Regularizing CNN Transfer Learning with Randomised Regression

Yang Zhong Atsuto Maki
Division of Robotics, Perception and Learning
KTH Royal Institute of Technology
{yzhong, atsuto}@kth.se

1. The scales of the datasets used in the paper

Our paper demonstrates and discusses a ConvNet regularizer which works without extra knowledge or data (unlike [3, 7, 6, 2]) on several transfer learning tasks for image classification. These tasks cover different scenarios of transfer learning including fine-grained object classification and close-set human face identification. They contain more sparse training samples than some commonly studied transfer learning tasks, e.g. CIFAR [4] and MNIST [5]. The total number of classes and the number of training sampler per class of these datasets are compared in Figure 1. It can be seen that the transfer learning tasks we study in this paper feature either a few samples per class or many categories with limited training samples; the total training samples are fewer than 6000 for the selected datasets except "CelebA-500" while its average training samples per class is lower than most of the selected datasets.

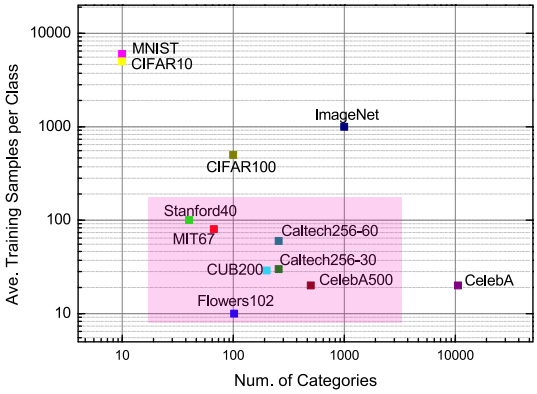


Figure 1: *The scales of datasets used in our experiments (in the magenta shadowed area) among other popular transfer learning benchmarks, e.g. CIFAR10 and MNIST, as well as the commonly used source datasets such as ImageNet [1] and CelebA datasets [8]. The average number of per-class training instances (vertical-axis) and the total number of classes (horizontal-axis) of each dataset are shown in $\log_{10}$ scale.*

2. Timing to bring in PtR

The pseudo regression task is brought into the training pipeline at a proper stage, and the timing was settled by considering the training efficiency and model effectiveness. That is, our pseudo-task branch is enabled to jointly train the network when the entire network is considered as relatively close to convergence [1].

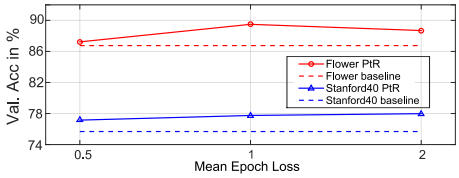


Figure 2: *Validation accuracy (in %) on Flower and Stanford40, according to different timing when PtR joins the network training on the course of model convergence using a varying threshold, T, for the mean epoch cross-entropy loss (with SML1 regression).*

As shown in Figure 2, we experimented with a varying mean epoch training loss (T) on Flower102 and Stanford40 to determine a practical one for efficient training and models' effectiveness. It can be observed that PtR improves on the baseline accuracies (fine-tuning) in all cases and the PtR at T=1 resulted in a higher accuracy on average. We chose T=1 as a practical setting in all the rest of experiments since a larger T does not contribute to the model effectiveness while negatively prolonging the training time.

3. Standard Deviations of PtR with ResNet

The average classification accuracies (in %) with standard deviations of the pseudo-task regularization (PtR) with the ResNet architectures are given in Table 1 to provide more details for Section 3 in the main manuscript to compare to [2, 6].

[1] This is different from multiple training stages as in [7] where the additional imposed regularizer starts to train the network only after the target classifier is converged on the target task.

Table 1: The ResNet test accuracies, with standard deviations given in parenthesis, on the datasets used in our work.

	ResNet-50		ResNet-101	
	Baseline	PtR	Baseline	PtR
CUB200	80.3% (0.19)	81.9% (0.23)	-	-
Flower102	91.0% (0.32)	91.8% (0.22)	90.6% (0.17)	91.6% (0.26)
MIT67	77.4% (0.72)	77.9% (0.65)	78.7% (1.10)	77.9% (0.63)
Caltech256-30	-	-	84.0% (0.15)	84.5% (0.19)
Caltech256-60	-	-	86.8% (0.18)	87.2% (0.07)

4. Impact of weight decay

Weight decay may have some observable impact on the performance of the fine-tuning baseline, but it can hardly impact the performance of PtR. We collect all the related results and show in Table 2 [2].

Table 2: Impact of weight decay in fine-tuning (FT) and PtR (ResNet50). "w/" and "wo/" stands for with default weight decay settings and without using weight decay respectively.

	FT w/	FT **wo/**	PtR w/	PtR **wo/**
CUB200	80.3%	81.0%	81.9%	82.0%
Caltech256-30	83.2%	83.2%	83.9%	83.9%
Caltech256-60	86.6%	86.7%	87.1%	87.0%

5. Training from scratch

We have also evaluated the PtR when training models from scratch as shown in Table 3. Comparing the regularization gain in Table 2 and Table 5 in the main paper, it can be observed that PtR demonstrates stronger regularization compared to the scenarios where the ImageNet initialization was used. It can also be noticed that the gain on the Caltech256 is consistently smaller than that of the CUB200 in both scenarios. Considering the slightly larger category on the Caltech256 (see Figure 1), it suggests that training with more real samples leaves lesser room for PtR to improve regularization.

Table 3: Comparing test accuracy of fine-tuning (FT) baseline and PtR with ResNet50 on the CUB200 and Caltech256-30 when training from scratch. The absolute performace gain in % is listed in the last column for ease of comparison.

	FT	PtR	Gain
CUB200	57.33%	61.71%	4.38%
Caltech256-30	46.21%	48.25%	2.04%

[2] All the numerical results in the supplementary document were averaged from five independent runs.

6. Disentangle batch normalization

Batch normalization has been commonly used as a regularization module in network training. To explore the interaction of its regularization effect with PtR, we applied it to off-the-shelf VGG-16 models with batch normalization layers. It is a good candidate for this study is because it features a more uniform convolutional-pooling structure that facilitates disentangling other factors (compared to ResNet for example) and is sufficiently deep. The comparative results are shown in Table 4.

Comparing to Table 2 in the main paper, it can be found that batch normalization improved the baseline for around 1.6%. The use of batch normalization seems to improve the PtR: together with batch normalization, PtR achieved an extra 1% regularization gain compared to that in Table 2.

Table 4: Comparing PtR to fine-tuning (FT) on the CUB200 using the ImageNet pre-trained VGG-16 with batch normalization layers.

	FT	PtR	Gain
CUB200	76.70%	80.73%	4.03%

7. Case studies on the effect of the Psudo-task Regularization (PtR)

We provide ten examples (cf. Section 5 of the paper) with details as to how PtR impacts the probability predictions in comparison to the corresponding responses from the baseline networks trained by the standard finetuning. Specifically, we compare how the predictions of PtR differ from the baseline in four scenarios:

1. PtR correctly rectifies finetuning's errors (i.e., true rectification);

2. PtR gives wrong predictions while the predictions by finetuning were right (i.e., false rectification);

3. Both PtR and finetuning give correct predictions (i.e., both correct);

4. Neither PtR nor finetuning gives correct predictions (i.e., both wrong).

Figure 3, 4, 5, and 6 show the case studies corresponding to each scenario, respectively. In each of them, input images following the scenario are shown at the left most, with the ground truth class labels on the top. Next to those, the raw probabilities given by PtR and finetuning are displayed in the grid. The predictions by the two methods are then sorted according to the magnitude, respectively, and we show the sorted probabilities which are larger than a small value of 0.5%, which we call major probabilities, in the third grid. The predicted labels (both by finetuning and PtR) are given in the legend, and the entropy of each distribution is provided on top. "Finetuning" and "predictions" are denoted as "FT" and "Pred", respectively.

In the right most, four images from the training set are shown for reference: two of them come from the rank-1 predicted class of PtR and another two images come from the second predicted class. This allows us, by comparing those samples and the sorted probabilities, to subjectively observe how the visual similarities have been valued in the predictions by PtR more easily.

Throughout the case studies demonstrated in the figures, an important characteristic of PtR can be found: with PtR the network tends to identify fewer classes as being similar to the input class. In particular, many of the minor probabilities (predicted as low values) are eliminated compared to the original distributions from the finetuning baseline. As a result, the predictions of PtR are in general less noisy than those by finetuning. We suspect that noisy predictions by finetuning are caused by overfitting; our case studies suggest that PtR is capable of alleviating such a problem in a transfer learning scenario. Besides, PtR often encourages the predictions of visually similar classes in various scenarios.

As well as the qualitative case studies provided so far, we also summarize some statistics of our case studies in addition to Section 5 in the main manuscript. The validation set of CUB200 contains 584 randomly selected images. The classification accuracy for finetuning and PtR are 79.8% and 80.8%, respectively. The number of true rectification is 27 and the number of false rectification is 21. PtR and finetuning make correct predictions on a common set of 445 images and they both make wrong predictions on 91 images.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li. Imagenet: A large-scale hierarchical image database. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[2] A. Dubey, O. Gupta, P. Guo, R. Raskar, R. Farrell, and N. Naik. Pairwise confusion for fine-grained visual classification. In *European Conference on Computer Vision (ECCV)*, 2018.

[3] W. Ge and Y. Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

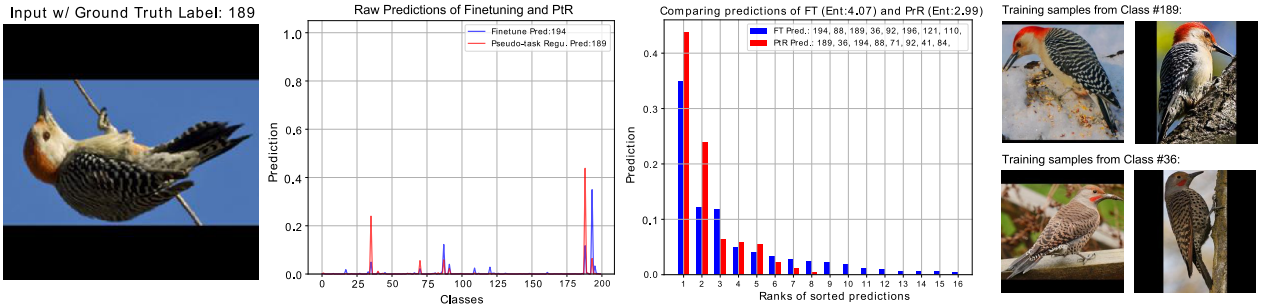
[4] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[5] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

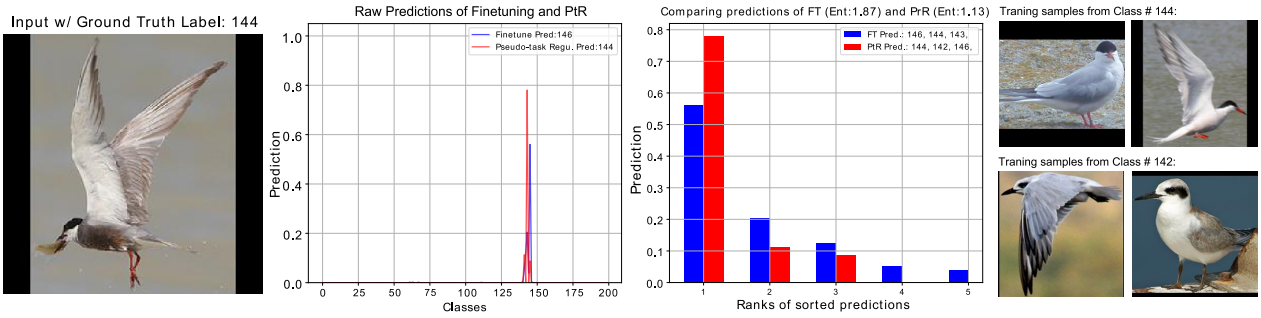
[6] X. Li, Y. Grandvalet, and F. Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning (ICML)*, 2018.

[7] Z. Li and D. Hoiem. Learning without forgetting. In *European Conference on Computer Vision (ECCV)*, 2016.

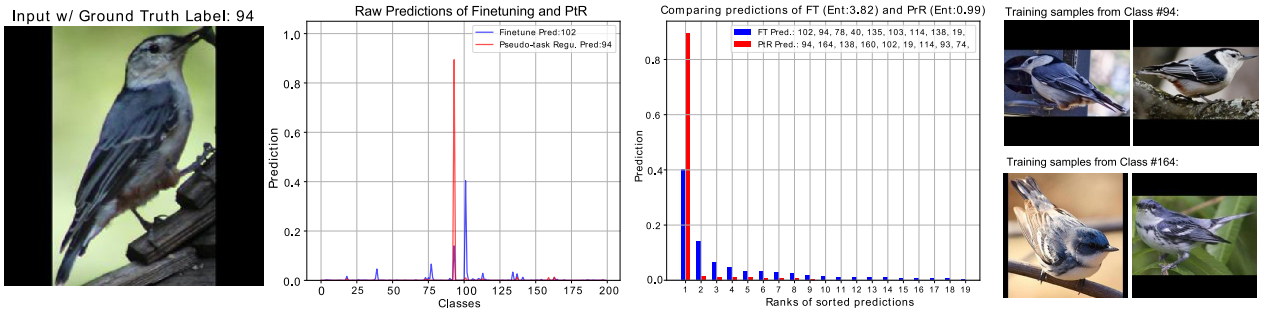
[8] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015.



(a) *PtR gives correct rank-1 class prediction and it is more certain about it than FT is about the misclassified result for the displayed sample. In addition, PtR also gives a relatively high score on the second prediction which is a class quite similar to the ground truth (compare patterns on the wings and colors of head of the training examples from Class 36 to Class 189). Only eight predicted major probabilities (those higher than 0.5%) are made while finetuning generates twice as many; the entropy of PtR turned out to be less than that of FT.*

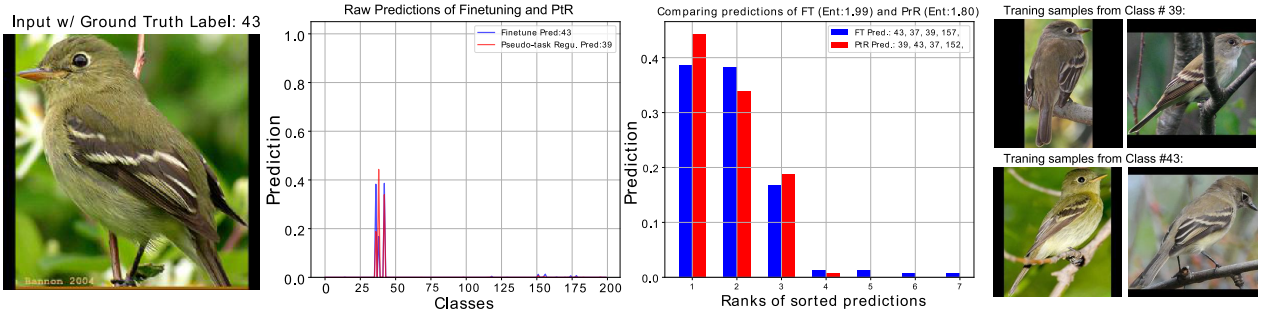


(b) *PtR is quite confident with rank-1 prediction and gives lower but still significant scores to visually similar classes as a result. Similarly to the example in (a), major prediction scores are given to only three classes by PtR whereas FT picks two more classes. The entropy of PtR is also lower than that of FT.*

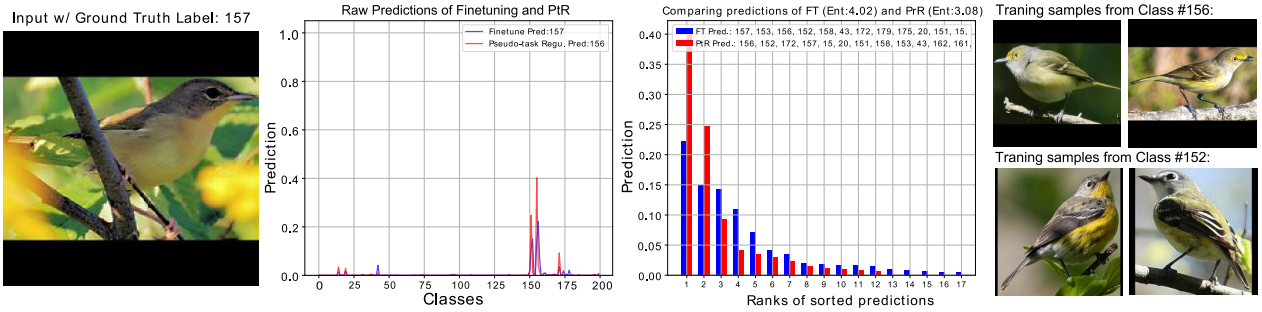


(c) *PtR gives the highest prediction score to the input image (among all of the three examples in this True Rectification scenario) and the prediction scores on other classes are quite marginal. But it can also be seen that PtR still tends to identify fewer similar classes given that only eight other classes receive a score around 1% while finetuning selects ten more classes.*

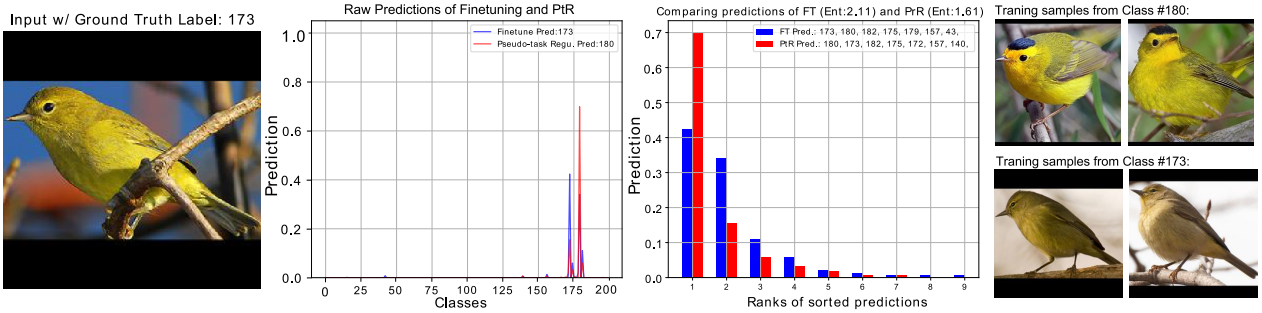
Figure 3: *Examples of True Rectifications.*



(a) *This is a hard positive example to finetuning on which PtR makes an error. From the predictions (see the 3rd grid), it can be seen that both methods identify the similar classes — Class 37, 39, and 43; FT generates a slightly higher score on the correct class. PtR again focuses on only four classes and the entropy of the prediction scores is slightly lower.*

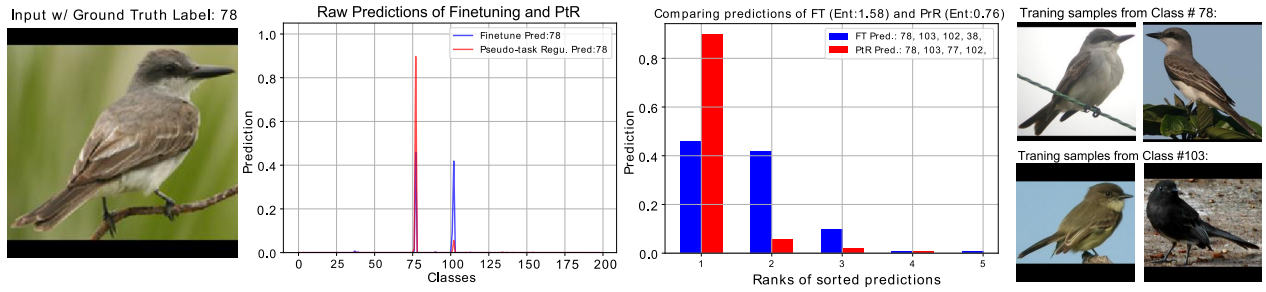


(b) *PtR gives more confidence than FT does for the top-2 predictions and it consequently assigns lower scores on other similar categories. The number of major predictions are fewer than that of FT. In this way, it can focus on a few similar classes and generates predictions with a lower entropy.*

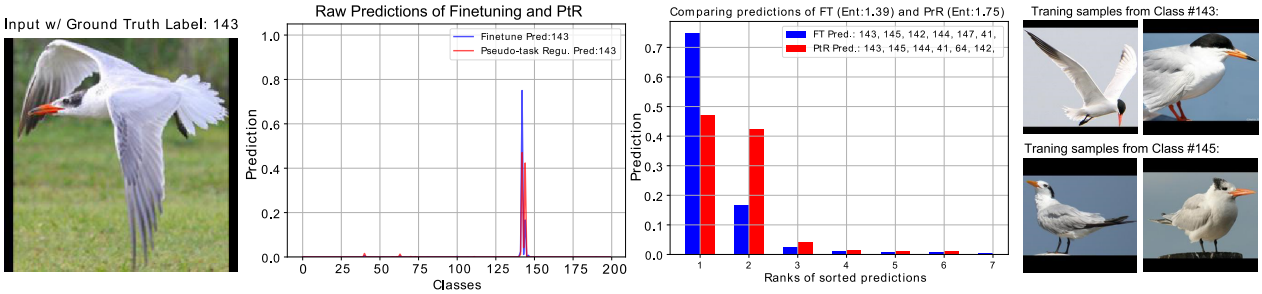


(c) *Although PtR gives high confidence on a similar but wrong class, it still identifies the correct class in the rank-2 prediction. The number of major predictions are slightly fewer than that of FT and they are less noisy as indicated by the lower entropy.*

Figure 4: *Examples of False Rectifications.*

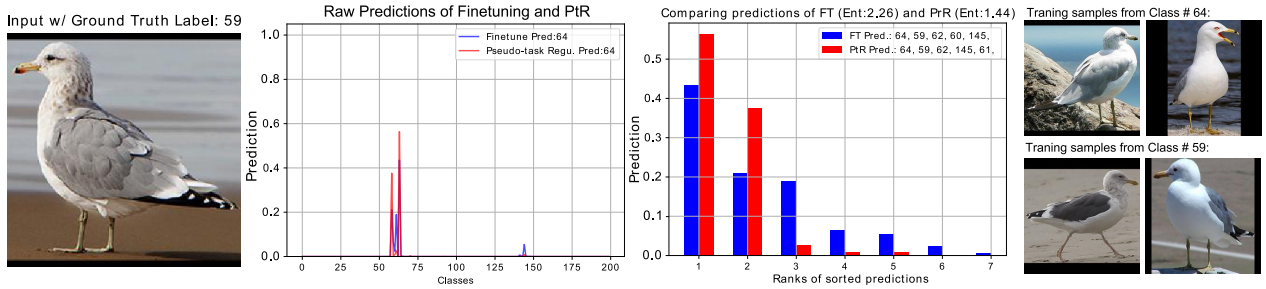


(a) *Another hard positive example for FT. It can be seen that the predictions made by PtR are more discriminative than those by FT, but at the same time PtR can still correctly identify the visually correlated Class 103. The number of major predictions of PtR is fewer than that of FT; the entropy of the PtR's prediction is also lower than FT.*

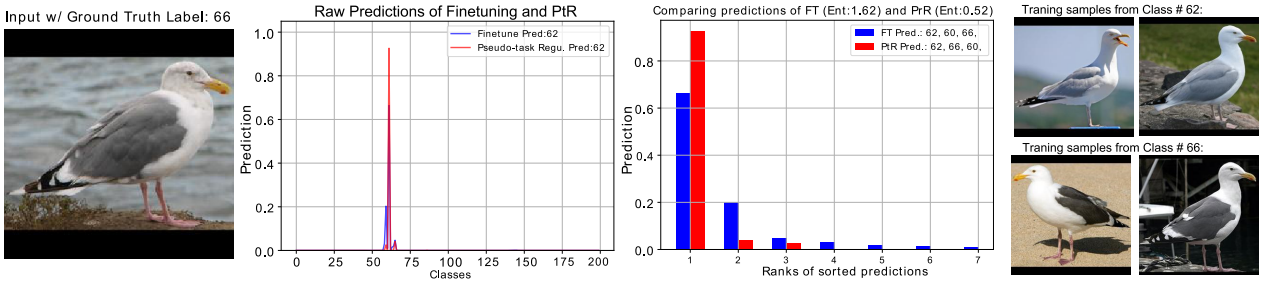


(b) *Although PtR still correctly identifies the object, the close scores (of rank-1 and rank-2 predictions) suggest that it also suspects the rank-2 predicted class. This means that the model trained by PtR has focused mostly on the first two predicted classes. The predictions of PtR are slightly noisier than FT, but the number of major probabilities is one fewer than that of FT.*

Figure 5: *Examples of Correct Predictions by Both Methods.*



(a) *Both FT and PtR make correct predictions at the second rank. PtR still predicts fewer minor probability and makes less noisy predictions than FT.*



(b) *Although both rank-1 predictions are wrong, PtR correctly identifies the right class at the rank-2 prediction. As PtR gives confidence in its rank-1 prediction, the rest of the identified classes receive lower scores. PtR in this case still produces fewer minor probabilities than FT.*

Figure 6: *Examples of Wrong Predictions by Both Methods.*