Kandinsky Conformal Prediction: Efficient Calibration of Image Segmentation Algorithms

Supplementary Material

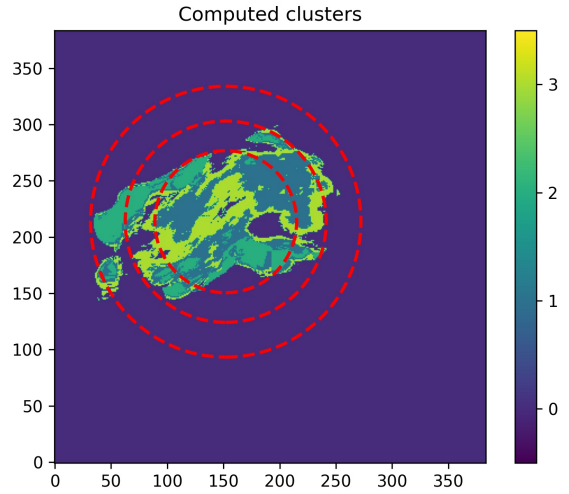

Figure 10. The computed k-means and GenAnn annuli clusters for the Medical Decathlon dataset with a small calibration set.

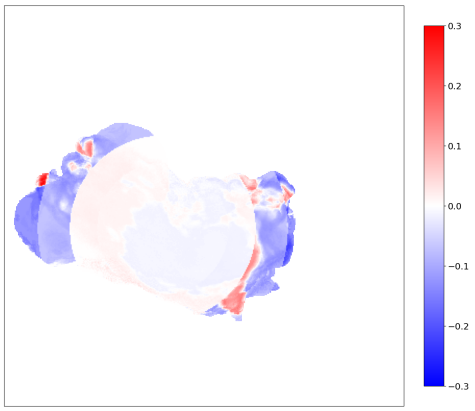

Figure 11. Subtraction image of coverage errors per pixel for imagewise vs. GenAnn calibration on the Decathlon-S dataset. Red indicates lower coverage error for the former, and blue indicates lower coverage error for the latter. Especially near the boundaries, the coverage error decreases for the GenAnn calibration.

6. Additional Cluster Plots

In Fig. 10 we provide an additional example of k-means clusters and GenAnn clusters, for the Decathlon-S experiment.

7. Details Genetic Algorithm

Let us start by providing a short insight into the workings of this approach. A candidate solution in a generation t, $x_i^{(t)}$, will consist of a finite set of real numbers, the set of (x, y) coordinates for a center, along with a set of radii for annuli clusters. The fitness function we utilize is the evaluation of the "internal" distances in each cluster. Each pixel p in a cluster of pixels will have an associated non-conformity curve n_p, which arises during the calibration procedure. The internal distance I_C for a cluster C of pixels is then defined as

$$I_C = \sum_{p,q \in C} d(n_p, n_q) \,, \tag{9}$$

where d is some metric between the two non-conformity curves. We measure the distance between non-conformity scores by comparing the values of both curves at a chosen set of quantiles. Let us denote $D_i^{(t)}$ as the set of clusters of pixels parameterized by a candidate solution, then the fitness function becomes

$$F\left(x_i^{(t)}\right) := \sum_{C \in D_i^{(t)}} I_C \,. \tag{10}$$

We detail the further specifics of the mutation, crossover, and replacement in the supplementary material, Sec. 7. The main advantage of this method is that provided you can find a parametrization of the clusters you wish to find, it can be utilized. We provide an overview of the algorithm process in Algorithm 1.

Algorithm 1 Differential Evolution Optimization

1: Initialize population with random candidate solutions $P^{(0)} = \{x_1^{(0)}, x_2^{(0)}, ..., x_N^{(0)}\}$
2: Evaluate the fitness $f_i^{(0)} = F(x_i^{(0)})$ for each individual
3: **for** each generation $t = 1, 2, ..., T$ **do**
4: Apply mutation $P_{\text{mut}}^{(t)} = M(P^{(t-1)})$
5: Create offspring via crossover $P_{\text{cross}}^{(t)} = C(P_{\text{mut}}^{(t)})$
6: Replace population $P^{(t+1)} = R(P^{(t-1)}, P_{\text{cross}}^{(t)})$
7: Re-evaluate the fitness for the new population
8: Check for termination condition
9: **end for**
10: Output the best solution $x^* = \arg\min_{x_i^{(T^*)}} f_i^{(T^*)}$ with T^* the last generation

For the genetic algorithm approach of finding annuli in the MS-COCO case, we have used the range $[-20, 20]$ as

bounds for the center x and y coordinates, and $[0, 150]$ for the two innermost radii and $[0, 300]$ for the outermost (as it is allowed to lie outside of the image if necessary). In the case of the medical decathlon dataset, the radii, the bounds of the center coordinates were $[-60, 60]$ and the inner radii up to 200 and outermost 400. These choices were made based on the image sizes. We did not find that it was necessary to add any 'repulsion' term in the objective function between different radii, as it is generally a more favorable solution to have separate annuli for lower objective function values.
For the mutation and crossover, we start by selecting three random individuals from the population, a, b, and C. The mutation will create a new, mutated individual by

$$x_{\text{mut}} = a + C_M(b - c)\,, \tag{11}$$

where C_M denotes the mutation factor, chosen to be 0.8 in our experiments. The operations in the mutation are pointwise on the individuals. Subsequently, a crossover will be performed between the candidate solution and the mutated one. Suppose each candidate solution is of length n, i.e., it has n parameters specifying clusters. We first define a crossover probability $p_C = 0.7$ and then uniformly sample n random numbers that lie in $[0, 1)$. Whenever the random number exceeds p_C, we accept the mutated parameter. After the mutation and crossover, we end up with a new candidate solution $\tilde{x}$, which replaces the original candidate solution x only if the fitness is lower.

We note that the computation of the 'internal' distance, i.e., the distance between the quantiles of the non-conformity curves, is very amenable to large matrix operations. Therefore, we have implemented it using PyTorch, allowing for a batched, GPU-compatible computation. All our computations were performed on an Nvidia A6000 GPU but can be on a CPU as well.

8. Details Fourier Concentric Clustering

In this section we explain how to construct a concentric decomposition of $\mathcal{R}_n := [0, n_1 - 1] \times [0, n_2 - 1]$ on which the variances of a user-prescribed quantity $\mathcal{J} : \mathcal{R}_n \to \mathbb{R}^p$ are minimized. Here $n := (n_1, n_2) \in \mathbb{N}^2$ corresponds to the measurements of the domain of the underlying image of interest. Specifically, we explain how to construct a sequence of simply connected nested subsets $V_0 \subset \ldots \subset V_{m-1} \subset \mathcal{R}_n$ such that $\sum_{l=0}^{m} \mathbb{E}_{A_l}\left(\|\mathcal{J} - \boldsymbol{\mu}_l\|^2\right)$ is minimal, where

$$A_l := \begin{cases} V_0, & l = 0, \\ V_l \setminus V_{l-1}, & 1 \le l \le m-1, \\ \mathcal{R}_n \setminus V_{m-1}, & l = m. \end{cases}$$

Here $m \in \mathbb{N}$ is the number of sets in the filtration $(V_l)_{l=0}^{m-1}$, $\|\cdot\|$ denotes the standard Euclidean norm on $\mathbb{R}^p$ and $\boldsymbol{\mu}_l = \mathbb{E}_{A_l}(\mathcal{J}) \in \mathbb{R}^p$ is the mean of $\mathcal{J}$ on A_l. We require that each set V_l has a smooth boundary and refer to A_l as a domain.

8.1. Explicit representation of the filtration

We start by constructing an explicit representation of each A_l amenable to numerical computations. First, define a new (Cartesian) coordinate system in which the midpoint $\boldsymbol{p} = (p_1, p_2)$ of $\mathcal{R}_n$ corresponds to the origin. Next, parameterize A_0 by assuming its boundary ∂A_0 is a polar curve, i.e., in polar coordinates we have

$$A_0 = \{(r, \theta) : 0 \le r \le r_0(\theta),\ \theta \in [0, 2\pi]\}\,,$$

where $r_0 : [0, 2\pi] \to (0, \infty)$ is a continuously differentiable 2π-periodic map. Similarly, we assume $r_{l-1}, r_l : [0, 2\pi] \to (0, \infty)$ are polar curves parameterizing the boundaries of V_{l-1} and V_l, respectively, and hence

$$A_l = \{(r, \theta)\ : r_{l-1}(\theta) \le r \le r_l(\theta),\ \theta \in [0, 2\pi]\}\,, \quad 1 \le l \le m-1.$$

For the final domain A_m, the "outer" boundary $\partial\mathcal{R}_n$ is fixed and no explicit parameterization is required:

$$A_m = \{(r, \theta) : r \ge r_{m-1}(\theta),\ 0 \le p_1 + r\cos\theta \le n_1 - 1,$$
$$0 \le p_2 + r\sin\theta \le n_2 - 1,\ \theta \in [0, 2\pi]\}\,.$$

The domains $(A_l)_{l=0}^{m}$ are fully determined by the polar curves $(r_l)_{l=0}^{m-1}$. Since these mappings are smooth and 2π-periodic, they admit uniformly convergent Fourier series. For this reason, we have chosen to represent the polar curves using finite Fourier expansions:

$$r_l(\theta) := \sum_{|k| \le K-1} a_{lk} e^{ik\theta}, \quad 0 \le l \le m-1,$$

where $(a_{lk})_{k=1-K}^{K-1} \in \mathbb{C}^{2K-1}$ are the Fourier coefficients of r_l. Here $K \in \mathbb{N}$ is a user-prescribed hyperparameter which we shall refer to as the order of the Fourier series. We only store the Fourier coefficients $a_l := (a_{lk})_{k=0}^{K-1} \in \mathbb{C}^K$, since r_l is real-valued.

Remark 8.1. In practice, we actually define

$$r_l(\theta) := \left(\sum_{|k| \le K-1} a_{lk} e^{ik\theta}\right)^2, \quad 0 \le l \le m-1,$$

to ensure that the resulting radii are positive. For the sake of presentation, however, we shall ignore this minor detail in the following sections.

8.2. Numerical evaluation of integrals

To compute the variances $\mathbb{E}_{A_l}\left(\|\mathcal{J} - \boldsymbol{\mu}_l\|^2\right)$, as (differentiable) functions of $\boldsymbol{a} := (a_0, \ldots, a_{m-1}) \in \mathbb{C}^{mK}$, we need to numerically evaluate integrals over the domains A_l. We

use a combination of Legendre and Fourier quadrature to accomplish this. In this section we explain in detail how to numerically integrate an arbitrary continuously differentiable map $f : \mathcal{R}_n \to \mathbb{R}$ over each domain.

For the innermost domain A_0, we have

$$\int_{A_0} f(x)\,\mathrm{d}x = \int_0^{2\pi}\int_0^{r_0(\theta)} f\left(p_1 + r\cos\theta, p_2 + r\sin\theta\right) r\,\mathrm{d}r\,\mathrm{d}\theta\,.$$

Notice that the inner integral is a 2π-periodic function of θ and continuously differentiable. Therefore, it admits a unique Fourier expansion. In particular, if $(c_{0k})_{k\in\mathbb{Z}}$ are the corresponding Fourier coefficients, then

$$\int_{A_0} f(x)\,\mathrm{d}x = 2\pi c_{00}. \tag{12}$$

Consequently, to approximate the desired integral, it suffices to approximate the zero$^{\text{th}}$ Fourier coefficient of

$$\theta \overset{I_0}{\mapsto} \int_0^{r_0(\theta)} f\left(p_1 + r\cos\theta, p_2 + r\sin\theta\right) r\,\mathrm{d}r\,. \tag{13}$$

For this purpose, we first sample (13) on an equidistributed grid of $[0, 2\pi]$ of size $K_I \in \mathbb{N}$. To approximate the integrals in (13) at fixed angles, we use Gaussian quadrature of order $K_G \in \mathbb{N}$. Next, we use the (inverse) Fast Fourier Transform (FFT) to approximate $(c_{0l})_{l=0}^{K_I-1}$, and in turn the desired integral using (12).

The integrals for $1 \le l \le m-1$ are approximated in a similar fashion. More precisely, for $1 \le l \le m-1$, we have

$$\int_{A_l} f(x)\,\mathrm{d}x = \int_0^{2\pi}\int_{r_{l-1}(\theta)}^{r_l(\theta)} f\left(p_1 + r\cos\theta, p_2 + r\sin\theta\right) r\,\mathrm{d}r\,\mathrm{d}\theta\,.$$

As before, we first note that

$$\theta \overset{I_l}{\mapsto} \int_0^{2\pi}\int_{r_{l-1}(\theta)}^{r_l(\theta)} f\left(p_1 + r\cos\theta, p_2 + r\sin\theta\right) r\,\mathrm{d}r\,\mathrm{d}\theta$$

is a continuously differentiable 2π-periodic function. Hence it admits a unique Fourier expansion. Therefore, if $(c_{lk})_{k\in\mathbb{Z}}$ are the associated Fourier coefficients, then

$$\int_{A_l} f(x)\,\mathrm{d}x = 2\pi c_{l0}, \quad 1 \le l \le m-1,$$

as before, and we approximate the right-hand side in exactly the same way.

Finally, we approximate the integral over A_m using a combination of two-dimensional quadrature and the computations above. More precisely, first observe that

$$\int_{A_m} f(x)\,\mathrm{d}x = \int_{\mathcal{R}_n} f(x)\,\mathrm{d}x - \int_{V_{m-1}} f(x)\,\mathrm{d}x.$$

We approximate the first integral using 2-dimensional Gaussian quadrature with order $(K_G, K_G) \in \mathbb{N}^2$. The second integral is approximated using the strategy explained in the previous paragraph.

Remark 8.2. One could use different orders K, K_I, K_G (and (K_G, K_G) for the final domain) on each domain. For simplicity, to reduce the number of hyperparameters, we have kept them the same across all domains.

8.3. Numerical evaluation of variances

Next, we explain how to compute $\mathbb{E}_{A_l}\left(\|\mathcal{J} - \boldsymbol{\mu}_l\|^2\right)$ given Fourier coefficients $\boldsymbol{a}$. To this end, first observe that the area of A_l is given by

$$\lambda(A_l) = \begin{cases} \frac{1}{2}\int_0^{2\pi} r_0(\theta)^2\,d\theta, & l = 0, \\ \frac{1}{2}\int_0^{2\pi} \left(r_l(\theta)^2 - r_{l-1}(\theta)^2\right)\,d\theta, & 1 \le l \le m-1, \\ (n_2-1)(n_1-1) - \frac{1}{2}\int_0^{2\pi} r_{m-1}(\theta)^2\,d\theta, & l = m, \end{cases}$$

where λ is the Lebesgue measure on $\mathcal{R}_n$. Using the Fourier representations of the polar curves r_l, we see that

$$\lambda(A_l) = \begin{cases} \pi\left(a_0 * a_0\right)_0, & l = 0, \\ \pi\left(\left(a_l * a_l\right)_0 - \left(a_{l-1} * a_{l-1}\right)_0\right), & 1 \le l \le m-1, \\ (n_2-1)(n_1-1) - \pi\left(a_{m-1} * a_{m-1}\right)_0, & l = m, \end{cases}$$

where $*$ denotes the two-sided discrete convolution. We use these computations to turn each A_l into a probability space $(A_l, \mathcal{B}(A_l), \mathbb{P}_l)$, where $\mathcal{B}(A_l)$ is the Borel-sigma algebra on A_l, and $\mathbb{P}_l = \frac{\lambda|_{A_l}}{\lambda(A_l)}$.

We may now readily use the approximation techniques described in the previous section to approximate the variance of $\mathcal{J}$ on A_l:

$$\mathbb{E}_{A_l}\left(\|\mathcal{J} - \boldsymbol{\mu}_l\|^2\right) = \frac{1}{\lambda(A_l)}\int_{A_l} \|\mathcal{J}(x) - \boldsymbol{\mu}_l\|^2\,\mathrm{d}x,$$
$$\boldsymbol{\mu}_l = \frac{1}{\lambda(A_l)}\int_{A_l} \mathcal{J}(x)\,\mathrm{d}x$$

for $0 \le l \le m$.

8.4. Setting up a minimization problem

Finally, we set up a minimization problem to find a concentric decomposition of $\mathcal{R}_n$. We choose the initial filtration

$(V_l)_{l=0}^{m-1}$ as a "noisy" set of concentric nested circles. This corresponds to the following initialization of the Fourier coefficients of $(r_l)_{l=0}^{m-1}$:

$$a_l = (R_l, 0, \dots, 0) + \varepsilon, \quad 0 \le l \le m-1,$$

where $\varepsilon \sim N(\mathbf{0}_K, \sigma \boldsymbol{I}_{K\times K})$, $\sigma > 0$, $R_l = \frac{l}{m} R_{\max}$, and $R_{\max} > 0$ is a hyperparameter chosen so that the circle with radius $R_{\max}$ and midpoint $\boldsymbol{p}$ is contained in $\mathcal{R}_n$. With this initialization, we use the BFGS-algorithm to solve

$$\min_{\boldsymbol{a}\in\mathbb{C}^{mK}} \left(\sum_{l=0}^{m} \mathbb{E}_{A_l} \left(\|\mathcal{J} - \boldsymbol{\mu}_l\|^2 \right) + w \sum_{l=1}^{m} \Lambda(\boldsymbol{a})_l \right),$$

where $w > 0$ is a fixed weight, and

$$\Lambda(\boldsymbol{a})_l := \begin{cases} \displaystyle\int_0^{2\pi} \frac{1}{r_l(\theta) - r_{l-1}(\theta)} \, \mathrm{d}\theta, & 1 \le l \le m-1, \\ \displaystyle\int_0^{2\pi} \frac{1}{R_{\max} - r_{m-1}(\theta)} \, \mathrm{d}\theta, & l = m, \end{cases}$$

is a penalty term that enforces the sets V_l to remain nested. That is, it enforces that the boundaries do not touch each other. The integrals are approximated, as before, by computing the zero$^{\text{th}}$ Fourier coefficients of the integrands, which in turn are obtained by sampling them on an equispaced grid of $[0, 2\pi]$ of size $K_\Lambda \in \mathbb{N}$ and using the (inverse) FFT.

Remark 8.3. The penalty term for $l = m$ could be improved substantially by explicitly parameterizing $\partial\mathcal{R}_n$. In particular, the current term yields a suboptimal filtration, since it confines the final polar curve r_{m-1} to a circle incscribed in $\mathcal{R}_n$.

9. Calibration set sizes

It is *a priori* not clear what the 'crossover' point is from which the Kandinsky method will outperform proper pixel-wise calibration. In Figure 12, we present the the coverage error for the pixelwise versus Kandinsky methods as a function of calibration dataset size. We mention that there is no calibration set 'too large'; the problem is that we would like as much training data as possible and minimize the amount of data set aside for calibration. From the Figure, we see that the Kandinsky method is approximately constant in its performance for all the shown calibration set sizes, whilst the pixel-wise calibration approach clearly exhibits larger coverage errors as the size of the calibration set decreases. Throughout the data points, we do observe that, also for the larger datasets, Kandinsky methods reduce the outliers in calibration. The crossover between the means of the methods happens around a calibration dataset size of five hundred for this specific scenario. However, it should be pointed out that these results are specific to the model and dataset under consideration, and we expect that the crossover point may shift in different scenarios.

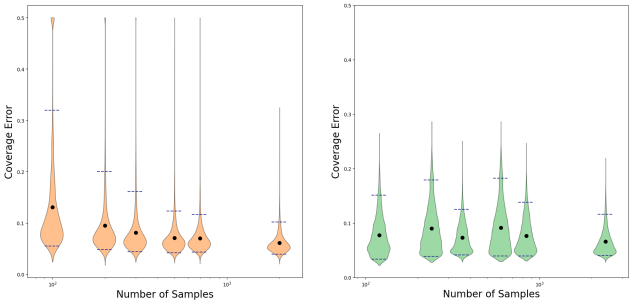

Figure 12. The pixel-wise calibration (left) starts with a higher error, slowly decreasing with increasing calibration dataset size. The Kandinsky method (right) remains roughly stable. For calibration dataset sizes of size 500 or less, Kandinsky is the favorable method.