete codebase — The codebase lacked elements vital to reproducing some experiments. Specifically, there was no reference to the CUB dataset and ProtoTree.
YAML files were provided only for the experiment with knowledge distillation from Resnet-50 to Resnet-18. We tried recreating the other experiments, but had to make too many assumptions about the missing values to be able to faithfully reproduce them. Furthermore, the code contained no reference to the experimental configuration that reused the teacher's decision module for the knowledge distillation student.

Computational restraints — We were unable to run Proto2Proto on the CPU, since the code attempts to use a GPU regardless of the specified argument in the YAML file. As a result, a GPU was required not only for training, but also for setup and testing of the code.
Furthermore, a significant constraint in reproducing the paper were the computational requirements needed to reproduce the authors' original experiments. The requirements to train the models with the provided hyperparameters were beyond our computational resources, as were the requirements to evaluate the PMS metric. We were unable to obtain the weights of the trained models to alleviate this issue. In further research, the reproduction could be conducted with more computational resources, allowing for a more accurate reproduction.

Visualization — Figure 1 is a recreation of Figure 1 in Keswani et al. [1], for which the authors claim to have used similar code to the original ProtoPNet paper. The code required for these visualizations was not included in the Proto2Proto repository. To reproduce these images, we modified the ProtoPNet codebase. Although the Proto2Proto model builds on ProtoPNet, the implementations differ. Additionally, the ProtoPNet code expects different directory and model structure. Furthermore, generating the prototype visualization requires data about the model's prototypes. We had to modify the train scripts, in order to retrieve the model prototypes needed.

5.4 Communication with original authors

We reached out to the authors to ask for trained model weights and unspecified hyperparameters. We did not receive a response.

References

1. M. Keswani, S. Ramakrishnan, N. Reddy, and V. N. Balasubramanian. "Proto2Proto: Can you recognize the car, the way I do?" In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2022)**. 2022. arXiv: 2204.11830.
2. X. Li, H. Xiong, X. Li, X. Wu, X. Zhang, J. Liu, J. Bian, and D. Dou. "Interpretable deep learning: Interpretation, interpretability, trustworthiness, and beyond." In: **Knowledge and Information Systems** 64.12 (2022), pp. 3197–3234.

3. J. Donnelly, A. J. Barnett, and C. Chen. "Deformable ProtoPNet: An Interpretable Image Classifier Using Deformable Prototypes." In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. 2022, pp. 10265–10275.
4. C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su. "This Looks Like That: Deep Learning for Interpretable Image Recognition." In: **Advances in Neural Information Processing Systems**. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.
5. M. Nauta, R. van Bree, and C. Seifert. "Neural prototype trees for interpretable fine-grained image recognition." In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2021, pp. 14933–14943.
6. J. Gou, B. Yu, S. J. Maybank, and D. Tao. "Knowledge distillation: A survey." In: **International Journal of Computer Vision** 129.6 (2021), pp. 1789–1819.
7. G. Hinton, O. Vinyals, and J. Dean. "Distilling the knowledge in a neural network." In: **arXiv preprint arXiv:1503.02531** (2015).
8. M. Hong, Y. Xie, C. Li, and Y. Qu. "Distilling image dehazing with heterogeneous task imitation." In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2020, pp. 3462–3471.
9. G. Hong, Z. Mao, X. Lin, and S. H. Chan. "Student-teacher learning from clean inputs to noisy inputs." In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2021, pp. 12075–12084.
10. J. Krause, M. Stark, J. Deng, and L. Fei-Fei. "3d object representations for fine-grained categorization." In: **Proceedings of the IEEE international conference on computer vision workshops**. 2013, pp. 554–561.
11. C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. "The caltech-ucsd birds-200-2011 dataset." In: (2011).
12. S. Chakraborty, R. Tomsett, R. Raghavendra, D. Harborne, M. Alzantot, F. Cerutti, M. Srivastava, A. Preece, S. Julier, R. M. Rao, et al. "Interpretability of deep learning models: A survey of results." In: **2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation (smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI)**. IEEE. 2017, pp. 1–6.
13. R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. "A Survey of Methods for Explaining Black Box Models." In: **ACM Comput. Surv.** 51.5 (2018).

A Hyperparameter deviations

We found two deviations between the hyperparameters as provided in the Proto2Proto codebase and the hyperparameters as reported in the paper.

A.1 Cluster loss

ProtoPNet supplement:

> In our experiments on both CUB-200-2011 and Stanford Cars, we set the coefficient of the cluster cost to 0.8, and the coefficient of the separation cost to 0.08 during stochastic gradient descent of layers before the last layer, and we set the coefficient of the L^1-regularization term (on the weight connection between each prototype of class k and the logit of class $k' \neq k$) to 10^{-4} during convex optimization of the last layer. For the coefficient of the cluster cost and the coefficient of the separation cost, we considered three different settings: (1, 0.1), (0.8, 0.08), (0.6, 0.06), and chose the pair (0.8, 0.08) using cross validation. For the coefficient of the L^1-regularization term, we considered 10^{-3}, 10^{-4}, and 10^{-5}, and chose 10^{-4} also by cross validation.

Proto2Proto YAML file:

```
lossList:
  crossEntropy:
    consider: true
    weight: 1.0
  clusterSep:
```

```
      consider: true
      clusterWeight: 1.0 # Not 0.8
      sepWeight: -0.08
    l1:
      consider: true
      weight: 1.0e-04
```

A.2 Possible values of τ

Proto2Proto supplement:

> For training $\tau_{train} = 100$ and for testing, we choose τ_{test} from the set $\{0.1, 0.45, 1, 5, inf\}$.

Proto2Proto code:

```
distance_thresholds = [0.01, 0.1, 0.2, 0.45, 1.0, 3.0, 5.0, None]
```