## A. Vision-only long-tail learning

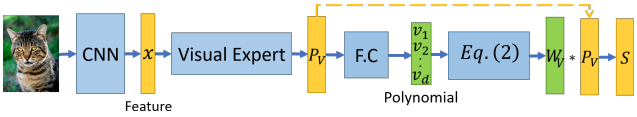


Fig. S 7: Architecture of vision-only smDRAGON.

To understand the effect of re-calibration we now study a simpler variant of DRAGON that can be applied to the more common vision-only long-tail learning. We name it smDRAGON for *single-modality*-DRAGON, and show that it achieves new state-of-the-art results, compared to uni-modals baselines, on ImageNet-LT, Places365-LT, CIFAR-10 and CIFAR-100. On iNaturalist smDRAGON is comparable to SoTA. To adapt to single-modality, we train smDRAGON only on the predictions of the visual-expert (VE). It outputs a single set of coefficients $\{\mathbf{w}_V(y)\}_{y\in\mathcal{Y}}$ to rescale the predictions of the visual expert, instead of two sets of coefficients. Subsequently, Eq. (1) reduces to $S(y) = \mathbf{w}_V(y)p_V(y)$.

In other words, smDRAGON is a simplified version of DRAGON that is trained on the predictions of the **visual-expert only** (no class descriptors being used). smDRAGON takes the predictions of a freezed visual-expert and rescale it by learning a single set of polynomial coefficients. During inference, smDRAGON balances the visual-expert predictions in a sample-by-sample basis.

Tables 8 and 9 compare smDRAGON against approaches in the unbalanced CIFAR-10 and CIFAR-100, as presented in [7]: **CE Loss**, **Resample [11]**, **Reweight [11]**, **Focal [11]** and **LDAM Loss [7]**. *DRW* [7] denotes models that were trained with the training schedule proposed by [7].

Table 10 compares smDRAGON with popular baselines and recent long-tail learning approaches in the ImageNet-LT and Places365-LT benchmarks. Those are the same baselines as in the Smooth-Tail setup (Section 6).

The results demonstrate that (1) smDRAGON outperforms all baselines and (2) combining smDRAGON with SoTA approaches (LDAM or DRW) has a synergistic effect.

Table 11 compares DRAGON with smDRAGON on CUB-LT. It shows that fusing information between modalities (third row) gives better results than re-scaling expert predictions alone (first and second rows).

Finally, on iNaturalist 2018, smDRAGON is comparable to SoTA, reaching 69.1% compared to 69.5% (CB-LWS [23]). Table S12 compares smDRAGON against long-tail learning approahces on iNatrualist2018 [20]. iNatualist2018 is a real-world large-scale long-tailed dataset which has 437,513 training images with a total of 8,142 classes.

| VISION-ONLY | UNB. CIFAR-10 | | | | UNB. CIFAR-100 | | | |
|---|---|---|---|---|---|---|---|---|
| IMBALANCE TYPE | LONG-TAIL | | TWO-LEVEL | | LONG-TAIL | | TWO-LEVEL | |
| IMBALANCE RATIO | 100 | 10 | 100 | 10 | 100 | 10 | 100 | 10 |
| [11] RESAMPLE | 29.4 | 13.2 | 38.1 | 15.4 | 66.56 | 44.9 | 66.2 | 46.9 |
| [11] REWEIGHT | 27.6 | 13.5 | 38.1 | 16.2 | 66.0 | 42.9 | 78.7 | 47.5 |
| [11] FOCAL | 25.4 | 12.9 | 39.7 | 16.5 | 64.0 | 42.0 | 80.2 | 50.0 |
| CE [7] | 29.6 | 13.6 | 36.7 | 17.5 | 61.7 | 44.3 | 61.4 | 45.4 |
| CE* **(VE)** | 29.8 | 13.1 | 36.6 | 17.8 | 61.7 | 43.8 | 61.6 | 45.7 |
| FOCAL [27] | 29.6 | 13.3 | 36.1 | 16.4 | 61.6 | 44.2 | 61.4 | 46.5 |
| LDAM [7] | 26.6 | 13.0 | 33.4 | 15.0 | 60.4 | 43.1 | 60.4 | 43.7 |
| **SMDRAGON (OURS)** | **22.1** | **12.2** | **27.1** | **12.4** | **58.0** | **42.2** | **54.4** | **41.0** |

Table S 8: **Vision-only long-tail**. Error rate of ResNet32 on unbalanced CIFAR-10 and CIFAR-100 [7], comparing smDRAGON and SoTA techniques. smDRAGON was trained over predictions of the cross-entropy model (CE*). Reported values are the top-1 validation error. Asterisks * denote results reproduced using published code.

| VISION-ONLY | UNB. CIFAR-10 | | | | UNB. CIFAR-100 | | | |
|---|---|---|---|---|---|---|---|---|
| IMBALANCE TYPE | LONG-TAIL | | TWO-LEVEL | | LONG-TAIL | | TWO-LEVEL | |
| IMBALANCE RATIO | 100 | 10 | 100 | 10 | 100 | 10 | 100 | 10 |
| CE-DRW* **(VE1)** | 24.7 | 13.5 | 28.6 | 13.9 | 59.2 | 42.2 | 58.9 | 45.0 |
| M-DRW [15] | 24.9 | 13.6 | 26.7 | 13.2 | 59.5 | 43.5 | 58.9 | 44.7 |
| LDAM-DRW [7] | 23.0 | 11.8 | 23.1 | 12.2 | 58.0 | 41.3 | 54.6 | 40.5 |
| LDAM-DRW* [7] **(VE2)** | 23.0 | 11.8 | 23.4 | 12.2 | 57.9 | 41.6 | 54.6 | 43.5 |
| **VE1 + SMDRAGON** | **20.4** | 12.1 | 21.5 | **11.9** | **56.5** | 42.11 | **53.3** | 40.6 |
| **VE2 + SMDRAGON** | 21.2 | **11.7** | **20.6** | 12.3 | 56.7 | **41.2** | 54.0 | **40.3** |

Table S 9: **Vision-only long-tail:** smDRAGON was trained on top models trained with DRW (VE1) or LDAM-DRW (VE2) [7]. Similar to Table 8 except that all models were trained with DRW schedule [7]. Reported values are top-1 validation error. Asterisks * denote results that we reproduced using code published by the authors of [7].

| Vision-Only | **Places365-LT** | **ImageNet-LT** | |
|---|---|---|---|
| | ResNet-50 | ResNet-10 | ResNeXt-50 |
| CE Loss* **(VE)** | 30.2 | 34.8 | 44.4 |
| Bal' Loss | 32.4 | 33.1 | - |
| Lifted Loss [41] | 35.2 | 30.8 | - |
| Focal Loss [27] | 34.6 | 30.5 | - |
| Range Loss [57] | 35.1 | 30.7 | - |
| FSLwF [16] | 34.9 | 28.4 | - |
| OLTR [29] | 35.9 | 34.1 | 37.7 |
| CB $\tau-$norm [23] | 37.9 | 40.6 | 49.4 |
| CB LWS [23] | 37.6 | 41.4 | 49.9 |
| **smDRAGON (ours)** | **38.1** | **42.0** | **50.1** |

Table S 10: **Vision-only long-tail learning:** smDRAGON achieves better $Acc_{PC}$ on Places365-LT and ImageNet-LT.

## B. Additional analysis of the familiarity effect

Here we provide a deeper analysis showing that DRAGON effectively addresses the "familiarity bias".

**The familiarity bias causes models to incorrectly favor head classes:** Figure S8(a) shows the confusion matrix

| | $Acc_{PC}$ | $Acc_{LT}$ |
|---|---|---|
| VISUAL EXPERT + SMDRAGON | 55.8 | 66.0 |
| SEMANTIC EXPERT + SMDRAGON | 57.7 | 63.4 |
| **DRAGON (OURS)** | **60.1** | **67.7** |

Table S 11: Ablation study, comparing SMDRAGON to DRAGON on CUB-LT: Fusing information between modalities improves performance (test set, CUB-LT).

| Vision-Only | **iNaturalist** ResNet-50 |
|---|---|
| [11] Focal | 61.1 |
| LDAM [7] | 64.6 |
| LDAM-DRW [7] | 68.0 |
| CB $\tau-$norm [23] | 69.3 |
| CB LWS [23] | 69.5 |
| **smDRAGON (ours)** | 69.1 |

Table S 12: **Vision-only long-tail:** Comparing SMDRAGON on long-tailed iNatrualist. Baseline results where copied directly from [7] and [23].

of a standard ResNet-101 trained on CUB-LT, as computed on the validation set. Classes, of the confusion matrix, are ordered by a decreasing number of training samples, with class #1 having many samples and class #200 have few samples. Black dots denote count larger than 15.

It illustrates two effects. First, the trained model correctly classifies head classes, based on the fact that the top rows have no incorrect (off-diagonal) predictions. Second, for mid and tail classes, predictions are clearly biased towards the head, since there are many more off-diagonal predictions to the left (head class predictions).

**DRAGON corrects for the familiarity bias:** Figures S8(b) and S8(c) demonstrate that DRAGON learns to offset the familiarity bias. The left panel (b) shows the familiarity effect on CUB-LT before recalibration. The right panel (c) shows that DRAGON corrects the familiarity bias and produces a more balanced average confidence across the head and tail classes.

**DRAGON re-calibrate predictions:** In the main paper (Figure 1(c)) we showed that a model that is trained on unbalanced data has higher confidence for head classes and it over-estimate them. **By reversing the familiarity bias, smDRAGON, implicitly, also re-calibrate experts predictions.** Figure 9 compares the reliability diagrams for smDRAGON against raw *ResNeXt-152*, *Temp Scaling [17]* and *Dirichlet Calibration [25]* (common and SoTA calibration approaches). We report both per-class-accuracy (ACC) and expected-calibration-error (ECE) for each model.

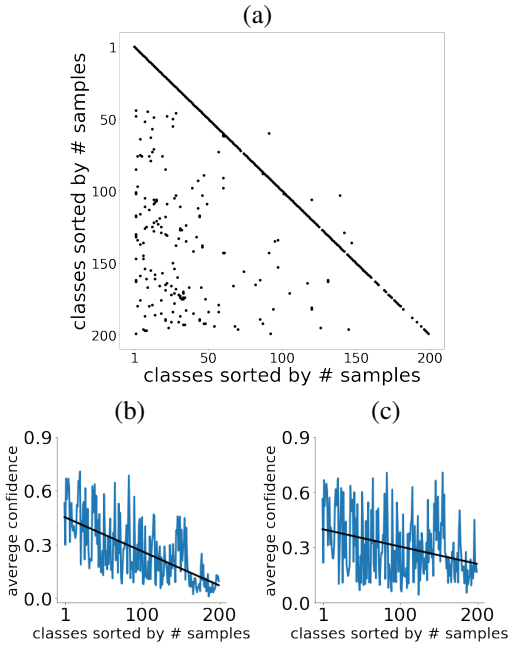


Fig. S 8: DRAGON learns to offset the familiarity bias. **(a)** A confusion matrix of a ResNet101 trained on CUB-LT as a function of the number of samples per class. The matrix shows markers for pairs of (gt, predicted) whose count is larger than 15. **(b)** Average-confidence per-class of the classifier. **(c)** Similar curve as (b) but for DRAGON. Black lines depict a linear regression line. DRAGON per-class confidence has smaller dependence on the number of samples in the train.

## C. Visual experts are better at the head, semantic experts excel at the tail

Here we provide supporting evidence to our observation from Section 4 of the main paper that semantic experts are better at the tail: "*Semantic descriptions of classes can be very useful for tail (low-shot) classes, because they allow models to recognize classes even with few or no training samples [26, 52, 4]*". Additionally, we demonstrate that the visual expert is better for the many-shot regime.

We focus on the Two-Level CUB distribution, and evaluate the accuracy for the many shot classes when restricting predictions to these classes (many-among-many), and separately the accuracy for the few-shot classes when predictions are restricted to these tail classes (few-among-few).

Figure S10(a) shows the accuracy over few-shot classes of both experts in few-among-few setting. The semantic expert outperforms the visual one, and this effect stronger with fewer samples. For example, with 1-shot learning, the semantic expert is almost 100% better than the Visual Expert. Additionally, when we measure the accuracy of the many-shot classes in the many-among-many setup (accuracy at

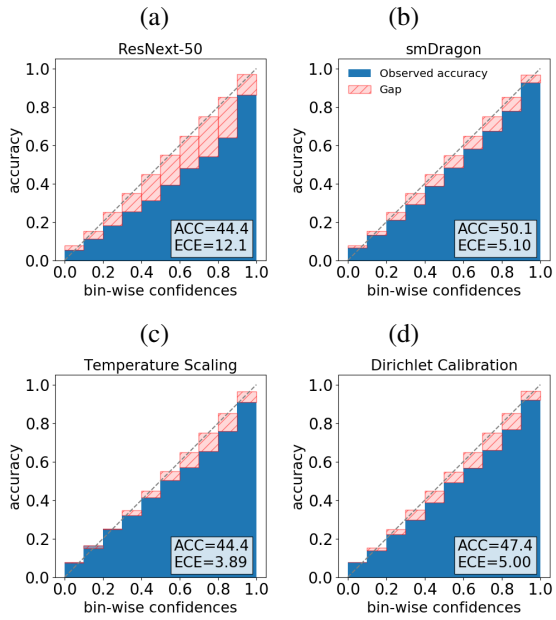


Fig. S 9: Reliability Diagrams on ImageNet-LT, for (a) raw ResNext-50, (b) smDRAGON, (c) Temperature-Scaling [17] and (d) Dirichlet-Calibration [25]. We report expected-calibration-error (ECE) and per-class accuracy (ACC)

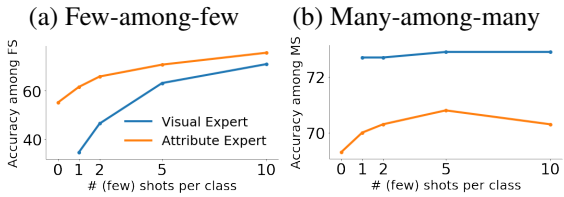


Fig. S 10: Accuracy as a function of number of samples at the tail, of the *Visual Expert* and the *Semantic Expert* used in our study. **(a)** Accuracy among few-shot classes; The Semantic expert outperforms the visual expert. **(b)** Accuracy among many-shot classes; The Visual expert outperforms, regardless of the number of samples at the tail.

the head), the visual expert is better than the semantic expert Figure S10(b).

## D. Training the fusion-module in small scale datasets

Our goal is to have the fusion-module learn to capture the correlations between the number of training samples and the output confidence (the familiarity bias), so it can adjust for it. Unfortunately, while the familiarity effect is substantial in the validation data and the test data, it may not present in the training data in small scale datasets. The reason is: Models tend to overfit and become overconfident over rare classes in the training set. This effect is illustrated in Figure

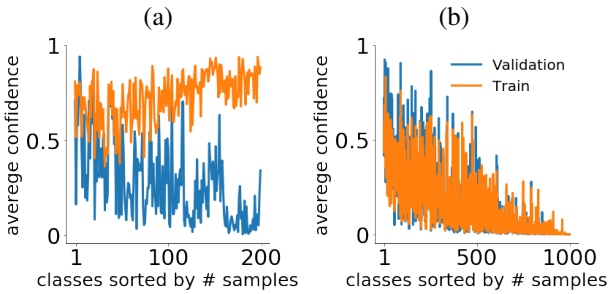


Fig. S 11: The familiarity bias effect: **(a)** On CUB-LT the effect is strong on validation samples (blue) but not on training samples (orange). **(b)** On ImageNet-LT the effect is prominent on both train and validation samples.

S11(a) (compare Train versus Validation curves) for CUB-LT. We observed the effect in also in SUN-LT and AWA-LT.

To address this mismatch, we hold-out 50% of the samples of the tail classes and 20% of the samples of the head classes of the training data and use it to simulate the response of experts to test samples. This set is used for training the fusion-module.

Note, that after training the fusion module, we re-train the experts on all the training data (including the hold-out set), in order to use all data available. (See Section E for more details).

In large-scale datasets, like ImageNet-LT, no hold-out set is needed and DRAGON is trained on the training set. There, the familiarity bias is also present on the training data (Figure S11(b)), as the models did not overfit the tail classes.

## E. Implementation details

**Training:** Considering the observation from Section D, regarding CUB-LT, SUN-LT and AWA-LT, we train the architecture in three steps: First, we train each expert on the training data excluding the hold-out set. Second, we freeze the expert weights and train the *fusion-module* on all the training set. Finally, we re-train the experts on all the training data in order to use all data available. For the hold-out set, we randomly draw half the samples of the tail classes and 20% of the samples of the head classes. For inference, we use the fusion-module trained at the second step with the experts trained at the third step.

**Platt-scaling:** We used Platt-scaling [34] to tune the combination coefficient $\lambda$ by adding constant bias $\beta$ and applying a sigmoid on top of its scores: $\lambda = \sigma\Big[f_0 - \beta\Big]$, where $\beta$ is a hyperparameter selected with cross validation.

**Fusion-module:** We trained the fusion-module using ADAM [24] optimizer. For large-scale datasets, like ImageNet-LT, Places-LT and iNaturalist, we used $L_2$ regularization, selected by hyperparameter optimization using grid search $\in \{10^{-5}, 10^{-4}, 10^{-3}\}$], to avoid overfitting.

| SORTING | $Acc_{PC}$ | $Acc_{LT}$ |
|---|---|---|
| NO-SORTING | 58.5 | 57.0 |
| **SORTING-BY-VISUAL-EXPERT** | **60.1** | **67.7** |
| SORTING-BY-SEMANTIC-EXPERT | 59.8 | 67.7 |

Table S 13: Ablation study, quantifying the contribution of sorting the fusion-module inputs (test set, CUB-LT).

| ARCHITECTURE | $Acc_{PC}$ | $Acc_{LT}$ |
|---|---|---|
| F.C. | 56.4 | 66.3 |
| F.C. & $1/n_y$ RE-SCALE | 56.4 | 56.2 |
| F.C. & NON-PARAMETRIC RE-SCALE | 58.2 | 64.3 |
| CONV. & NON-PARAMETRIC RE-SCALE | 58.4 | 67.1 |
| CONV. & SINGLE PARAMETRIC RE-SCALE | 59.3 | 64.3 |
| **DRAGON (OURS)** | **60.1** | **67.7** |

Table S 14: Ablation study, comparing different fusion and re-scaling approaches. The results show the contribution of the convolutional backbone and the re-scaling method for the two experts (test set, CUB-LT).

**Hyper-parameter tuning:** We determined the number of training epochs (early-stopping), selected architecture alternatives, and tuned hyperparameters using the validation set, using $Acc_{LT}$ for *Smooth-Tail* and *Vision-only*, and $Acc_{PC}$ for *Two-Level*.

For *DRAGON*: We optimized the following hyper-parameters: (1) Number of filters in the convolution layer $\in \{1, .., 4\}$. (2) Degree of polynomial in Eq.2 $\in \{2, 3, 4\}$. (3) Learning rate $\in \{10^{-5}, 10^{-4}, 10^{-3}\}$. (4) Bias term of Platts rescaling $\beta \in [-2, 2]$.

For *CADA-VAE [38]*: We applied a grid search for the latent embedding space $\in [12, 25, 50, 64, 100, 200, 250]$, variational-autoencoder learning rate $\in [0.0001, 0.00015, ..., 0.015]$ and classifier learning rate $\in [0.0001, 0.0005, ..., 0.1]$. We used a batch size of 64.

For *Focal Loss [27]*: We applied a grid-search for gamma $\in [1, 2, ..., 15]$ and alpha $\in [0.1, 0.2, 0.5, 0.75, 0.9, 1]$.

For *Range Loss [27]*: We applied a grid-search for alpha $\in [0.1, 0.2, 0.5, 0.75, 0.9, 1]$ and beta $\in [0.1, 0.2, 0.5, 0.75, 0.9, 1]$.

For *Anchor Loss [27]*: We applied a grid-search for gamma $\in [0.1, 0.5, 1, ..., 15]$ and slack $\in [0.001, 0.005, 0.01, ..., 0.5]$.

For *LDAM Loss [27]* we applied a grid-search for C $\in [0.1, 0.2, ..., 0.9]$.

## E.1. Computing $Acc_{LT}$:

$Acc_{LT}$ measures the accuracy over a test distribution that resembles the training distribution. However, the test and validation samples of CUB-LT, SUN-LT and AWA-LT have a different distribution because they were originated from an approximate uniform distribution. Thus, to compute $Acc_{LT}$ we measure the accuracy for each individual class, and then take a weighted sum according to the class frequencies in the training set. Specifically, for each class, we assign a weight $P_{train}(y)$ according to the train-set distribution such that $0 < P_{train}(y) < 1$ and $\sum_y p_{train}(y) = 1$. Then we compute the accuracy per class and report the weighted average across all classes: $Acc_{LT} = \sum_{y=1}^{k} p_{train}(y) acc(y)$. This is equivalent to transforming the test set to have the same distribution as the train set.

| TRAINING PROCESS | $Acc_{PC}$ |
|---|---|
| ALL-TRAIN | 56.6 |
| END-TO-END | 46.4 |
| **THREE-STAGE-TRAINING** | **60.1** |

Table S 15: Ablation study, quantifying the contribution the effect of three-stage training as proposed in Section E. (test-set, CUB)

## E.2. A clarification about the Smooth-Tail benchmark

In this section, we explain how the long-tail benchmark was aligned with the two-level benchmark, as was mentioned in the paragraph that describes the long-tailed datasets (Section 6 of the main paper).

To align the long-tail benchmark with the two-level benchmark, we first ordered the classes according to their number of samples in the *two-level* distribution. Then we calculated the number of samples for each class according to the required long-tail distribution, and accordingly drew samples to construct the training set.

## E.3. Training CADA on Smooth-Tail benchmark

In this section, we explain how we trained CADA-VAE [38] for the long-tail benchmark.

To evaluate CADA-VAE [38] on long-tail benchmarks we used the code published by the authors and followed the training protocol exactly as they used for the two-level distribution. Since the protocol relies on a hard distinction between head classes and tail classes, we had to choose where to partition the smooth long-tail distribution to head and tail. Our solution is simple. It is based on the fact that we aligned the order of classes in the long-tail distribution to be the same order as in the two-level split (Section E.2). The alignment allowed us to use the same partition to head and tail as used for the two-level benchmark.

| Model | $Acc_{ms}$ | $Acc_{fs}$ | $Acc_H$ |
|---|---|---|---|
| Most Common Class* | 0.7, 0.7, 0.7, 0.7 | 0, 0, 0, 0 | 0, 0, 0, 0 |
| LDAM [7]* | 71.5, 71.9, 71.6, 71.5 | 1.2, 5.9, 24.1, 41.2 | 2.4, 10.9, 36.0, 52.2 |
| REVISE [43] | - | - | 36.3, 41.1, 44.6, 50.9 |
| CA-VAE [38] | 58.2, 57.6, 60.0, 62.2 | 44.8, 51.6, 59.4, 62.3 | 50.6, 54.4, 59.6, 62.2 |
| DA-VAE [38] | 50.6, 56.0, 56.8, 56.8 | 47.9, 53.2, 61.0, 65.4 | 49.2, 54.6, 58.8, 60.8 |
| CADA-VAE [38] | 59.6, 60.9, 62.3, 63.1 | 51.4, 57.5, 63.6, 68.8 | 55.2, 59.2, 63.0, 64.9 |
| CE Loss* **(VE)** | 72.7, 72.9, 72.7, 72.0 | 0.6, 3.7, 19.1, 38.6 | 1.2, 6.9, 30.2, 50.2 |
| LAGO [4]* **(SE)** | 69.2, 69.0, 69.0, 68.1 | 13.8, 21.9, 38.1, 51.5 | 23.0, 33.2, 49.0, 58.6 |
| **DRAGON (ours)** | 58.0, 62.9, 63.3, 66.1 | 52.8, 55.9, 63.8, 69.6 | **55.3, 59.2, 63.5, 67.8** |

(a) Two-Level CUB

| Model | $Acc_{ms}$ | $Acc_{fs}$ | $Acc_H$ |
|---|---|---|---|
| Most Common Class* | 0.2, 0.2, 0.2, 0.2 | 0, 0, 0, 0 | 0, 0, 0, 0 |
| LDAM [7]* | 43.7, 44.0, 44.2, 44.3 | 2.2, 6.6, 19.0, 31.8 | 4.3, 11.5, 26.6, 37.0 |
| REVISE [43] | - | - | 27.4, 33.4, 37.4, 40.8 |
| CA-VAE [38] | 35.8, 37.5, 37.5, 39.0 | 40.0, 46.5, 53.8, 55.7 | 37.8, 41.4, 44.2, 45.8 |
| DA-VAE [38] | 34.8, 37.3, 38.6, 38.2 | 41.4, 45.1, 50.2, 54.8 | 37.8, 40.8, 43.6, 45.1 |
| CADA-VAE [38] | 37.6, 38.2, 39.4, 41.9 | 44.1, 49.0, 55.3, 55.1 | 40.6, 43.0, 46.0, 47.6 |
| CE Loss* **(VE)** | 46.3, 46.3, 46.2, 45.6 | 0.9, 4.9, 17.2, 33.0 | 1.8, 8.9, 25.1, 38.3 |
| LAGO [4]* **(SE)** | 30.6, 30.4, 30.7, 31.0 | 14.4, 18.7, 21.9, 25.2 | 19.6, 23.2, 25.6, 27.8 |
| **DRAGON (ours)** | 37.2, 39.2, 40.5, 41.6 | 45.5, 49.6, 55.1, 57.2 | **41.0, 43.8, 46.7, 48.2** |

(b) Two-Level SUN

| Model | $Acc_{ms}$ | $Acc_{fs}$ | $Acc_H$ |
|---|---|---|---|
| Most Common Class* | 2.5, 2.5, 2.5, 2.5 | 0, 0, 0, 0 | 0, 0, 0, 0 |
| LDAM [7]* | 90.7, 90.7, 90.5, 90.5 | 6.6, 14.4, 26.6, 41.6 | 12.4, 24.8, 41.1, 57.0 |
| REVISE [43] | - | - | 56.1, 60.3, 64.1, 67.8 |
| CA-VAE [38] | 73.4, 77.7, 81.0, 81.0 | 56.8, 66.0, 72.8, 77.1 | 64.0, 71.3, 76.6, 79.0 |
| DA-VAE [38] | 74.0, 74.6, 73.5, 73.9 | 63.0, 71.4, 77.7, 79.8 | 68.0, 73.0, 75.6, 76.8 |
| CADA-VAE [38] | 76.6, 79.4, 81.9, 82.6 | 63.8, 68.7, 74.8, 78.0 | **69.6, 73.7, 78.2,** 80.2 |
| CE Loss* **(VE)** | 90.7, 90.9, 89.7, 87.9 | 5.9, 11.2, 32.6, 57.9 | |
| LAGO [4]* **(SE)** | 82.6, 81.9, 81.7, 81.5 | 11.5, 20.6, 46.2, 59.4 | 20.2, 33.0, 59.0, 68.7 |
| **DRAGON (ours)** | 74.5, 76.7, 79.2, 81.7 | 61.1, 62.9, 74.3, 82.1 | 67.1, 69.1, 76.7, **81.9** |

(c) Two-Level AWA

Table S 16: Comparing DRAGON with SoTA GFSL models and baselines with increasing number of few-shot training samples on the CUB, SUN and AWA datasets. We report per-class $Acc_{ms}$, $Acc_{fs}$ and $Acc_H$. Each cell represents 1-shot,2-shot,5-shot and 10-shot accuracies

# F. Additional metrics

## F.1. $Acc_{fs}$ and $Acc_{ms}$ on Two-Level benchmarks

Table S16 provides the results of $Acc_{fs}$ and $Acc_{ms}$ (described in section 7) for the Two-Level benchmark. We show results for 1,2,5 and 10-shots. At the main paper we reported the results for the $Acc_H$ metrics, which is derived from $Acc_{fs}$ and $Acc_{ms}$ reported here.

## F.2. Ablation results on the test set

In the main paper (9) we described results for ablation study on the validation set. Here we report results for the same model variant on the test set.

Tables S13 and S14 show the results of the ablation study on the test set. It shows the same behavior as the ablation study on the validation set that was reported in the main paper. Table S15 compares three different training protocols: (1) *All-Train*: Training the DRAGON fusion-module naively

without a hold-out set. (2) *End-To-End*: Training all the architecture (both experts and fusion-module) end to end in an early fusion manner. (3) *Three-Stage-Training*: Training our models as explained in section E.