# MAREA: Active Learning for Object Detection Using Nested Local Embeddings

V.A. Sokolov[1], L.M. Teplyakov[1], and I.A. Matveev[1]

Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, 119333 Russia
sokolov.va@phystech.edu teplyakov.l@ya.ru matveev@frccsc.ru

**Abstract.** Modern active-learning methods for neural-network detectors are limited either to global image embeddings or are incompatible with architectures that use multiscale feature maps. AREA, a method based on local multiscale embeddings, only partially addresses this issue because it includes an embedding dimensionality-reduction stage, which causes the embeddings to lose part of their useful information and slows down the selection procedure. This paper presents the next stage in the development of this method, MAREA (Matryoshka Active learning with Region Embedding Alignment). The method relies on a family of nested object embeddings obtained from a matryoshka-style modification of the YOLOv8 detection head and interleaves features from different pyramid levels so that every prefix remains a consistent object representation. Local features are extracted from intermediate feature maps of the detection head using RoiAlign, after which data selection is performed by object novelty with respect to the already labeled set using coarse-to-fine search. This accelerates the selection procedure. Experiments on four datasets – Pascal VOC, CrowdHuman, COCO2017, and the industrial ind_safety dataset – show that MAREA outperforms AREA and the best known method on average and is especially effective on industrial datasets.

**Keywords:** active learning, object detection, YOLOv8, object embeddings, matryoshka, RoiAlign, coarse-to-fine search

## 1 Introduction

Deep neural networks are currently considered the best-performing solution in most computer-vision problems. At the same time, training modern architectures requires collecting and annotating large amounts of data [1]. In many applications this task is further complicated by the high cost of expert annotation, for example in medical-image analysis or industrial-process monitoring [2,3]. Active learning (AL) addresses this problem by selecting the most informative images for annotation according to the current state of the model.

Object detection is one of the most common practical computer-vision tasks [4]. It requires both classifying objects and localizing them in an image. Annotation for object detection is especially labor-intensive and expensive because each object in an image must be both assigned a class and marked with an accurate

bounding box. In many applied domains, object-detection datasets also have specific video-analytics properties: redundant video streams, rare truly informative events, and the need for frequent fine-tuning [2]. These properties make active learning particularly relevant for object detection.

Acquisition functions can be divided into three groups: uncertainty-based, diversity-based, and hybrid methods. Uncertainty-based methods use information about how confident the model is in its prediction, i.e. how informative an individual image is. Diversity-based methods use information about mutual similarity between images and select a set of images that are as different from each other as possible. Hybrid methods combine these two principles [2]. Uncertainty-based acquisition functions are well represented in the literature, whereas diversity-based functions have been studied less extensively [18]. In object detection for video analytics, the model must recognize objects of different scales in real time and under varying weather and visual conditions. Object embeddings make it possible to account for these properties: because embedding spaces group vectors of similar objects, the relative positions of vectors can be used to thin image streams and distinguish internal clusters within the same object class. However, existing diversity-based methods do not satisfy all of these requirements. In particular, Plug and Play Active Learning (PPAL) [18], one of the strongest published baselines, is incompatible with fast detectors that use multiscale embeddings and has quadratic computational complexity in the number of new examples.

To address these problems, AREA (Active learning with Region Embedding Alignment) was proposed in previous work [19]. AREA extracts local object embeddings from several levels of the YOLOv8 feature pyramid [20] using Region of Interest Align (RoiAlign) [21]. RoiAlign aligns local regions across pyramid levels without quantization loss. New images are selected according to distances between object embeddings from the unlabeled pool and from the already labeled set. Experiments on Pascal VOC and COCO2017 showed a stable improvement over random selection and higher speed than PPAL.

However, AREA has two important limitations. First, its embeddings require preliminary dimensionality reduction, which slows down selection and may lose information. Second, because of the large depth of activation maps, the method must use compact but less informative representations in the space of model prediction probabilities.

This paper proposes MAREA (Matryoshka Active learning with Region Embedding Alignment), extending AREA through nested Matryoshka-style representations. These representations are obtained from a matryoshka-style modification of the YOLOv8 detection head. Nested embeddings make it possible to work directly in the model's hidden-representation space using coarse-to-fine search over shorter embedding prefixes and remove the need for additional dimensionality reduction at the selection stage. The experiments are conducted on VOC, CrowdHuman, COCO2017, and the industrial ind_safety dataset. The results show a substantial improvement in the speed and quality of data selection over AREA and PPAL.

## 2 Problem Statement and Method Taxonomy

Consider pool-based batch active learning. The pool is the set of available but not yet annotated images, and the batch is a subset of $b$ images selected for annotation in one round.

Let $L_0 \subset D$ be the initial labeled set, $U_0 = D \setminus L_0$ be the unlabeled pool, $V$ be the validation set, $O$ be the annotation oracle, $B$ be the annotation budget, $b$ be the batch size, $f_\theta$ be the model with current parameters $\theta$, $A(L_t, U_t, f_\theta)$ be the selection criterion at step $t$, and $Q(F_\theta, V)$ be the model-quality measure with target value $Q^*$.

At each round $t = 1, \ldots, T$, where $T = B/b$, the following steps are performed:

1. Train the model $f_\theta$ on the current labeled set $L_{t-1}$.
2. Select the batch
$$S_t^* = \arg \max_{S \subset U_{t-1},\ |S|=b} A(L_t, U_t, f_\theta). \tag{2.1}$$
In other words, this step selects the subset of $b$ images from the current unlabeled pool that is considered most informative. Each selected image is then sent to the oracle for annotation.
3. Annotate the selected batch:
$$y_x = O(x), \quad \forall x \in S_t^*. \tag{2.2}$$
4. Update the sets:
$$L_t = L_{t-1} \cup \{(x, y_x) \mid x \in S_t^*\}, \qquad U_t = U_{t-1} \setminus S_t^*. \tag{2.3}$$
The newly annotated data are added to the training set, and their unlabeled versions are removed from the candidate pool.
5. Train the model on the updated training set and measure the model quality $Q_t$.

The loop continues until the annotation budget $B$ is exhausted or the target quality $Q^*$ is reached. Formally, the task is

$$\max_{S_1^*, \ldots, S_T^*} Q(f_{\theta_T}) \quad \text{subject to} \quad \begin{cases} |L_T| - |L_0| \le B, \\ Q(f_{\theta_t}) \le Q^*. \end{cases} \tag{2.4}$$

Thus, the goal is to select a sequence of batches that gives the best final model quality under a fixed annotation budget.

The selection criterion $A(L_t, U_t, f_\theta)$ determines which images from the pool $U$ are most valuable for annotation. We briefly review the main families of such criteria and typical examples.

Uncertainty-based methods estimate the model's uncertainty and implement an exploitation strategy [2,5]. For a single image $x$, they define an acquisition score $a(x)$ that can then be aggregated into a batch acquisition criterion $A(L_t, U_t, f_\theta)$.

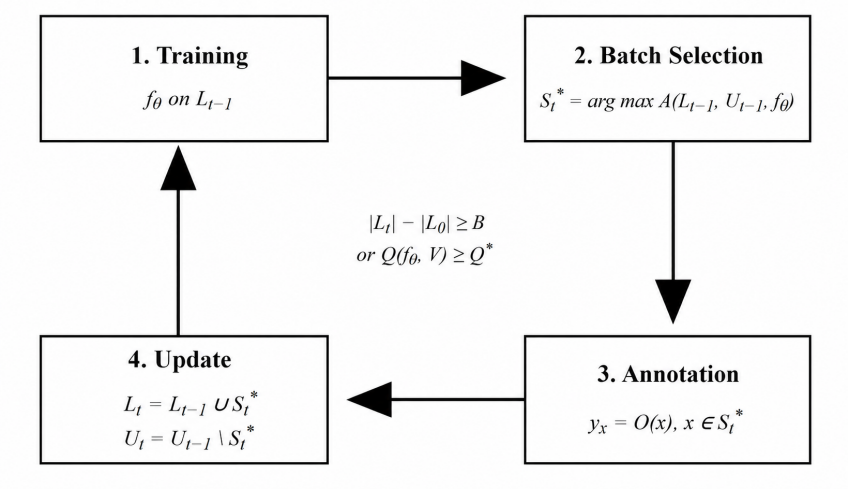



**Fig. 1.** Pool-based batch active-learning scheme.

Entropy Sampling selects images with maximum entropy of the posterior class distribution [5,6]:

$$a(x) = H(p(y \mid x, f_\theta)) = -\sum_{c=1}^{C} p_c(x) \log p_c(x), \tag{2.5}$$

where $p_c(x) = p(y = c \mid x, f_\theta)$.

Least Confidence prioritizes images with the lowest maximum class probability [7]:

$$a(x) = 1 - \max_c p(y = c \mid x, f_\theta). \tag{2.6}$$

Margin Sampling uses the difference between the two highest class probabilities [8,9]:

$$a(x) = 1 - (p_1(x) - p_2(x)), \tag{2.7}$$

where $p_1(x)$ and $p_2(x)$ are the two largest components of the class-probability vector. The smaller the margin, the less certain the model decision.

Disagreement-based methods assign high acquisition scores to images for which different model versions or prediction procedures give substantially different answers [10]. A standard measure of such disagreement is vote entropy:

$$a(x) = -\sum_{c=1}^{C} \frac{V(c,x)}{M} \log \frac{V(c,x)}{M}, \tag{2.8}$$

where $V(c, x)$ is the number of votes for class $c$ and $M$ is the total number of predictions in the committee. Sources of disagreement include Monte Carlo

dropout [11], ensembles [12], augmentations [13], and predictions from different levels of a multiscale representation [14].

Diversity-based methods implement an exploration strategy and select a set of images that covers the feature space with minimal redundancy inside the batch [2,15]. This family includes CoreSet [15], $k$-means++ initialization [16], and density-based approaches [17].

Hybrid methods combine uncertainty and diversity [2,5]. They are usually implemented either as a two-stage procedure or as a single criterion combining both components:

$$A(L_t, U_t, f_\theta) = \lambda A_{\text{unc}}(L_t, U_t, f_\theta) + (1-\lambda) A_{\text{div}}(L_t, U_t, f_\theta), \quad \lambda \in [0,1]. \tag{2.9}$$

The effectiveness of a hybrid method depends on how consistently its uncertainty and diversity components work within one selection procedure and how computationally compatible they are [18].

## 3 Baseline Diversity Selection and AREA

As an example of a hybrid method, consider Plug and Play Active Learning (PPAL), proposed by Yang et al. [18]. The method is two-stage: its uncertainty stage first forms an intermediate candidate set $H_t \subset U_{t-1}$, and its diversity stage then selects the final batch $S_t^*$. In this paper we compare specifically with `PPAL-diversity`.

At the diversity stage, PPAL selects a subset of images that minimizes the maximum pairwise similarity within the set:

$$A_{\text{PPAL-div}}(L_t, U_t, f_\theta) = - \max_{x_i, x_j \in S} S(O_i, O_j), \tag{3.1}$$

so PPAL-diversity selects the batch $S_t^*$ that maximizes this criterion over $H_t$. Here $S(O_i, O_j)$ is the similarity between images $x_i$ and $x_j$ computed from local object representations. Let $f_{a,i}$ and $f_{b,j}$ be embeddings of objects $o_{a,i}$ and $o_{b,j}$. Similarity between object $o_{a,i}$ from image-object set $O_a$ and image $O_b$ is

$$s(o_{a,i}, O_b) = \begin{cases} \max\limits_j \left( \dfrac{f_{a,i} \cdot f_{b,j}}{\|f_{a,i}\| \cdot \|f_{b,j}\|} \right) + 1, & c_{b,j} = c_{a,i}, \\ 0, & \text{if } O_b \text{ has no object of the same class.} \end{cases} \tag{3.2}$$

Image similarity is then computed as a class-confidence-weighted average over all objects:

$$S'(O_a, O_b) = \frac{1}{\sum_i d_{a,i}} \sum_i d_{a,i}\, s(o_{a,i}, O_b). \tag{3.3}$$

The final symmetric similarity is

$$S(O_a, O_b) = \frac{1}{2}\big(S'(O_a, O_b) + S'(O_b, O_a)\big). \tag{3.4}$$

In practice, the diversity stage is implemented in two steps. First, $k$-Center-Greedy solves a coverage problem by selecting a fixed-size subset of centers $S \subset U_{t-1}$ that minimizes the maximum distance from any unlabeled point to the nearest point in $L_{t-1} \cup S$:

$$\min_{S \subset U_{t-1},\, |S|=k} \max_{u \in U_{t-1}} d(u, L_{t-1} \cup S). \tag{3.5}$$

Since the exact solution is NP-hard, a greedy algorithm iteratively adds the point most distant from the current core:

$$s_j = \arg \max_{u \in U_{t-1} \setminus S_{j-1}} \min_{a \in L_{t-1} \cup S_{j-1}} d(u, a), \tag{3.6}$$

where $d(u, a)$ is the distance between embeddings. The complexity is $O(k \cdot |U_{t-1}| \cdot D)$, where $D$ is the embedding dimension.

Second, PPAL applies a modified $k$-means++ algorithm [16]. In classical $k$-means++, the probability of selecting point $u$ is proportional to the squared distance to the nearest already selected center $c \in C$:

$$P(u) = \frac{\min_{c \in C} d^2(u, c)}{\sum_{w \in U_{t-1}} \min_{c \in C} d^2(w, c)}. \tag{3.7}$$

In PPAL [18], the results of $k$-Center-Greedy are used as an initial approximation for centroids, followed by iterative medoid refinement. This optimizes the point distribution while accounting for density, reduces noise sensitivity, and compensates for geometric outliers.

PPAL has several limitations. First, it is poorly compatible with modern detectors that use multiscale representations, because it requires matching objects to a specific feature map for embedding extraction. If activation maps have different channel depths, similarity between the resulting embeddings of different dimensionality cannot be computed directly. Therefore, PPAL uses vectors in the model-prediction probability space. Second, the diversity stage has quadratic complexity in the number of compared images and is practical only for relatively small batches, which increases the number of active-learning iterations. The preliminary filtering stage reduces the constant factor but does not remove the scaling issue: the original method recommends passing $4b$–$6b$ candidates to the diversity stage, where $b$ is the final annotation-batch size. Thus, when only a few active-learning rounds remain, this intermediate set can still cover a large part of the unlabeled pool: for example, if $|L| = 2b$ and $|U| = 8b$, it covers 50%–75% of $U$. Third, objects are compared only within the same predicted class, ignoring similarity between different classes. Fourth, embedding extraction by average pooling is close to RoiAlign with `roi=[1,1]` and does not preserve spatial information, which can hurt performance on overlapping objects.

AREA [19] is naturally aligned with FPN architectures [22] and extracts local object embeddings from multiple pyramid levels using RoiAlign. Embeddings from different layers are concatenated into one vector.

Let $E(x) = \{e_{x,i}^{(l)}\}$ be the set of object embeddings of image $x$ extracted from different pyramid levels, and let $E_{L_{t-1}}$ be the union of embeddings corresponding to the already labeled set. After dimensionality reduction using Principal Component Analysis (PCA), the acquisition score is

$$a_{\mathrm{AREA}}(x, f_\theta) = \max_{e \in \Phi(E(x))} \min_{z \in \Phi(E_{L_{t-1}})} d_{\cos}(e, z), \tag{3.8}$$

where $d_{\cos}$ is cosine distance. The AREA batch-selection criterion is

$$A_{\mathrm{AREA}}(L_t, U_t, f_\theta) = \sum_{x \in S} a_{\mathrm{AREA}}(x, f_\theta). \tag{3.9}$$

Thus, images containing objects most distant from the labeled set in embedding space receive high priority. This approach accounts for local objects at different scales and is better suited to multiscale detectors than PPAL.

Computationally, AREA and MAREA are faster than PPAL because of the structure of the selection procedure:

1. **One-pass stateless greedy selection.** Classical CoreSet and PPAL require $k$ sequential steps. At each step, distances from all unselected candidates to the newly selected center must be recomputed, resulting in $O(k \cdot |U_{t-1}| \cdot D)$ complexity. In AREA, each candidate is scored independently against the fixed labeled set $L_{t-1}$ by a nearest-neighbor rule, reducing selection to one distance-computation pass followed by top-$k$ extraction in linear time $O(|U_{t-1}|)$.
2. **Compatibility with approximate nearest-neighbor search.** Since the reference set in AREA does not change during the round, a single static ANN index such as HNSWlib or Annoy can be built. In PPAL, dynamic addition of new centers at each of the $k$ greedy steps would require rebuilding or updating the index $k$ times, so PPAL has to rely on exact pairwise distances.
3. **Parallelization.** Independent scoring allows distances for all pool candidates to be computed in batches of 1024 vectors, which efficiently accelerates computation.

AREA also has limitations. PCA before distance computation increases selection complexity and may distort the embedding-space structure. In addition, AREA uses embeddings from detector outputs, which contain class probabilities and bounding-box coordinates. These representations are compact but less expressive than embeddings from layers before the detection head, where semantic and spatial information is better preserved [22,23].

## 4 Proposed Method

To address AREA's limitations, we use matryoshka-style nested embeddings produced by a modification of the YOLOv8 detection head, and define MAREA as an extension of AREA to this representation structure. Section 3.1 describes the embedding source, and Section 3.2 describes MAREA itself.

### 4.1 Source of Nested Embeddings

The source of nested representations in this paper is a modification of the YOLOv8 detection head. The standard YOLOv8 detection head [20] receives feature maps from several pyramid levels and independently produces regression and classification outputs for each level using the full channel depth of the input tensor. In the matryoshka-style modification, the same head is additionally trained on a family of channel prefixes: for a set of divisors $D = \{d_1, \ldots, d_K\}$, the first $\lfloor C_l/d_k \rfloor$ channels are taken from the feature map, auxiliary regression and classification outputs are computed through the same head, and the final loss is a weighted sum of standard YOLOv8 detection losses over all depths.

The full depth corresponds to ordinary detector operation and preserves standard YOLOv8 behavior. Shorter channel prefixes form a consistent family of representations and are used later only as an embedding source. No additional trainable parameters are introduced: only the input channels of the first convolution in each branch are sliced, while existing weight tensors are reused as slices. In our implementation the divisors are `8,4,1` and the prefix weights are $[0.2, 0.4, 1.0]$. For stable multi-mode training, we use shared assignment of predictions to targets, frozen BatchNorm running statistics during auxiliary passes, a loss-weight schedule, and stochastic depth selection at each step.

For each detected object, RoiAlign [21] is applied to intermediate detection-head feature maps using its bounding box, yielding a local object representation at each pyramid level. To keep prefixes of the final vector meaningful and consistent across pyramid levels, we use interleaving: each level representation is split into equal channel blocks, and the final embedding is formed by grouping blocks with the same indices across all levels. Therefore, the first 1/8 or 1/4 of the final embedding contains the corresponding fraction of features from all pyramid levels, not only from one level. Thus, each object $o$ receives a family of nested embeddings

$$e(o) = \{e_{1/8}(o),\, e_{1/4}(o),\, e_1(o)\}, \tag{4.1}$$

where $e_{1/8}$ and $e_{1/4}$ are shorter prefixes and $e_1$ is the full-depth representation. This family is used as the embedding source in MAREA.

### 4.2 MAREA (Matryoshka Active learning with Region Embedding Alignment)

The transition from AREA to MAREA consists of replacing the object-embedding source and the search procedure. The selection criterion keeps the meaning of AREA: priority is given to images containing objects that are farthest from the already labeled set. Instead of compact detector-output representations, MAREA uses embeddings from intermediate feature maps at the input to the matryoshka-trained detection head. This makes it possible to compute distances in a more informative feature space and to use coarse-to-fine search over nested prefixes.

The experiments use a fixed extraction and search configuration: YOLOv8s, feature maps `/model.15/cv2/act/Mul`, `/model.18/cv2/act/Mul`, and `/model.`

`21/cv2/act/Mul`, RoiAlign with output size $4 \times 4$, channel-block interleaving before the final vector is formed, L2 normalization without PCA, and HNSW [24] with cosine distance. The depth divisors are `8,4,1`; coarse-to-fine search follows $D/8 \rightarrow D/4 \rightarrow D$: the coarsest prefix forms $4b$ candidates, the intermediate prefix keeps $2b$, and the full embedding chooses a batch of size $b$.

Formally, MAREA is defined as follows:

1. Train a YOLOv8s detector with the matryoshka-style detection head on the current labeled set $L_{t-1}$. In the main configuration, shared assignment is enabled, BatchNorm statistics are frozen during auxiliary passes, prefix weights are $[0.2, 0.4, 1.0]$, auxiliary passes start at epoch 30, the auxiliary-weight warmup lasts 10 epochs, two depth modes are used at each step, and the depth divisors are `8,4,1`.
2. For every image from $L_{t-1}$ and $U_{t-1}$, obtain object predictions and bounding boxes.
3. For every detected object, extract local representations from the selected feature maps using RoiAlign of size $4 \times 4$ and form nested embeddings by interleaving:
$$e(o) = \{e_{1/8}(o), e_{1/4}(o), e_1(o)\}. \tag{4.2}$$
4. Form the labeled embedding set $E_L$ and unlabeled-pool embedding set $E_U$. Before search, embeddings are L2-normalized and PCA is not applied.
5. For each image $x \in U_{t-1}$, compute object novelty analogously to AREA. Let $E(x)$ be the set of object embeddings of image $x$. In full-search mode,
$$a_{\mathrm{MAREA}}(x, f_\theta) = \max_{e \in E(x)} \min_{z \in E_L} d_{\cos}(e_1, z_1), \tag{4.3}$$
where $e_1$ and $z_1$ are full-depth representations of the unlabeled and labeled objects. Nearest objects in $E_L$ are found with HNSW. In coarse-to-fine mode, search first runs over $e_{1/8}$, then refines over $e_{1/4}$, and computes the final score over $e_1$.
6. Define the batch-selection criterion as the sum of selected-image scores:
$$A_{\mathrm{MAREA}}(L_t, U_t, f_\theta) = \sum_{x \in S} a_{\mathrm{MAREA}}(x, f_\theta). \tag{4.4}$$
7. Select the batch:
$$S_t^* = \arg \max_{S \subset U_{t-1},\ |S|=b} A_{\mathrm{MAREA}}(L_t, U_t, f_\theta). \tag{4.5}$$
8. Send the selected images to the oracle, add the labels to $L_t$, and remove the images from $U_t$:
$$y_x = O(x), \quad x \in S_t^*, \tag{4.6}$$
$$L_t = L_{t-1} \cup \{(x, y_x) \mid x \in S_t^*\}, \qquad U_t = U_{t-1} \setminus S_t^*. \tag{4.7}$$

**Table 1.** Datasets used in the experiments.

| Dataset | Domain | Classes | Train | Val |
|---|---|---|---|---|
| Pascal VOC | general domain | 20 | 16551 | 4952 |
| COCO2017 | general domain | 80 | 118287 | 5000 |
| CrowdHuman | images of people | 1 | 15000 | 4370 |
| ind_safety | industrial safety-monitoring dataset | 3 | 69925 | 7029 |

## 5 Experimental Protocol

Experiments use a YOLOv8s detector initialized with weights pretrained on Open Images v7. For all datasets, the input image size is $640 \times 640$, the batch size is 48, training lasts 65 epochs, and standard YOLOv8 hyperparameters are used. Computations are performed on NVIDIA RTX 3090 GPUs and on a server with four NVIDIA H100 GPUs.

Four datasets are considered. Their characteristics are shown in Table 1.

Active learning is performed on nested fractions of the full training set:

$$0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7. \tag{5.1}$$

An active-learning chain is a sequence of training and data-acquisition steps over these nested fractions: the selection result at the previous step defines the training set for the next one.

At each step, the model is trained on the current labeled fraction, and its weights are then used to select the next portion of data. For the first acquisition, from 0.2 to 0.3, the model trained on a random 0.2 subset is used. Later transitions use the model trained on the previous fraction formed by the corresponding selection method.

To separate the effect of active selection from the effect of the training scheme, random-selection control chains are also used. They have the same nested fractions, but new images are added at random. We compare ordinary random YOLOv8s training (RANDOM) and a random chain where the detector is trained with the matryoshka scheme (RANDOM-MAT).

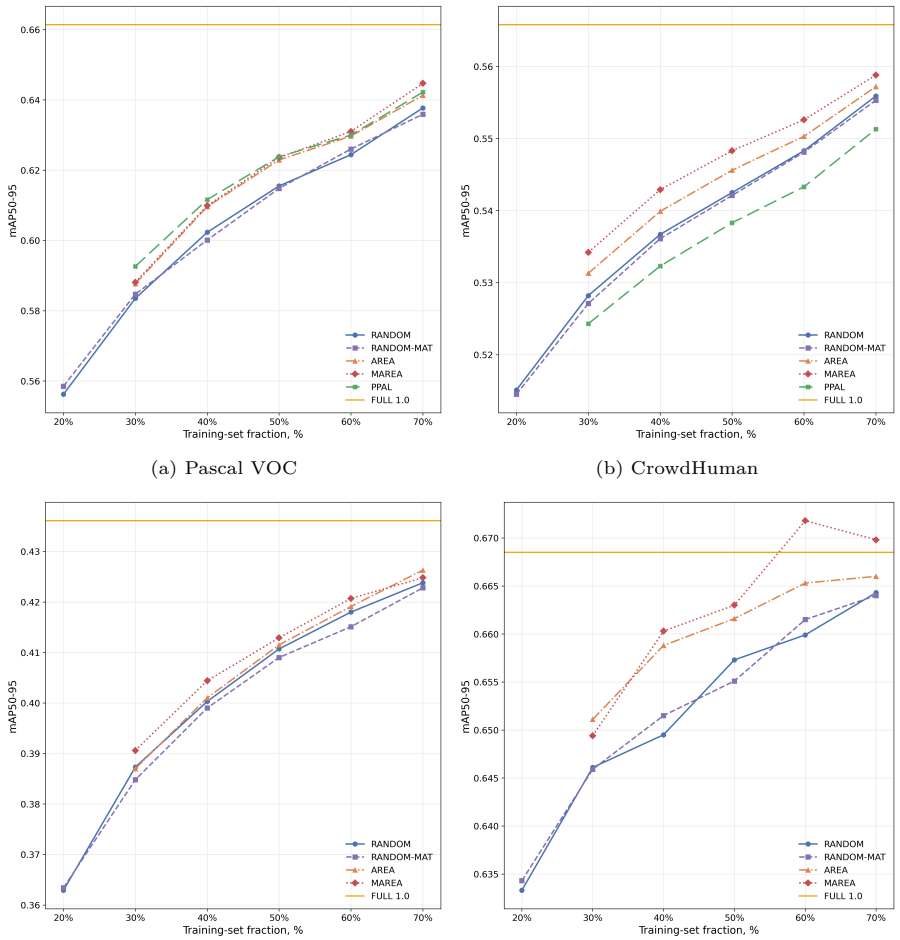
## 6 Active-Learning Chain Results

Figure 2 shows mean mAP50-95 values over four independent random seeds. Tables report the mean and the 95% confidence interval. The orange horizontal line corresponds to YOLOv8s trained on the full training set.

Training on the full dataset reaches **0.6614** mAP50-95 on Pascal VOC, **0.5658** on CrowdHuman, **0.4361** on COCO2017, and **0.6685** on ind_safety.

Table 6 reports the average difference in mAP50-95 between MAREA and the baselines over the common fractions 0.3–0.7. This is a discrete analogue of the difference between areas under the active-learning curves.

The comparison between RANDOM and RANDOM-MAT shows that the matryoshka detector modification itself almost does not change detection quality




(a) Pascal VOC (b) CrowdHuman

(c) COCO2017 (d) ind_safety

**Fig. 2.** Comparison of active-learning chains: (a) Pascal VOC; (b) CrowdHuman; (c) COCO2017; (d) ind_safety.

**Table 2.** Comparison of active-learning chains on Pascal VOC.

| Fraction | RANDOM | RANDOM-MAT | AREA | MAREA | PPAL |
|---|---|---|---|---|---|
| 0.2 | 0.5562 ± 0.0080 | **0.5585 ± 0.0039** | – | – | – |
| 0.3 | 0.5835 ± 0.0032 | 0.5847 ± 0.0013 | 0.5877 ± 0.0078 | 0.5881 ± 0.0054 | **0.5926 ± 0.0062** |
| 0.4 | 0.6023 ± 0.0033 | 0.6001 ± 0.0037 | 0.6096 ± 0.0069 | 0.6099 ± 0.0024 | **0.6116 ± 0.0038** |
| 0.5 | 0.6155 ± 0.0026 | 0.6148 ± 0.0025 | 0.6229 ± 0.0032 | 0.6236 ± 0.0016 | **0.6238 ± 0.0042** |
| 0.6 | 0.6244 ± 0.0025 | 0.6260 ± 0.0024 | 0.6297 ± 0.0028 | **0.6310 ± 0.0017** | 0.6300 ± 0.0028 |
| 0.7 | 0.6377 ± 0.0040 | 0.6359 ± 0.0017 | 0.6413 ± 0.0028 | **0.6447 ± 0.0039** | 0.6422 ± 0.0029 |

**Table 3.** Comparison of active-learning chains on CrowdHuman.

| Fraction | RANDOM | RANDOM-MAT | AREA | MAREA | PPAL |
|---|---|---|---|---|---|
| 0.2 | **0.5151 ± 0.0011** | 0.5145 ± 0.0011 | – | – | – |
| 0.3 | 0.5282 ± 0.0005 | 0.5271 ± 0.0008 | 0.5313 ± 0.0008 | **0.5342 ± 0.0011** | 0.5243 ± 0.0009 |
| 0.4 | 0.5367 ± 0.0006 | 0.5361 ± 0.0009 | 0.5399 ± 0.0013 | **0.5429 ± 0.0017** | 0.5323 ± 0.0024 |
| 0.5 | 0.5425 ± 0.0008 | 0.5421 ± 0.0005 | 0.5456 ± 0.0005 | **0.5483 ± 0.0004** | 0.5383 ± 0.0010 |
| 0.6 | 0.5483 ± 0.0005 | 0.5481 ± 0.0006 | 0.5503 ± 0.0007 | **0.5526 ± 0.0002** | 0.5433 ± 0.0015 |
| 0.7 | 0.5559 ± 0.0003 | 0.5553 ± 0.0007 | 0.5572 ± 0.0005 | **0.5588 ± 0.0011** | 0.5513 ± 0.0011 |

**Table 4.** Comparison of active-learning chains on COCO2017.

| Fraction | RANDOM | RANDOM-MAT | AREA | MAREA |
|---|---|---|---|---|
| 0.2 | 0.3629 ± 0.0023 | **0.3634 ± 0.0005** | – | – |
| 0.3 | 0.3873 ± 0.0013 | 0.3848 ± 0.0032 | 0.3870 ± 0.0038 | **0.3906 ± 0.0016** |
| 0.4 | 0.4003 ± 0.0028 | 0.3990 ± 0.0029 | 0.4010 ± 0.0037 | **0.4044 ± 0.0014** |
| 0.5 | 0.4107 ± 0.0010 | 0.4090 ± 0.0018 | 0.4115 ± 0.0014 | **0.4129 ± 0.0034** |
| 0.6 | 0.4180 ± 0.0027 | 0.4151 ± 0.0025 | 0.4191 ± 0.0009 | **0.4207 ± 0.0027** |
| 0.7 | 0.4238 ± 0.0011 | 0.4228 ± 0.0005 | **0.4263 ± 0.0007** | 0.4248 ± 0.0018 |

**Table 5.** Comparison of active-learning chains on ind_safety.

| Fraction | RANDOM | RANDOM-MAT | AREA | MAREA |
|---|---|---|---|---|
| 0.2 | 0.6333 ± 0.0069 | **0.6343 ± 0.0048** | – | – |
| 0.3 | 0.6461 ± 0.0060 | 0.6459 ± 0.0039 | **0.6511 ± 0.0018** | 0.6494 ± 0.0034 |
| 0.4 | 0.6495 ± 0.0022 | 0.6515 ± 0.0024 | 0.6588 ± 0.0014 | **0.6603 ± 0.0050** |
| 0.5 | 0.6573 ± 0.0036 | 0.6551 ± 0.0049 | 0.6616 ± 0.0035 | **0.6630 ± 0.0058** |
| 0.6 | 0.6599 ± 0.0035 | 0.6615 ± 0.0021 | 0.6653 ± 0.0064 | **0.6718 ± 0.0041** |
| 0.7 | 0.6643 ± 0.0019 | 0.6640 ± 0.0047 | 0.6660 ± 0.0036 | **0.6698 ± 0.0007** |

**Table 6.** Average gains of MAREA over the baselines at fractions 0.3–0.7.

| Dataset | MAREA - RANDOM | MAREA - AREA | MAREA - PPAL |
|---|---|---|---|
| Pascal VOC | +0.0068 | +0.0012 | -0.0006 |
| CrowdHuman | +0.0050 | +0.0025 | +0.0094 |
| COCO2017 | +0.0027 | +0.0017 | – |
| ind_safety | +0.0075 | +0.0023 | – |

under random data selection. The curves of these two control chains are close in most cases, which means that the main effect of MAREA is associated with using nested object embeddings in the active-selection criterion, not with the modified YOLOv8 being systematically better on random subsets.

On Pascal VOC, MAREA outperforms AREA on average and nearly closes the gap with PPAL: PPAL remains a strong baseline at early fractions, but MAREA becomes better at larger fractions. On CrowdHuman, the result is more direct: MAREA outperforms both AREA and PPAL, indicating the benefit of local representations from the YOLOv8 head in a specialized domain with dense objects of one class.

On COCO2017, MAREA also gives a positive average gain over AREA, although the advantage decreases at the largest fraction. On ind_safety, MAREA shows the largest gain over RANDOM and at fractions 0.6 and 0.7 exceeds the model trained on the full training set. This is important because ind_safety is an industrial video-stream dataset: within-class visual clusters arise because of glare, snow, rain, and other acquisition conditions, and active learning can select underrepresented modes rather than merely additional images.

MAREA curves mostly remain below the full-training line at fraction 1.0, as expected: active learning uses only part of the annotations. However, the gap to the full model decreases as the labeled fraction grows, and the advantage of MAREA over AREA shows that the improvement is related not only to training-set size, but also to the quality of the selection criterion.

In terms of runtime, MAREA also retains a practical advantage over PPAL. On CrowdHuman, the full split-preparation time is 516 seconds for MAREA versus 1029 seconds for PPAL, so PPAL is about 1.99 times slower. Compared with AREA, the full split-preparation cycle is also faster for MAREA: 516 seconds versus 730 seconds, i.e. about 1.41 times. The embedding-selection stage in MAREA is itself slower than in AREA because of multistage search, but the full cycle is faster because the expensive PCA preprocessing step is removed. For COCO2017 and ind_safety, MAREA is faster than AREA: 2488 seconds versus 2712 seconds, so AREA is about 1.09 times slower.

## 7 Conclusion

This paper proposed MAREA, an active-learning method for object detection that extends AREA using nested matryoshka-style local embeddings and coarse-to-fine search over their prefixes. The representation source is a modification of the YOLOv8 detection head. Unlike AREA, which uses compact detector-output representations, MAREA extracts object embeddings from intermediate feature maps and preserves their nested channel-depth structure. This enables distance computation in a more informative feature space and coarse-to-fine search over shorter prefixes.

The main result is that MAREA improves active-selection quality over AREA on all four datasets and also compares favorably with PPAL. The summary table shows positive average gains over the common fractions in all experiment series.

The largest gain over random selection is observed on ind_safety, and the largest gain over PPAL is observed on CrowdHuman. This indicates that the method is useful not only as an internal improvement of AREA, but also as a competitive alternative external control method for active learning in object detection.

The result on the industrial ind_safety dataset is especially important. It consists of video-stream frames and contains heterogeneous visual modes within classes caused by glare, snow, rain, and other acquisition conditions. In this setting, active learning can select underrepresented data modes, so model quality may improve not only because the number of images increases, but also because the training-set composition becomes better.

Future work is connected with a hybrid two-stage algorithm that uses disagreement between nested embeddings of different depths as an additional uncertainty signal, and with extending nested embeddings to a wider class of detectors with feature-pyramid structure.

## References

1. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539
2. Garcia, D., Carias, J., Adao, T., Jesus, R., Cunha, A., Magalhaes, L.G.: Ten years of active learning techniques and object detection: A systematic review. Applied Sciences **13**(19), 10667 (2023). https://doi.org/10.3390/app131910667
3. Ren, P., Xiao, Y., Chang, X., Huang, P., Li, Z., Chen, X., Wang, X.: A survey of deep active learning. ACM Computing Surveys **54**(9), 1–40 (2021). https://doi.org/10.1145/3472291
4. Zou, Z., Chen, K., Shi, Z., Guo, Y., Ye, J.: Object detection in 20 years: A survey. Proceedings of the IEEE **111**(3), 257–276 (2023). https://doi.org/10.1109/JPROC.2023.3238524
5. Settles, B.: Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, pp. 1–67 (2009)
6. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal **27**(3), 379–423 (1948). https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
7. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 3–12. Dublin, Ireland (1994). https://doi.org/10.1007/978-1-4471-2099-5_1
8. Joshi, A.J., Porikli, F., Papanikolopoulos, N.: Multi-class active learning for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2372–2379. Miami, USA (2009). https://doi.org/10.1109/CVPR.2009.5206627
9. Scheffer, T., Decomain, C., Wrobel, S.: Active hidden Markov models for information extraction. In: Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis, pp. 309–318. Cascais, Portugal (2001). https://doi.org/10.1007/3-540-44816-0_31
10. Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: Proceedings of the 5th Annual Workshop on Computational Learning Theory, pp. 287–294. Pittsburgh, USA (1992). https://doi.org/10.1145/130385.130417

11. Gal, Y., Islam, R., Ghahramani, Z.: Deep Bayesian active learning with image data. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 1183–1192. Sydney, Australia (2017)
12. Beluch, W.H., Genewein, T., Nurnberger, A., Kohler, J.M.: The power of ensembles for active learning in image classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9368–9377. Salt Lake City, USA (2018). https://doi.org/10.1109/CVPR.2018.00976
13. Yu, W., Zhu, S., Yang, T., Chen, C.: Consistency-based active learning for object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 3951–3960. New Orleans, USA (2022). https://doi.org/10.1109/CVPRW56347.2022.00441
14. Yuan, T., Wan, F., Fu, M., Liu, J., Xu, S., Ji, X., Ye, Q.: Multiple instance active learning for object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5330–5339. Nashville, USA (2021). https://doi.org/10.1109/CVPR46437.2021.00529
15. Sener, O., Savarese, S.: Active learning for convolutional neural networks: A core-set approach. In: Proceedings of the 6th International Conference on Learning Representations, pp. 1–13. Vancouver, Canada (2018). `https://openreview.net/forum?id=H1aIuk-RW`, last accessed 2026/06/24
16. Arthur, D., Vassilvitskii, S.: $k$-means++: The advantages of careful seeding. In: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035. New Orleans, USA (2007)
17. Nguyen, H.T., Smeulders, A.: Active learning using pre-clustering. In: Proceedings of the 21st International Conference on Machine Learning, pp. 79–86. Banff, Canada (2004). https://doi.org/10.1145/1015330.1015349
18. Yang, C., Huang, L., Crowley, E.J.: Plug and play active learning for object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 17784–17793. Seattle, USA (2024). https://doi.org/10.1109/CVPR52733.2024.01683
19. Sokolov, V.A., Matveev, I.A., Teplyakov, L.M.: Extracting embeddings for active learning corresponding to regions of interest in an image. Pattern Recognition and Image Analysis (2025). https://doi.org/10.1134/S1054661825600772
20. Jocher, G., Chaurasia, A., Qiu, J.: Ultralytics YOLOv8 (2023). `https://github.com/ultralytics/ultralytics`, last accessed 2026/06/24
21. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988. Venice, Italy (2017). https://doi.org/10.1109/ICCV.2017.322
22. Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117–2125. Honolulu, USA (2017). https://doi.org/10.1109/CVPR.2017.106
23. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(8), 1798–1828 (2013). https://doi.org/10.1109/TPAMI.2013.50
24. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(4), 824–836 (2020). https://doi.org/10.1109/TPAMI.2018.2889473