1. Additional Analyses

In this section, we present a few additional analyses we omitted from the main text due to space constraints. All our analyses are conducted on iNat2019-CL and we report the performance on all novel classes in iNat2019-CL.

1.1. Semi-supervised Few-shot Learning with Coarsely-labeled Data

In the main text, we assume that no coarse labels (regardless of reference or query examples) are revealed during evaluation. However, it is also realistic to have the coarse labels of the reference examples available while the coarse labels of the query examples remain unknown. The availability of this information opens up the possibility of different inference procedures, including but not limited to semi-supervised inference approaches. To show that having a stronger representation can aid inference methods that leverage coarse labels, we develop a simple semi-supervised learning technique that utilizes the coarse label in D_{ref} during inference. Specifically, for each class k with coarse label $p(k)$, we extend its reference set as follows:

1. We construct the prototype of the class c_k
2. We randomly sample 100 examples *with the same coarse label* $p(k)$ from D_{rep}^{fine} and D_{rep}^{coarse} respectively to construct a *candidate set* of size 200.
3. Finally, we extend the reference set using the top 10 most similar examples to c_k based on cosine similarity.

Once the reference set has been extended, we use the same nearest prototype inference as in the main text. We call this method nearest-neighbor extension (NN-Ext). For comparison, we also consider another variant - NN-Ext-Any that constructs the candidate set using all examples, not just those with the same coarse label. We present the 1-shot result in table 1.

Three key observations from the table stand out. First, no matter what the representation, using the extended reference set leads to significantly better accuracy when compared to the unexpanded reference set. Second, filtering the candidate set using the coarse labels is crucial: without such filtering, accuracy actually decreases across the board. Third, among all the NN-Ext variants, the strongest is the one which uses the representation produced by PAS.

1.2. Effect of the Amounts of Coarsely-labeled Data

Given that PAS leverages additional coarsely-labeled data to yield better feature representations, its representation should perform better with more coarsely-labeled data. To verify this intuition, we keep the amount of examples from the novel-seen classes constant and vary the amount coarse label available during representation learning; examples with coarse label are pseudo-labeled with the filtered teacher's prediction whereas examples without the coarse label are pseudo-labeled with the teacher's prediction. We plot the one-shot performance of PAS in figure 1. Indeed, the coarse labels are key to PAS and PAS yields stronger representations when more coarse labels are available.

Repr.	No Extension	NN-Ext	NN-Ext-Any
Baseline	$20.46^{\pm.05}$	$26.31^{\pm.04}$	$16.02^{\pm.04}$
Repr-Coarse	$19.89^{\pm.04}$	$22.10^{\pm.03}$	$15.72^{\pm.04}$
Self-training	$22.94^{\pm.05}$	$28.83^{\pm.05}$	$17.81^{\pm.05}$
Repr-Multi	$24.72^{\pm.05}$	$29.57^{\pm.04}$	$18.73^{\pm.05}$
PAS	$25.21^{\pm.06}$	$30.98^{\pm.05}$	$19.47^{\pm.06}$
Upper Bound	$27.30^{\pm.06}$	$34.99^{\pm.05}$	$20.85^{\pm.06}$

Table 1. Average 1-shot Top-1 Per Class Accuracy and 95% confidence interval of NN-Ext and NN-Ext-Any across 1000 runs.

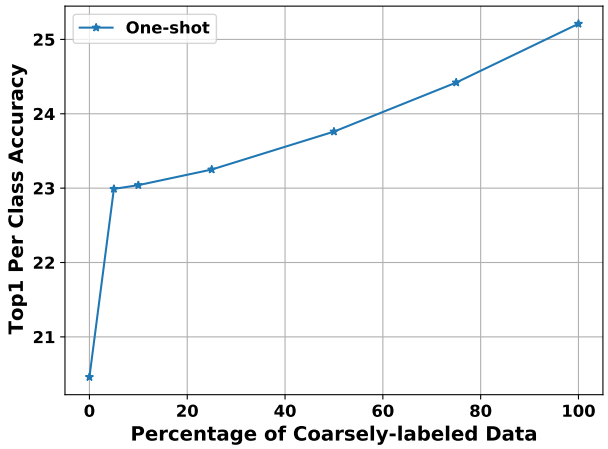

Figure 1. Average 1-shot performance of PAS trained with various amount of coarse labels.

1.3. Effect of Teachers with Stronger Representation

Given the superiority of Repr-Multi to the Baseline, one might consider substituting the single-tasked teacher used in PAS with a multi-tasked model. To experiment with this idea, we replace the teacher in PAS with the classification model trained to obtain Repr-Multi and report the performance of the resulting student representation in table 2. The performance of the students do not differ much so we opt to use the single-tasked teacher for simplicity.

2. Full Results

We omitted the confidence intervals when reporting several results in the main text for brevity so we intend to report those results with confidence intervals in this section. Table 5 corresponds to table 2 in the main text, in which we report

Teacher	k=1	k=5	all
Original	$25.21^{\pm.05}$	$43.27^{\pm.03}$	$61.04^{\pm.00}$
Multi-tasked	$25.42^{\pm.05}$	$43.65^{\pm.03}$	$61.11^{\pm.00}$

Table 2. Average k-shot Accuracy and 95 % confidence intervals of PAS with different teachers.

the comparison between PAS and various representations. Table 3 supplements table 4 in the main text in which we investigate large reduction of base classes. Table 4 supplements table 6 in the main text in which we investigate the effect of unseen supercategories.

Large Reduction in Base Classes		
Method	k=1	5
Baseline - Original	$20.46^{\pm.05}$	$39.22^{\pm.03}$
PAS- Original	$25.21^{\pm.05}$	$43.27^{\pm.03}$
Baseline	$14.19^{\pm.03}$	$28.38^{\pm.02}$
PAS	$21.80^{\pm.05}$	$37.31^{\pm.03}$

Table 3. Average k-shot performance of different representations evaluated on the original iNat2019-CL novel classes. PAS- Original and Baseline - Original are trained on the original base dataset. This table supplements table 5 in the main text.

Removing Some Coarse Labels		
Method	k=1	5
Baseline	$18.15^{\pm.04}$	$35.72^{\pm.03}$
Repr-Multi	$21.92^{\pm.05}$	$37.23^{\pm.03}$
PAS	$23.14^{\pm.05}$	$40.73^{\pm.03}$

Table 4. Average k-shot performance (on iNat2019-CL novel classes) of various representations trained on a base dataset with unseen supercategories. This table supplements table 6 in the main text.

3. Training and Architectural Details

In this section, we describe the architecture used and the training details of all the models in the paper.

3.1. Backbone Architecture

As mentioned in the main text, we use ResNet18 [2] as our default backbone. The original ResNet18 is designed for input images of resolution 224x224. Given that the resolution of images in our datasets are smaller (84x84 and 32x32), we modify the first convolutional layer by changing the kernel size to 3, stride to 1 and padding to 1. We remove the first max pooling layer to avoid downsampling of the images and the last ReLU activation as suggested in [1].

3.2. Training Details

3.2.1 Representation Learning

In this section, we describe the training details used to train all representations used in the main paper (i.e., Baseline, Repr-Coarse, Self-training, Repr-Multi, Upper Bound). We trained all representations using SGD with momentum 0.9 and weight decay set to 5e-4. In the first epoch of training, we linearly increase the learning rate from zero to 0.1 and then drop the learning rate by a factor of 10 every one-third of the total number of training epochs. Models for iNat-2019-CL (or its variants) and CIFAR-100-CL are trained for 300 epochs and models for tieredImageNet-CL are trained for 90 epochs. All models are initialized randomly (including both teachers and students). We augment the training images with the random crop and random horizontal flip when training the models.

3.2.2 FEAT and MetaOptNet

We modified the official implementation of MetaOptNet [1] and FEAT [2] released by the authors. Below are some modifications we made:

MetaOptNet. We replaced the backbone architecture to the ResNet18 architecture in section 3.1. We did not remove the last ReLU activation and removed global averaging pooling to adopt similar architecture used in the original implementations. We changed the number episodes per update to 4 for iNat2019-CL and tieredImageNet-CL due to hardware constraints.

FEAT. We replaced the backbone with the ResNet18 in 3.1 and did not remove the last ReLU activation. To pretrain the backbone, we used linear classifier (following the original implementation) and the same training procedures in 3.2.

Regardless of the datasets, we trained all the models using the hyperparameters the author reported for the original tieredImageNet and did not do hyperparameter tuning.

[1] MetaOpt: https://github.com/kjunelee/MetaOptNet
[2] FEAT: https://github.com/Sha-Lab/FEAT

	iNat2019-CL								
	Novel			Novel-seen			Novel-unseen		
Method	k=1	5	all	k=1	5	all	k=1	5	all
Baseline	$20.46^{\pm.05}$	$39.22^{\pm.03}$	$57.22^{\pm.00}$	$28.68^{\pm.08}$	$50.68^{\pm.04}$	$67.25^{\pm.00}$	$28.14^{\pm.08}$	$50.37^{\pm.04}$	$67.49^{\pm.00}$
Repr-Coarse	$19.89^{\pm.04}$	$29.32^{\pm.03}$	$41.72^{\pm.00}$	$33.50^{\pm.07}$	$44.62^{\pm.04}$	$57.62^{\pm.00}$	$28.09^{\pm.07}$	$40.32^{\pm.04}$	$51.39^{\pm.00}$
Self-training	$22.94^{\pm.05}$	$42.17^{\pm.03}$	$59.69^{\pm.00}$	$33.18^{\pm.09}$	$54.79^{\pm.04}$	$69.85^{\pm.00}$	$29.95^{\pm.08}$	**52.11**$^{\pm.04}$	**69.87**$^{\pm.00}$
Repr-Multi	$24.72^{\pm.05}$	$41.42^{\pm.03}$	$57.34^{\pm.00}$	$38.24^{\pm.09}$	$56.77^{\pm.04}$	$70.72^{\pm.00}$	**32.03**$^{\pm.09}$	$51.21^{\pm.04}$	$65.88^{\pm.00}$
PAS	**25.21**$^{\pm.05}$	**43.27**$^{\pm.03}$	**61.04**$^{\pm.00}$	**39.06**$^{\pm.09}$	**58.76**$^{\pm.04}$	**73.63**$^{\pm.00}$	$30.91^{\pm.08}$	$51.85^{\pm.04}$	$69.12^{\pm.00}$
Upper Bound	$27.30^{\pm.06}$	$47.98^{\pm.03}$	$64.20^{\pm.00}$	$41.64^{\pm.10}$	$64.61^{\pm.03}$	$75.36^{\pm.00}$	$30.71^{\pm.09}$	$53.77^{\pm.04}$	$72.29^{\pm.00}$

	tieredImageNet-CL								
	Novel			Novel-seen			Novel-unseen		
Method	k=1	5	all	k=1	5	all	k=1	5	all
Baseline	$32.16^{\pm.09}$	$53.36^{\pm.04}$	$68.97^{\pm.00}$	$41.22^{\pm.15}$	$62.92^{\pm.06}$	$77.19^{\pm.00}$	$54.19^{\pm.19}$	$75.50^{\pm.06}$	$85.51^{\pm.00}$
Repr-Coarse	$25.69^{\pm.07}$	$37.19^{\pm.04}$	$49.76^{\pm.00}$	$38.14^{\pm.11}$	$48.83^{\pm.07}$	$62.55^{\pm.00}$	$41.64^{\pm.16}$	$55.70^{\pm.08}$	$66.32^{\pm.00}$
Self-training	$35.49^{\pm.10}$	$57.26^{\pm.04}$	$70.87^{\pm.00}$	$48.12^{\pm.16}$	$69.11^{\pm.06}$	**80.60**$^{\pm.00}$	**54.71**$^{\pm.19}$	**75.89**$^{\pm.06}$	**86.08**$^{\pm.00}$
Repr-Multi	$37.16^{\pm.10}$	$57.27^{\pm.04}$	$70.20^{\pm.00}$	$49.54^{\pm.16}$	$68.38^{\pm.07}$	$80.28^{\pm.00}$	$53.28^{\pm.19}$	$72.94^{\pm.07}$	$83.31^{\pm.00}$
PAS	**38.11**$^{\pm.10}$	**59.08**$^{\pm.04}$	**71.84**$^{\pm.00}$	**50.60**$^{\pm.16}$	**69.52**$^{\pm.07}$	$80.40^{\pm.00}$	$53.18^{\pm.19}$	$74.68^{\pm.07}$	$85.12^{\pm.00}$
Upper Bound	$42.86^{\pm.11}$	$65.68^{\pm.04}$	$76.71^{\pm.00}$	$60.03^{\pm.18}$	$80.67^{\pm.05}$	$87.14^{\pm.00}$	$55.94^{\pm.20}$	$76.96^{\pm.06}$	$86.55^{\pm.00}$

	CIFAR-100-CL								
	Novel			Novel-seen			Novel-unseen		
Method	k=1	5	all	k=1	5	all	k=1	5	all
Baseline	$20.32^{\pm.09}$	$33.24^{\pm.05}$	$42.67^{\pm.00}$	$25.50^{\pm.13}$	$39.95^{\pm.07}$	$50.45^{\pm.00}$	$34.37^{\pm.22}$	$51.80^{\pm.10}$	$64.00^{\pm.00}$
Repr-Coarse	$31.56^{\pm.10}$	$38.90^{\pm.06}$	$47.87^{\pm.00}$	$45.74^{\pm.13}$	$53.36^{\pm.08}$	$63.10^{\pm.00}$	$37.52^{\pm.25}$	$50.65^{\pm.10}$	$55.20^{\pm.00}$
Self-training	$25.68^{\pm.10}$	$42.43^{\pm.06}$	$54.93^{\pm.00}$	$32.96^{\pm.14}$	$51.42^{\pm.08}$	$63.30^{\pm.00}$	$38.24^{\pm.22}$	**57.51**$^{\pm.10}$	**69.50**$^{\pm.00}$
Repr-Multi	**34.99**$^{\pm.12}$	$46.30^{\pm.06}$	$55.07^{\pm.00}$	**49.18**$^{\pm.16}$	$60.51^{\pm.09}$	$69.20^{\pm.00}$	**39.00**$^{\pm.24}$	$53.69^{\pm.09}$	$61.20^{\pm.00}$
PAS	**35.00**$^{\pm.11}$	**48.42**$^{\pm.06}$	**58.37**$^{\pm.00}$	$48.57^{\pm.15}$	**61.95**$^{\pm.08}$	**72.65**$^{\pm.00}$	$37.92^{\pm.22}$	$54.91^{\pm.10}$	$65.10^{\pm.00}$
Upper Bound	$51.83^{\pm.17}$	$64.97^{\pm.04}$	$69.17^{\pm.00}$	$73.75^{\pm.22}$	$85.02^{\pm.02}$	$85.45^{\pm.00}$	$36.53^{\pm.23}$	$56.25^{\pm.10}$	$70.30^{\pm.00}$

Table 5. Average top-1 per class accuracy and 95% confidence intervals of various representations across 1000 runs. For each novel categories, we use k=1, 5 and all reference examples. This table supplements table 2 in the main text.

References

[1] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018. 2

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2