A Proofs

Proof of Proposition 1. Let G denote the distribution of the score $S = s(X, Y)$ for a randomly sampled example $(X, Y) \sim F$. For any $m \in \{1, ..., M\}$, let $G^{(m)}$ denote the distribution of the score conditioned on Y being in cluster m. Consider a randomly sampled test example $(X_{\text{test}}, Y_{\text{test}})$ with a label in cluster m, so its corresponding score $s_{\textsf{test}} = s(X_{\text{test}}, Y_{\text{test}})$ follows distribution $G^{(m)}$. Now consider $\{s_i\}_{i \in \mathcal{I}_2(m)}$, the scores for examples in the proper calibration dataset with labels in cluster m. Since each element of $\{s_i\}_{i \in \mathcal{I}_2(m)}$ follows distribution $G^{(m)}$ and is chosen to be included in the proper calibration dataset independently from the other elements and $s_{\textsf{test}}$, the element of $\{s_i\}_{i \in \mathcal{I}_2(m)}$ and $s_{\textsf{test}}$ are independent and identically distributed, which means that they are exchangeable. Thus, by the standard proof of coverage for conformal prediction (see, e.g., [1]), the desired result follows.

Proof of Proposition 2. This is a direct result of exchangeability and Proposition 1.

Proof of Proposition 3. This proof follows the structure of the proof of Proposition 4 in [2]. Let $S = s(X, Y)$ for $(X, Y) \sim F$ be a random variable representing the score of a randomly sampled example. Let $\mathcal{S}$ denote the set of values that S can take. Let $G^y(s) = \mathbb{P}(S \leq s \mid Y = y)$ denote the cdf of S when the label is y. Define $\mathcal{Y}^{(m)} = \{y \in \mathcal{Y} : \tilde{h}(y) = m\}$ as the set of classes in cluster m and let $G^{(m)}(s) = \mathbb{P}(S \leq s \mid Y \in \mathcal{Y}^{(m})$ denote the cdf of S when the label Y is in cluster m. Let $S^{(m)}$ be a random variable with cdf $G^{(m)}$ and for an arbitrary $y \in \mathcal{Y}^{(m)}$, let S^y be a random variable with cdf G^y.

Since we assume that the TV distance between the score distribution for every pair of classes in cluster m is bounded by ϵ, and $G^{(m)}$ is a mixture of these distributions, it follows that

$$\mathrm{TV}(S^y, S^{(m)}) \leq \epsilon.$$

By definition of TV distance, this is equivalent to

$$\sup_{A \in \mathcal{S}} \left| \mathbb{P}(S \in A \mid Y = y) - \mathbb{P}(S \in A \mid Y \in \mathcal{Y}^{(m)}) \right| \leq \epsilon,$$

which we can rewrite as

$$\sup_{f \in \mathcal{F}_{\mathbb{1}}} \left| \mathbb{E}[f(S) \mid Y = y] - \mathbb{E}[f(S) \in A \mid Y \in \mathcal{Y}^{(m)}] \right| \leq \epsilon,$$

where $\mathcal{F}_{\mathbb{1}} = f : \mathcal{S} \to [0, 1]$. Define $g(s) = \mathbb{1}\{s \geq \hat{q}(m)\}$. Since $g \in \mathcal{F}_{\mathbb{1}}$, we have

$$\mathbb{E}[\mathbb{1}\{S \geq \hat{q}(m)\} \mid Y = y] - \mathbb{E}[\mathbb{1}\{S \geq \hat{q}(m)\} \mid Y \in \mathcal{Y}^{(m)}] \leq \epsilon,$$

which can be expressed as

$$\mathbb{P}(S \geq \hat{q}(m) \mid Y = y) - \mathbb{P}(S \geq \hat{q}(m) \mid Y \in \mathcal{Y}^{(m)}) \leq \epsilon.$$

Since the CLUSTERED procedure will exclude the true label Y from the prediction set C exactly when $S \geq \hat{q}(m)$, the probabilities can be re-expressed in terms of mis-coverage:

$$\mathbb{P}(Y \notin C(X) \mid Y = y) - \mathbb{P}(Y \notin C(X) \mid Y \in \mathcal{Y}^{(m)}) \leq \epsilon.$$

By Proposition 1, we know $\mathbb{P}(Y \notin C(X) \mid Y \in \mathcal{Y}^{(m)}) \leq \alpha$, so

$$\mathbb{P}(Y \notin C(X) \mid Y = y) \leq \alpha + \epsilon.$$

Taking the complement yields

$$\mathbb{P}(Y \in C(X) \mid Y = y) \geq 1 - \alpha - \epsilon.$$

This is true for all $y \in \mathcal{Y}^{(m)}$ and for every cluster $m = 1, ..., M$.

B Experiment details

B.1 Score functions

We perform experiments using three score functions:

- softmax: The conformal score of an input x and a label y is one minus the softmax score:

$$s_{\mathsf{softmax}}(x, y) = 1 - f_y(x)$$

  where $f_y(x)$ is entry y of the softmax vector of input x.

- APS: *Adaptive Prediction Sets* are designed to achieve approximate X-conditional coverage [17]. The conformal score of input x and label y is computed as follows: For $y = 1, ..., |\mathcal{Y}|$, let $\widehat{p}_y(x)$ be an estimate of $\mathbb{P}(Y = y \mid X = x)$. We use the softmax score as our $\widehat{p}$, so $\widehat{p}_y(x) = f_y(x)$. Let $\widehat{p}_{(i)}(x)$ be the i-th largest $\widehat{p}_{(y)}(x)$. Define j to be the index in the sorted order that corresponds to class y, i.e., $\widehat{p}_{(j)}(x) = \widehat{p}_y(x)$. Then,

$$s_{\mathsf{APS}}(x, y) = \sum_{i=1}^{j-1} \widehat{p}_{(i)}(x) + \mathsf{Unif}([0, \widehat{p}_{(j)}(x)])$$

- RAPS: One problem with APS is that the resulting prediction sets are often very large. *Regularized Adaptive Prediction Sets* [3] modifies APS by introducing an additive regularization term designed to reduce the prediction set sizes:

$$s_{\mathsf{RAPS}}(x, y) = s_{\mathsf{APS}}(x, y) + \max(0, \lambda(o_x(y) - k_{reg}))$$

  where $o_x(y)$ denotes the ranking of y among the values of $\widehat{p}_k(x)$ for all classes k (e.g., $o_x(y) = 1$ if $\widehat{p}_y(x)$ is larger than all other $\widehat{p}_k(x)$), and λ and k_{reg} are user-chosen parameters. In our experiments, we use $\lambda = 0.01$ and $k_{reg} = 5$, which Angelopoulos et al. found to work well for ImageNet [3].

B.2 Model training

An important consideration when training our models is that we need to reserve sufficient data for evaluating the class-conditional coverage of the conformal prediction methods. In practice, this means we should aim to exclude at least 250 examples per class from the model training dataset so that we can use those untouched examples for validation (i.e., applying the conformal methods and computing coverage and set size metrics).

For all datasets except ImageNet, we use a ResNet-50 as our predictive model. We initialize to the `IMAGENET1K_V2` pre-trained weights, then fine-tune all parameters by training on the dataset-specific data. We apply the model to the validation data to obtain softmax scores.

ImageNet. Setting up ImageNet for our setting is a bit tricky because we want sufficient data for performing validation, but we also need this data to be separate from the model training data. Unfortunately, the ImageNet validation set only contains 50 examples per class, which is not enough for validation in our setting. Fortunately, we have access to more labeled data from the ImageNet training set, which has roughly 1000 examples per class). However, if we want to use this data for validation, we cannot use our ResNet-50 initialized to the `IMAGENET1K_V2` pretrained weights, as these weights are obtained by training on the ImageNet training set and would violate the assumption of independence of the validation and model training datasets. To approximately satisfy this independence assumption, we instead use SimCLR-v2 [5], which is trained on the ImageNet training set *without labels*, to extract feature vectors of length 6144 for all images in the ImageNet training set. We then use 10% of these feature vectors for training a linear head (i.e., a single fully connected neural network layer). After training for 10 epochs, the model achieves a validation accuracy of 78%. We then apply the linear head to the remaining 90% of the feature vectors to obtain softmax scores.

CIFAR-100. In total, there are 600 images per class (500 from the training set and 100 from the validation set). We combine the data and randomly sample 50% for model training, leaving the remaining data for testing our procedure. After training for 30 epochs, the validation accuracy is 60%.

Places365. This dataset contains more than 10 million images of 365 classes. Each class has 5000 to 30000 examples. We randomly sample 90% of the data for model training and use the remaining data for testing our procedure. After training for one epoch, the validation accuracy is 52%.

iNaturalist. This dataset has class labels of varying specificity. At the `species` level, there are 6414 classes with 300 examples each (290 training examples and 10 validation examples) and a total of 10000 classes with at least 150 examples. We operate at the `family` level, which groups the species into 1103 classes. We randomly sample 50% of the data for model training and use the remaining for testing our procedure. After training for one epoch, the validation accuracy is 69%. However, due to class imbalance and the randomness of the model training set construction, some classes have insufficient validation samples. We filter out classes with fewer than 250 validation examples, which leaves us with 633 classes. The entries of the softmax vectors that correspond to rare classes are removed and the vector is renormalized to sum to one.

B.3 Choosing clustering parameters

In order to perform CLUSTERED, there are two parameters that must be chosen: γ, the probability that a calibration example will be assigned to the clustering dataset, and M, the number of clusters that will be requested when performing k-means.

As mentioned in Section 3.1, we make use of two intuitive heuristics to choose these parameters. We restate these heuristics in more detail here.

- First, to distinguish between more clusters (or distributions), we need more samples from each distribution. As a rough guess, to distinguish between two distributions, we want at least four samples per distribution; to distinguish between five distributions, we want at least ten samples per distribution. In other words, we want the number of clustering examples per class to be at least twice as large as the number of clusters. This heuristic can be expressed as
$$\gamma\tilde{n} \geq 2M, \tag{3}$$
where $\gamma\tilde{n}$ is the expected number of clustering examples for the rarest class not assigned to the null cluster.
- Second, we want enough data for computing the conformal quantiles for each cluster. We translate this into asking for at least 150 examples per cluster on average. This heuristic can be expressed as
$$(1-\gamma)\tilde{n}\frac{K}{M} \geq 150, \tag{4}$$
where $\frac{K}{M}$ is the average number of classes per cluster and $(1-\gamma)\tilde{n}$ is the expected number of proper calibration examples for the rarest class not assigned to the null cluster.

Changing the inequalities of (3) and (4) into equalities and solving for γ and M yields
$$M = \frac{\gamma\tilde{n}}{2} \qquad \text{and} \qquad \gamma = \frac{K}{K+75}.$$

Varying the clustering parameters. Although our method for choosing parameter values is arguably ad-hoc, we find that it does not really matter what parameter values are used, as long as they fall into a reasonable range. As the heatmaps in Figure 3 illustrate, the performance of CLUSTERED is not very sensitive to γ and M. When $n_{\text{avg}} = 10$, the heuristic chooses $\gamma = 0.89$ and $M = 4$. When $n_{\text{avg}} = 50$, the heuristic chooses $\gamma \in [0.88, 0.92]$ and $M \in [7, 12]$ (since the calibration dataset is randomly sampled, and γ and M are chosen based on the calibration dataset, there can be some randomness in the chosen values). However, there are large areas surrounding these chosen values that would yield similar performance. We observe that the heuristics do not always choose the parameter values that yield the lowest CovGap. The heatmaps show that the optimal parameter values are dependent not only on dataset characteristics, but also on the score function. Future work could be done to extract further performance improvements by determining a better method for choosing γ and M.

B.4 Measuring dataset class balance in Table 1

The class balance metric in Table 1 is defined as the number of examples in the rarest 5% of classes divided by the expected number of examples if the class distribution were perfectly uniform. This metric is bounded between 0 and 1, with lower values denoting more class imbalance. The metric is computed on the validation datasets, which are sampled uniformly at random from the publicly available versions of each dataset.

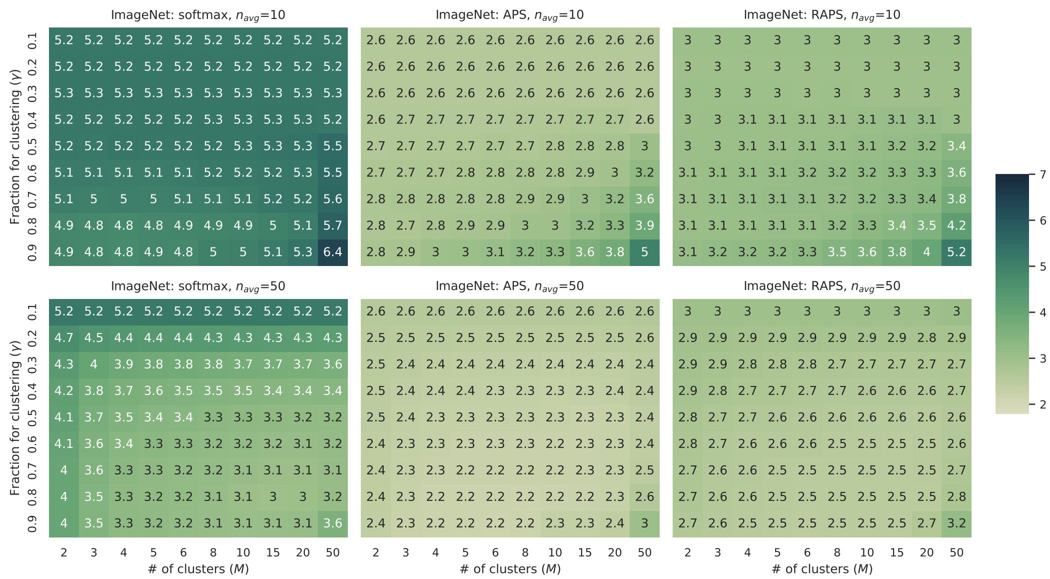

Figure 3: The average class coverage gap on ImageNet for $n_{\text{avg}} = 10, 50$ using softmax, APS, and RAPS as we vary the clustering parameters. Each entry is computed across 10 random splits of the data into calibration and validation sets.

C Additional experimental results

We present additional experimental results in this section. As in the main text, shaded regions in plots denote ± 1.96 times the standard errors.

C.1 RAPS CovGap results

Figure 4 shows the CovGap on all datasets when we use RAPS as our score function.

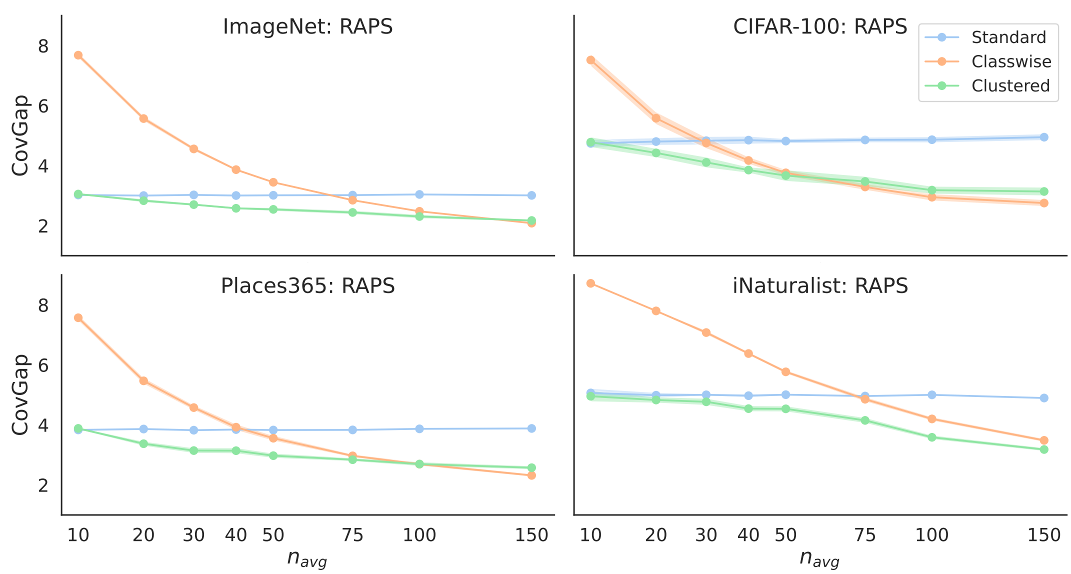

Figure 4: Average class coverage gap for ImageNet, CIFAR-100, Places365, and iNaturalist using RAPS scores, as we vary the average number of calibration examples per class.

C.2 Additional metrics

Average set size. To supplement Table 2 from the main text, which reports AvgSize for four values of n_{avg}, Figure 5 plots AvgSize for all values of n_{avg} that we use in our experimental setup. Note that RAPS sharply reduces AvgSize relative to APS on ImageNet and also induces a slight reduction for the other three datasets. This assymetric reduction is likely in large part due to the fact that the RAPS hyperparameters, which control the strength of the set-size regularization, were tuned on ImageNet. The set sizes of RAPS on other datasets could likely be improved by tuning the hyperparameters for each dataset.

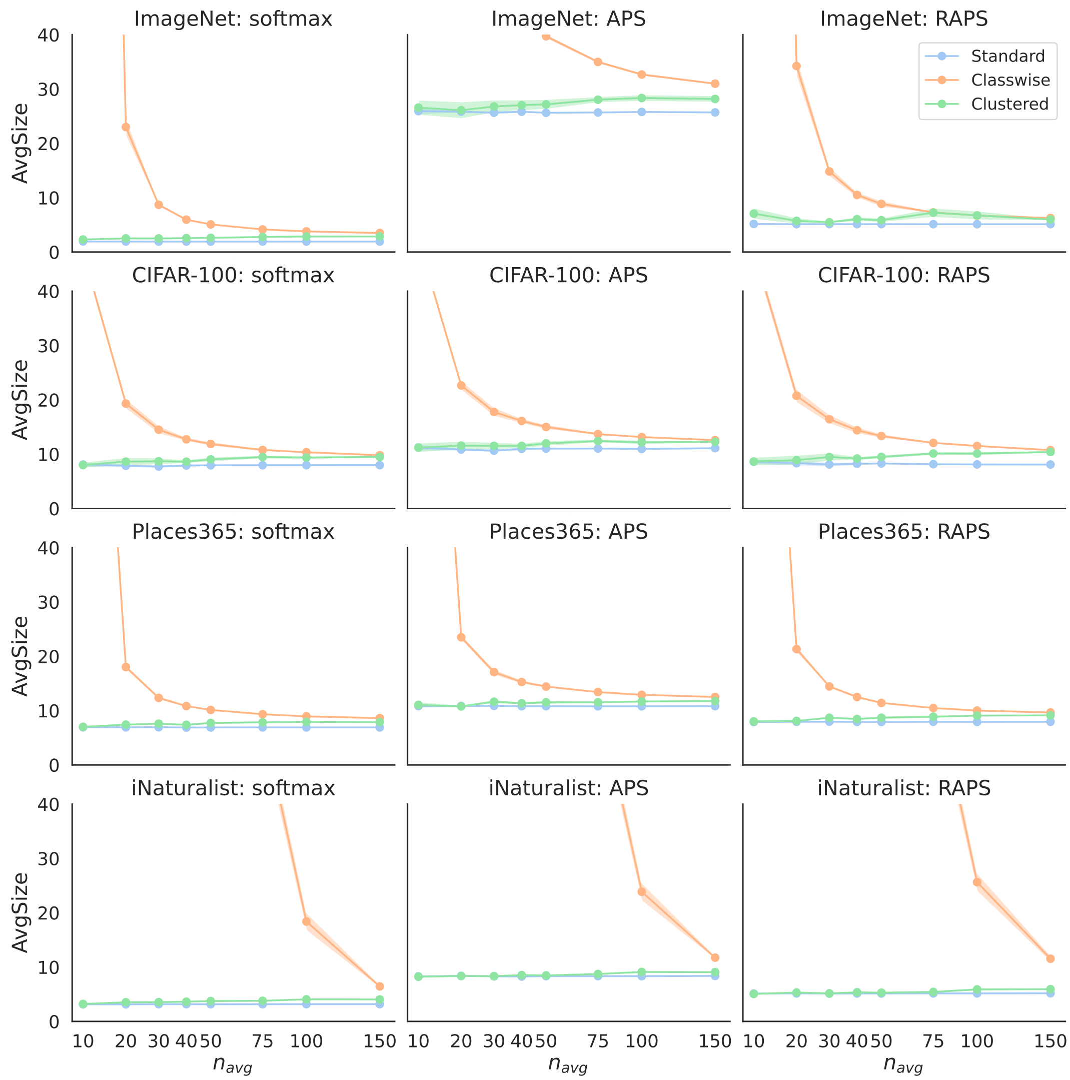

Figure 5: Average set size for ImageNet, CIFAR-100, Places365, and iNaturalist using softmax, APS, and RAPS scores, as we vary the average number of calibration examples per class.

Fraction under-covered. In many practical settings, we want to limit the number of classes that are severely under-covered, which we define as having a class-conditional coverage that is more than 10% below the desired coverage level. We define FracUnderCov to be the fraction of classes that are serverely under-covered:

$$\text{FracUnderCov} = \frac{1}{|\mathcal{Y}|}\sum_{y=1}^{|\mathcal{Y}|} \mathbb{1}\left\{c_y \leq 1 - \alpha - 0.1\right\},$$

recalling that c_y is the class-conditional coverage for class y.

Figure 6 plots FracUnderCov for all experimental settings. Comparing to the CovGap plots in Figure 2 and Figure 4, we see that the trends in FracUnderCov generally mirror the trends in CovGap. However, FracUnderCov is a much noisier metric, as evidenced by the large error bars. Another flaw of FracUnderCov as a metric is it is unable to penalize uninformatively large set sizes. This is best seen in the performance of CLASSWISE on iNaturalist: for every score function, CLASSWISE has very low FracUnderCov, but this is achieved by producing extremely large prediction sets, as shown in the bottom row of Figure 5. On the other hand, CovGap is able to impose a slight penalty on this kind of behavior since unnecessarily large set sizes often lead to over-coverage, and CovGap penalizes over-coverage.

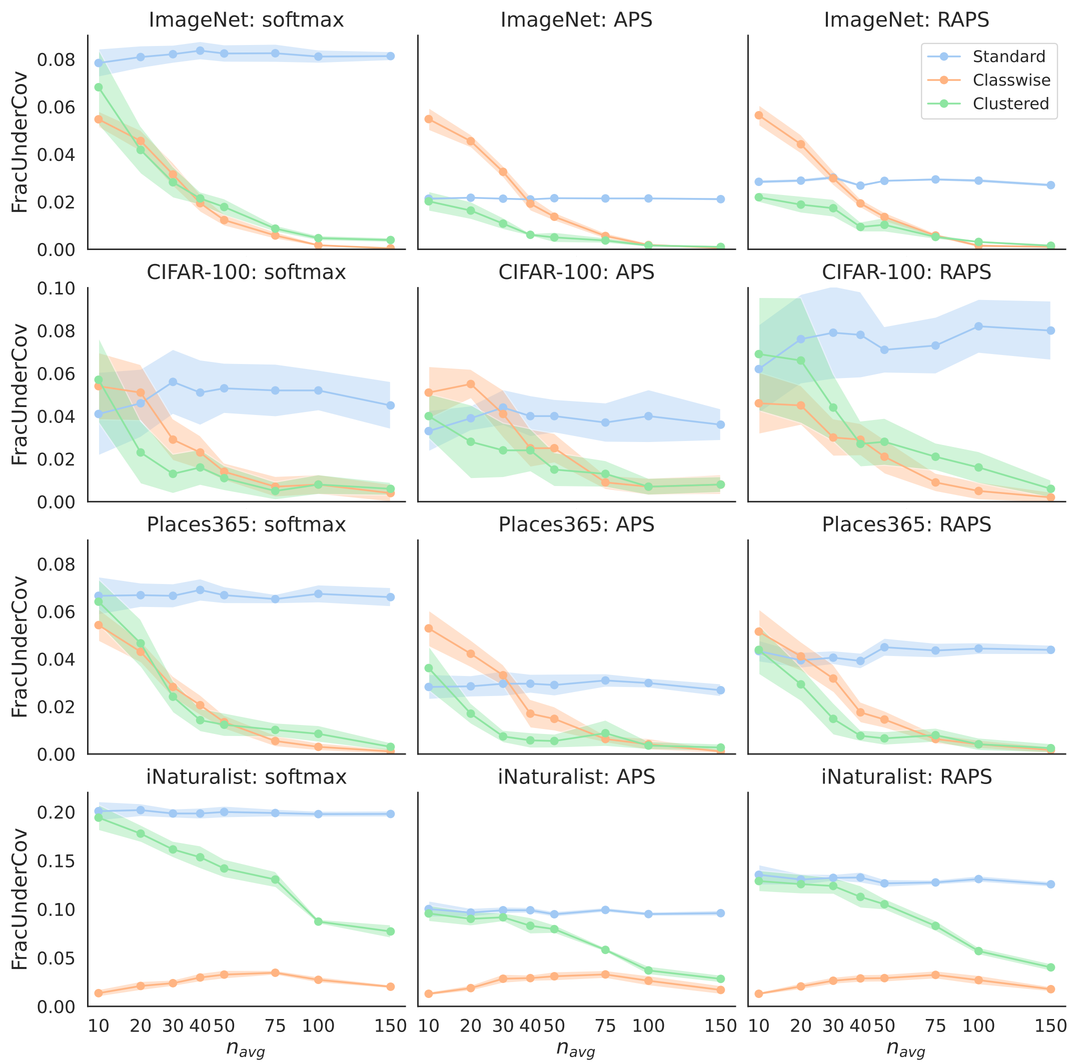

Figure 6: Fraction of very under-covered classes for ImageNet, CIFAR-100, Places365, and iNaturalist using softmax, APS, and RAPS scores, as we vary the average number of calibration examples per class.