On the reproducibility of Proto2Proto: Can you recognize the car, the way I do?

Diego Garcia Cerdas Rens Kierkels Thomas Jurriaans Sergei Agaronian
University of Amsterdam

Reproducibility Summary

Scope of Reproducibility

We study the reproducibility of the article *«Proto2Proto: Can you recognize the car, the way I do?»* by Keswani et al. Proto2Proto is a method that transfers the interpretability from one prototype-based model to another via *knowledge distillation*. Our goal is to reproduce the three main claims of the paper. It states that the proposed method increases the (i) interpretability transfer, the (ii) accuracy of the model; and (iii) teacher-student prototype similarity.

Methodology

We use the codebase published by the authors to confirm their claims about the Proto2Proto method. However, we could only reproduce the results using a third-party pretrained teacher due to time limitations in training and the absence of pre-trained models provided by the authors. Restructuring and documentation of their code were necessary to get the code running.

Results

We reproduced the experiments for VGG-11 and VGG-16 with a VGG-19 teacher, on the CUB-200-2011 (Birds) dataset, as well as the visualisation of the prototypes. The interpretability measures and accuracy show similar outcomes to those of the authors, albeit on a smaller scope due to limited computational resources. This partially verifies the claims of the authors.

What was easy

The publicly available code greatly facilitated the reproduction process, allowing us to focus on evaluating the results and methodology despite some incorrect or outdated instructions in the authors' repository.

What was difficult

Running the experiments was a challenging task due to the lack of comprehensive documentation, making it difficult to understand the various components and functions. Furthermore, the absence of some crucial hyperparameters made it harder to reproduce the study results.

Communication with original authors

The authors were contacted by email and asked to share pretrained weights and hyperparameters of the model. Unfortunately, no response was received before finalising this report.

1 Introduction

Deep learning models have become increasingly popular in recent years due to their ability to achieve state-of-the-art performance on a wide range of tasks. However, as these models become more complex and powerful, the question of interpretability has also become increasingly important. Deep learning models are currently being implemented in a wide range of sensitive areas, including, but not limited to, healthcare, finance, and criminal justice [9, 11, 2]. The interpretability of these models is crucial in terms of trust, causality, transferability, fairness, and other ethical considerations [7].

To address this problem, prototypical networks have been proposed as a solution. ProtoPNet [1] and ProtoTree [8] train the model to identify a set of prototypes or representative examples in the training data that are most relevant to a given prediction. These prototypes can provide explanations for the model's predictions in terms of the most similar examples from the training data. This helps users understand how the model makes its predictions to identify potential biases and classification problems. However, these efforts did not include the translation of interpretability to shallow networks.

[4] propose a novel method *Proto2Proto*, which aims to transfer the interpretability of a deep teacher model to a shallower student network through *knowledge distillation* (KD). Their results indicate that the proposed method successfully achieves this while maintaining competitive classification performance.

This report aims to reproduce the findings of Keswani et al. The report is structured as follows: Section 2 outlines the claims made in the original paper and our objectives. Section 3 provides the background information necessary to understand the work. Section 4 and 5 summarise our approach to reproducing the authors' claims. In Section 6, we present our results and compare them with the original paper. Finally, Section 7 concludes the report by discussing our experience with reproducing Keswani et al.

2 Scope of reproducibility

The authors compare the performance of two equivalent networks: their *Proto2Proto student*, trained through knowledge distillation from a teacher network, and a *baseline student*, trained without knowledge distillation. To determine the scope of our reproduction, we list main claims of the paper:

Claim 1: Proto2Proto student performs better in interpretability metrics compared to the baseline student.

Claim 2: Proto2Proto student outperforms the baseline student in terms of the top-1 accuracy score.

Claim 3: Proto2Proto student prototypes retain faithfulness to the teacher model, compared to the baseline student.

Claim 1 is validated by the authors through a set of experiments, where trained networks are evaluated on three novel interpretability metrics introduced in the paper. The validity of Claim 2 is confirmed by examining the classification accuracy obtained from the experiments. Finally, Claim 3 is supported by a qualitative evaluation of example prototypes generated by the trained models.

The validity of the claims made in the original paper can be assessed by reproducing their experiments. An implementation is provided by the authors; however, it is not trivial to use due to a lack of documentation. In addition, the computational needs for a full reproduction exceed our available resources.

Therefore, the primary objectives of this reproducibility study are:

- Reproduce a subset of the authors' experiments under limited computational resources and verify the trends reported in their paper.
- Restructure the existing evaluation code into a well-documented codebase with improved time efficiency.

3 Background

This section describes the principles behind the Proto2Proto method: *prototype-based networks*, *knowledge distillation*, and the metrics used to assess the quality of interpretability transfer.

Prototypical networks

A prototypical neural network, such as ProtoPNet, learns class-specific prototypes that represent the most important regions in training data. It typically consists of four modules: a *backbone* encoder that creates a compact representation of input data, an *add-on module* that outputs local representations, a *comparison module* that compares these local representations to the network's prototypes using similarity scores, and a *decision module* used to predict a class. Their transparent reasoning progress through the use of prototypes is considered to make prototypical networks inherently interpretable.

During the training of a prototypical network, the weights of the prototypes are projected or "pushed" at a regular interval during *push epochs*. In such an epoch, the weights of all prototypes are set to be equal to the values of the nearest latent image patch. This ensures that the prototypes correspond to specific features that exist in the actual training data.

In this paper, when mentioning prototypical networks, we specifically refer to ProtoPnet architectures. Furthermore, we differentiate between ProtoPnet architectures by mentioning the type of neural network used as a backbone encoder (e.g. VGG-19).

Active patches

In ProtoPNet's comparison module, each prototype is associated with the closest local representations, which are then used by the decision module. These representations are denoted by *active patches*. A hyperparameter τ defines the distance threshold to consider a patch active or inactive. We refer to the threshold used during training and testing as τ_{train} and τ_{test} respectively.

Knowledge Distillation

During the knowledge distillation process, a deep teacher model is used to generate the output probabilities for the training data, while a shallower student model is trained to generalise these output probabilities to make its own predictions. This process transfers knowledge from the teacher model to the student to improve the student's performance. After knowledge distillation, the shallower student network reaches a performance level closer to those of deeper models such as the teacher.

Proto2Proto applies this process to prototypical networks while also transferring the teacher's interpretability to the student, which would otherwise be lost. To do so, they introduce two novel loss functions: patch-prototype correspondence loss and global explanation loss. Their details are beyond the scope of this reproducibility study.

Interpretability transfer metrics

Proto2Proto introduces three novel metrics to determine the faithfulness of the student to the teacher in terms of interpretability.

- **Average number of Active Patches (AAP)**, the average number of active local representations. Closer AAP values of student and teacher result in better interpretability transfer. This metric is calculated for individual models.
- **Average Jaccard Similarity of Active Patches with Teacher (AJS)**, which determines the overlap of active local representations of the student with the teacher. The higher the value of a student-teacher pair, the closer they are.
- **Prototype matching score (PMS)**, which evaluates the closeness of the prototypes of the teacher and the student. To compute this score, the correspondence between the teacher and student prototypes is required, which is yet unknown for the baseline student. Similarly to the authors' experiments, we use the Hungarian Matching Algorithm (HMA) to match the prototypes. The similarity of local patches across models that activate the corresponding prototypes is determined by using *Modified-Jaccard-Similarity* as a distance metric to compare the prototypes of the teacher and the student. Details on this can be found in the Proto2Proto supplement material[5]. The results are averaged over the values of α, with $\alpha = (0.1, 0.2, 0.3, ...1.0)$.

The complete definition of these metrics can be found in the original paper.

4 Methodology

The authors conduct experiments on two fine-grained classification benchmark datasets: Caltech-UCSD Birds-200-2011 (CUB-200-2011) [10] and Stanford Cars [6]. They evaluate the performance of their proposed method using various backbone architectures, including ResNet and VGG. This results in a total of seven experiments, with three networks each. For each experiment, they provide the obtained interpretability metrics and test accuracy. They further provide example prototypes from their experiments and interpret them qualitatively.

Our objective is to reproduce their results using the original implementation made available on GitHub. However, the number of experiments, each requiring three trained models, makes full reproduction unfeasible due to limited resources. As a result, we restrict the scope of our reproduction to a subset of experiments. Furthermore, executing the evaluation code provided by the authors is nontrivial due to its lack of documentation. To address this, we provide a restructured and documented evaluation repository, which we use to obtain our results. This section describes the steps taken to reproduce the selected experiments and improve the evaluation code provided.

4.1 Reproducing model training

In this report, we focus on replicating two of Proto2Proto's knowledge transfer experiments: *VGG-19 teacher to VGG-16 student* and *VGG-19 teacher to VGG-11 student*. To bypass resource limitations, we use a pretrained VGG-19 teacher network for both experiments. The Proto2Proto and baseline students in each experiment are trained from scratch for a limited number of epochs. The following sections detail the process of loading the pretrained teacher and training the students.

4.1.1 Loading pretrained teacher

We use a pretrained ProtoPNet model as the teacher network, which was made available by [3] in their GitHub repository. According to the source paper, this model was trained on CUB-200-2011 using the settings and hyperparameters outlined in the original ProtoPNet publication, thereby enabling its use for Proto2Proto's experiments.

To be able to use this third-party checkpoint in the Proto2Proto implementation, we first extract the checkpoint's `state_dict` variable. To attain a direct correspondence with Proto2Proto's model definition, we simply adjust the names of the additional layer's weights. Instead of directly loading the model definition from a checkpoint file, we modified the checkpoint loading code to first initialise a new model and then load a given `state_dict` variable.

4.1.2 Training students

To train the VGG-11 and VGG-16 student networks, we use the original implementation without major modifications. However, the authors provide only example code for a *Resnet50 teacher to Resnet18 student* experiment using the Stanford Cars dataset, which we do not intend to reproduce. Therefore, we adapt these example argument files to include the appropriate hyperparameters for each of our experiments, which are detailed in Section 5.3. The architecture hyperparameters of both of our students correspond to those of our pre-trained teacher: 2000 prototypes of shape (1, 1, 128) with logarithmic activation function.

To minimise the expenses associated with fully training each model, we conduct training only up to the first push epoch. This approach allows us to observe trends during the early stages of training while retaining the benefits of a model with finetuned prototypes.

Since the pretrained teacher network was initially trained on the CUB-200-2011 dataset, we also train each student network on this dataset. The specifics of the CUB-200-2011 dataset are described in Section 5.2. The authors do not provide instructions on downloading and processing these data, so we follow those available in the ProtoTree repository.

4.2 Reproducing model evaluation

The authors provide evaluation code to calculate AAP, AJS, and PMS for all networks of a particular experiment in a single run. However, we found that this code was not sufficiently documented and was nontrivial to understand and execute. The code to find the nearest training patches for each prototype was also included, but the implementation did not provide usage examples. To facilitate our reproduction efforts, we reorganised and documented all the required evaluation code. The following sections outline the steps taken to restructure and implement the evaluation.

4.2.1 Restructuring code

To be able to compute the interpretability transfer metrics more easily, we provide a new wrapper class that focuses on one model at a time. This wrapper handles loading the model weights and provides separate measures for computing each metric, including accuracy. It reuses the computation of similarity scores to prevent redundant operations. It can also save the nearest training patches for each prototype.

The computation of PMS in the authors' implementation required extensive computing time. We provide an equivalent algorithm, optimised through the use of tensor operations, which reduces the computation time by a factor of approximately 20.

This restructured codebase provides all model weights, hyperparameter files, results, and analyses used in this paper. All evaluations run for this paper use our restructured implementation.

4.2.2 Running evaluation

To obtain our results for Claim 1, we compute AAP and AJS for all τ_{test} in [0.01, 0.1, 0.2, 0.45, 1.0, 3.0, 5.0, None]. The value τ_{test} = None corresponds to an infinite distance, as specified by Proto2Proto's authors. We compute PMS using only τ_{test} = None, as done in the author's implementation. In order to obtain results for Claim 2, only executing the evaluation and noting the accuracy score was necessary.

Lastly, to obtain results for Claim 3, we first find the nearest training patches without augmentation for each model's prototype. These prototypes must then be matched between the teacher and the students in order to compare them. This is done by finding the allocation that got the lowest cost during the PMS calculation. Reusing the values calculated during the PMS calculation, the computing time is significantly reduced.

5 Experimental setup

5.1 Computational requirements and limitations

All experiments were performed in the LISA Compute Cluster. The computing node used is equipped with an Nvidia Titan RTX GPU. In total, our reproducibility study cost 12.5 hours of GPU computing time. Approximately 10 hours of this was running the evaluation code, 2 hours of knowledge distillation training, and 30 minutes of traning the baseline student.

5.2 Datasets

The CUB-200-2011 dataset is a widely used benchmark for fine-grained visual classification of birds. It consists of 11,788 images of 200 bird species, with approximately 60 images per species. The data set is divided into two parts: a training set of 5994 images, and a testing set of 5794 images.

We perform offline data augmentation as during the training of our teacher model, using random rotation, skew, shear, distortion, and left-right flip to enlarge the training set so that each class has approximately 1200 training images.

5.3 Hyperparameters

We use the hyperparameters specified in the original paper and its accompanying material. For hyperparameters not mentioned in the paper, we refer to the only experiment implemented in the authors' codebase (Resnet50 to Resnet18), assuming that these hyperparameters are kept unchanged across all backbone architectures. An important change is the number of training epochs, which we changed from 300 (in the original implementation) to 12 in our experiments. Hyperparameters are listed in Appendix A.

6 Results

6.1 Interpretability and classification accuracy

To evaluate Claim 1 and Claim 2, the interpretability metrics and the top-1 accuracy scores obtained from our experiments are summarised in Table 1. The AAP and AJS scores correspond to $\tau_{\text{test}} = 0.1$. As can be observed, the Proto2Proto student achieves interpretability metrics that are closer to those of the teacher, compared to the baseline student. For example, the Proto2Proto student achieves an increase of 0.13 and 0.20 for AJS for the VGG-11 and VGG-16 models, respectively, and an increase of 0.39 and 0.49 for PMS. Furthermore, the Proto2Proto student shows a substantial improvement in classification accuracy, with a 22% and 21% increase for the VGG-11 and VGG-16 backbones, respectively.

We also observe that this trend holds for the different τ_{test} values reported in the original supplementary material. These values are shown in Figure 1 for the VGG-19 to VGG-16 experiment. A similar result is observed for the VGG-19 to VGG-11 experiment, as shown in Appendix B.

Architecture	AAP	AJS (↑)	PMS (↑)	Top-1 Accuracy (↑)
VGG-19 (Teacher)	6.51	1.0	1,0	76.23
VGG-11 (Student)	21.26	0.11	0.24	44.87
VGG-19 to VGG-11 (KD)	10.51	**0.24 (+0.13)**	**0.63 (+0.39)**	**66.72 (+21.85)**
VGG-19 (Teacher)	6.51	1.0	1.0	76.23
VGG-16 (Student)	16.20	0.12	0.26	52.28
VGG-19 to VGG-16 (KD)	8.65	**0.32 (+0.20)**	**0.75 (+0.49)**	**73.18 (+20.90)**

Table 1: Results of teacher and student models in terms of interpretability metrics and top-1 accuracy. AAP and AJS results correspond to $\tau_{\text{test}} = 0.1$

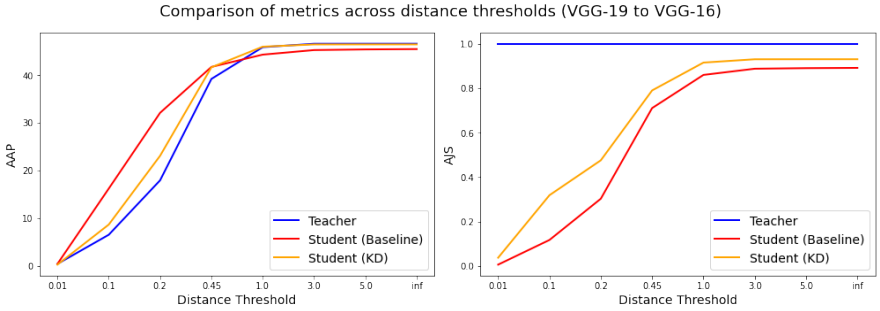

Figure 1: Effect of τ_{test} on AAP and AJS for the VGG-19 to VGG-16 experiment.

6.2 Prototype similarity

Figure 2 evaluates Claim 3 through prototype visualisations. The prototypes generated from the Proto2Proto student and the baseline student models are compared and both exhibit varying degrees of similarity to the original teacher model. The left section presents the majoritary case, where the Proto2Proto student is more faithful to the teacher. However, to refrain from generalizations, the middle section demonstrates instances of equal faithfulness between the two students, and the right section presents cases where an objective comparison is difficult to make. More visual examples can be found in Appendix C.

7 Discussion

The results of our replication study indicate that it is possible to reproduce the findings reported by the authors in the original article. However, these results have lower confidence than the original due to the limitations and restrictions faced during the replication process.

The lack of shared teacher weights made it necessary to use third-party pretrained weights, which involved making several assumptions. This increases the potential for experiment bias and reduces the generalisability of the results. The absence of pretrained teacher weights in the codebase also limits the ease of reproducing the experiments and raises concerns about the robustness of the claims made in the original article.

In the following section, we will discuss the results in the context of the reproducibility of the Proto2Proto paper and the limitations faced during the replication process.

7.1 Interpretability

The original article presents three new metrics to evaluate the transfer of interpretability from a teacher model to a student model in the context of knowledge distillation. Our evaluation supports Claim 1 that the Proto2Proto student outperforms the baseline student in terms of interpretability, as measured by these metrics.

However, while the calculation method of the metrics was included in the codebase provided by the authors, it was not fully functional and lacked sufficient documentation, making the evaluation process challenging. The presence of commented-out code and unused sections in the codebase further complicated the calculation of results.

Despite these limitations, the introduction of these innovative metrics by the authors is a commendable contribution to the field and provides a valuable tool for evaluating the interpretability of KD student models. More work is needed to make the evaluation process smoother and more accessible to the scientific community.

7.2 Accuracy

Claim 2 states that the Proto2Proto student outperforms the baseline student in terms of the top-1 accuracy score. Our evaluation supports this claim, as the Proto2Proto student achieved a substantial improvement in classification accuracy with a 22% and 21% increase for the VGG-11 and VGG-16 backbones, respectively.

However, the code provided by the authors to calculate the accuracy scores required restructuring before it could be run properly. This highlights the importance of providing well-structured and well-documented code to facilitate the replication and evaluation of results by the scientific community.

In general, the results of our evaluation provide strong support for the claim made by the authors and demonstrate the effectiveness of the Proto2Proto method in improving the accuracy of KD student models.

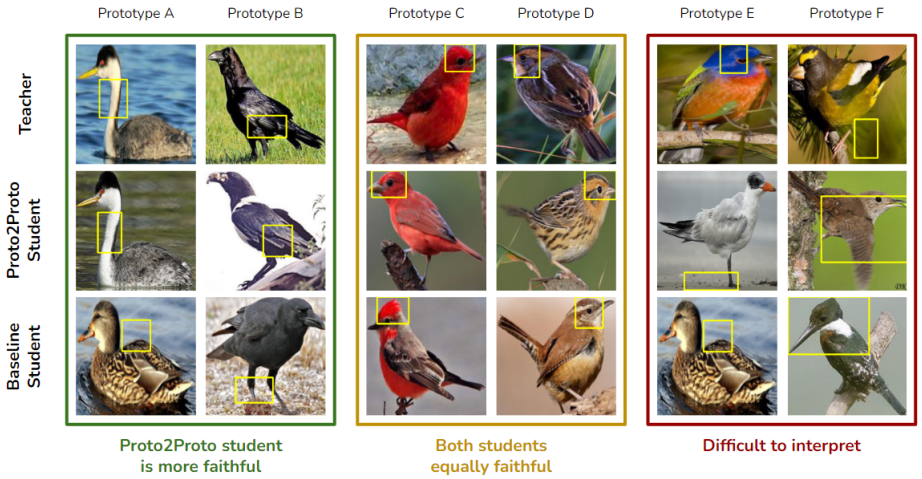

Figure 2: Prototype visualization for VGG-19 to VGG-16 experiment. The student prototypes exhibit varying degrees of similarity to the teacher model.

7.3 Similarity

Our findings do not fully support Claim 3 of the original paper, which highlights the ability of knowledge distillation to retain a faithful representation of the teacher model prototypes. Given the subjective nature of this claim, our visual analysis of the prototypes revealed instances in which both the Proto2Proto student and the baseline student exhibited varying degrees of similarity to the original teacher model. These results suggest that the faithfulness of the student model does not depend on knowledge distillation.

It is worth noting that our study was limited by available resources and computational constraints. It is possible that if the model had been trained for more epochs, the results might have been different.

In general, the visual analysis of prototypes is subjective in nature and can be affected by the presence of elements in the image patches by accident. To address this issue, the use of a more objective metric, such as the PMS metric introduced in the original paper, would be advisable to evaluate the proximity between the teacher and student prototypes.

In contrast, human judgement can be used to assess the fixation of the teacher prototypes on specific object attributes. However, a large-scale human survey would be required to objectively verify this claim, which is beyond the scope of this reproduction study.

7.4 What was easy

The author's decision to make their code publicly available proved to be a valuable asset for the reproducibility of the study. Instead of having to implement everything from scratch, we were able to run the provided code, which greatly facilitated the reproduction process within the limited time frame available. The authors' repository included instructions on how to install and run the code, although in some instances they were incorrect or outdated. Despite this, with some additional investigative work, we were able to successfully set up the codebase for training and use the appropriate dataset. Overall, the availability of the authors' code greatly streamlined the reproduction process and allowed us to focus on evaluating the results and methodology of the study.

7.5 What was difficult

Despite the authors' implementation being publicly available, it proved to be a challenging task. The lack of comprehensive documentation, combined with the highly encapsulated nature of the code, resulted in a significant amount of time spent understanding the various components and their functions. In order to evaluate the results, their code had to be restructured. In addition to this, due to missing information and computational restrictions, our results can only partially verify their claims.

7.6 Communication with original authors

In an effort to replicate the results of the original article, we attempted to obtain the trained model weights of the teacher model, the baseline student, and the KD student from the original authors of Proto2Proto. Furthermore, we requested information on the training hyperparameters used in the experiments, as they were not provided in the original article or the supplementary materials. Despite efforts to communicate by email, the original authors did not respond before the finalisation of this report.

References

[1] Chaofan Chen et al. "This Looks Like That: Deep Learning for Interpretable Image Recognition". In: **Advances in Neural Information Processing Systems**. Vol. 32. 2019. arXiv: 1806.10574.

[2] Fatima Dakalbab et al. "Artificial intelligence & crime prediction: A systematic literature review". In: **Social Sciences & Humanities Open** 6.1 (Jan. 2022), p. 100342. ISSN: 25902911. DOI: 10.1016/j.ssaho.2022.100342.

[3] Adrian Hoffmann et al. “This Looks Like That... Does it? Shortcomings of Latent Space Prototype Interpretability in Deep Networks”. In: **ICML 2021 Workshop on Theoretic Foundation, Criticism, and Application Trend of Explainable AI** (May 2021). DOI: 10.48550/arxiv.2105.02968. arXiv: 2105.02968. URL: http://arxiv.org/abs/2105.02968.

[4] Monish Keswani et al. “Proto2Proto: Can you recognize the car, the way I do?” In: **IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. Institute of Electrical and Electronics Engineers (IEEE), June 2022, pp. 10223–10233. ISBN: 9781665469463. DOI: 10.1109/cvpr52688.2022.00999. arXiv: 2204.11830. URL: https://arxiv.org/abs/2204.11830v2.

[5] Monish Keswani et al. “Supplementary Material for Proto2Proto: Can you recognize the car, the way I do?” In: 2022.

[6] Jonathan Krause et al. “3D Object Representations for Fine-Grained Categorization”. In: **Proceedings of the IEEE International Conference on Computer Vision**. 2013, pp. 554–561. ISBN: 9781479930227. DOI: 10.1109/ICCVW.2013.77.

[7] Zachary C. Lipton. “The Mythos of Model Interpretability”. In: **Queue** 16.3 (June 2018), pp. 31–57. ISSN: 1542-7730. DOI: 10.1145/3236386.3241340. URL: https://dl.acm.org/doi/10.1145/3236386.3241340.

[8] Meike Nauta, Ron van Bree, and Christin Seifert. “Neural Prototype Trees for Interpretable Fine-grained Image Recognition”. In: **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**. IEEE Computer Society, Dec. 2021, pp. 14928–14938. ISBN: 9781665445092. DOI: 10.1109/CVPR46437.2021.01469. arXiv: 2012.02046. URL: https://arxiv.org/abs/2012.02046v2.

[9] Si Shi et al. “Machine learning-driven credit risk: a systemic review”. In: **Neural Computing and Applications** 34.17 (Sept. 2022), pp. 14327–14339. ISSN: 14333058. DOI: 10.1007/s00521-022-07472-2. URL: https://dl.acm.org/doi/10.1007/s00521-022-07472-2.

[10] Catherine Wah et al. **The Caltech-UCSD Birds-200-2011 Dataset**. Tech. rep. CNS-TR-2011-001. 2011. URL: https://resolver.caltech.edu/CaltechAUTHORS:20111026-120541847.

[11] S. Kevin Zhou et al. “A Review of Deep Learning in Medical Imaging: Imaging Traits, Technology Trends, Case Studies with Progress Highlights, and Future Promises”. In: **Proceedings of the IEEE**. Vol. 109. 5. 2021, pp. 820–838. DOI: 10.1109/JPROC.2021.3054390. arXiv: 2008.09104.

A Hyperparameters

Number of prototypes	2000	
Prototype shape	(1,1,128)	
Training epochs	12	
CrossEntropy Loss weight	1.0	
ClusterSep Loss cluster weight	1.0	
ClusterSep Loss sep weight	-0.08	
L1 Loss weight	1e-4	
AddOn Loss weight	-	10
Proto Loss weight	-	10
lrNet	2e-4	
lrBlock	3e-3	
lrProto	3e-3	
lrLastLayer	1e-4	
weightDecay	1e-3	
Gamma	0.1	
Step size	4	
Step start	16	
Warm epochs	6	
Push start	12	
Optimizer	Adam	AdamW
Batch size	128	64

Table 2: Hyperparameters for training. These parameters are used for both VGG-11 and VGG-16 students.

B AAP and AJP - τ_{test} relation for VGG19 to VGG11

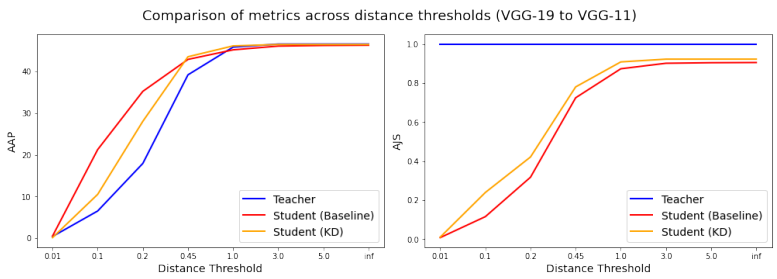

Figure 3: Effect of τ_{test} on AAP and AJS for the VGG-19 to VGG-11 experiment.

C Additional prototype visualisations

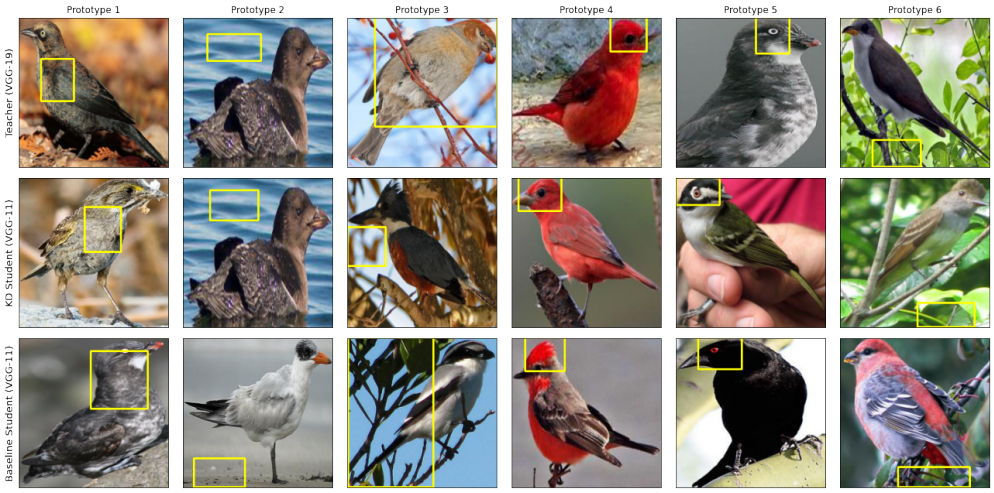

Figure 4: Comparison of sample prototypes of test images between Teacher, Baseline Student and Proto2Proto (P2P) Student for the VGG-19 to VGG-11 experiment.

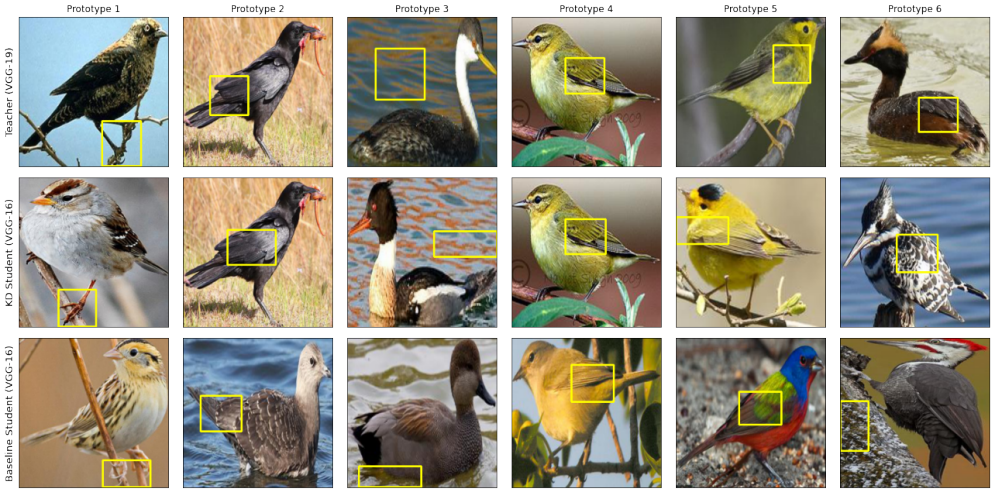

Figure 5: Comparison of sample prototypes of test images between Teacher, Baseline Student and Proto2Proto (P2P) Student for the VGG-19 to VGG-16 experiment.