Urban Object Recognition in Drone Imagery: A Hybrid Pipeline Combining YOLO and Optimal Transport

Fatima-Zahra Aklila*, Mohsen Akoum*, Bechir El Hannachi*, Achraf Ktaib*, Noah Parisse*
*CentraleSupélec, Université Paris-Saclay

***Abstract*—In this work, we propose a hybrid framework for object recognition in aerial urban imagery that combines YOLO-based object detection with optimal transport techniques for post-hoc classification. In our approach, objects are first detected using the YOLO model, which provides bounding boxes and initial class predictions. To refine these predictions, we extract deep visual features from the detected regions using a ResNet50 network and apply a simple optimal transport-based classifier to match these features to class prototypes. This method leverages the distributional alignment capabilities of optimal transport to improve classification robustness, particularly in complex urban environments characterized by high object density, occlusions, and inter-class visual similarities. We evaluate our approach on the VisDrone dataset of aerial urban scenes including vehicles, buildings, pedestrians, and other common urban elements. While experimental results show that our method doesn't outperform YOLO's built-in classification, it shows promise, especially that we could only feed it a very small part of the data.**

***Index Terms*—Computer Vision, Object Detection, Classification, Optimal Transport**

I. Introduction

The increasing availability of high-resolution aerial imagery captured by drones has enabled new applications in urban monitoring, infrastructure planning, and real-time surveillance. A central challenge in these applications is *object recognition*—the task of accurately detecting and classifying objects such as vehicles, pedestrians, and buildings from overhead views. However, object recognition in drone imagery is particularly difficult due to the small size of objects, their high density, occlusions, and strong visual similarities across classes. These factors can significantly reduce the performance of standard end-to-end deep learning models.

To address these challenges, we propose a hybrid recognition framework that explicitly decouples object *detection* from *classification*. Our approach first leverages a state-of-the-art object detector, YOLOv11, to localize candidate objects in drone-captured urban scenes. We introduce a second classification stage that combines deep feature extraction with optimal transport theory to refine class predictions.

The pipeline of our method is structured as follows:

1) **Detection:** Objects are detected using the YOLOv11 model, yielding bounding boxes and one class (object).
2) **Feature Extraction:** From each detected region, high-level visual features are extracted using a pre-trained ResNet50 network, fine-tuned on our dataset.
3) **Classification via Optimal Transport:** We compute the distance between the extracted features and a set of class prototypes using an entropic regularized Wasserstein distance (Sinkhorn divergence). The final class is selected based on the distance to the prototypes.

This modular architecture allows us to use the geometric robustness of optimal transport to better handle ambiguous or occluded objects and to enhance classification accuracy in real-world aerial scenes.

A. Mathematical notations

- The inner product on $\mathbb{R}^n$:
  $\forall x, y \in \mathbb{R}^n, \langle x, y\rangle = \sum_{i=1}^n x_i y_i$
- The L^1 and L^2 norms on $\mathbb{R}^n$:

$$\|x\|_1 = \sum_{i=1}^n |x_i|$$

$$\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$$

- The element-wise product and quotient of two vectors of $\mathbb{R}^n$:

$$\forall x, y \in \mathbb{R}^n, x \cdot y = (x_i y_i)_{i \in [\![1,n]\!]}$$

$$\forall x \in \mathbb{R}^n, \forall y \in (\mathbb{R}^*)^n, x \cdot / y = (x_i / y_i)_{i \in [\![1,n]\!]}$$

- $\mathbb{R}^{n\times p}$ (resp. $\mathbb{R}_+^{n\times p}$): The set of $n \times p$ real (resp. non-negative) matrices; $n, p \in \mathbb{N}^*$
- The Frobenius inner product on $\mathbb{R}^{n\times p}$:
  $\forall A, B \in \mathbb{R}^{n\times p}, \langle A, B\rangle = \sum_{i=1}^n \sum_{j=1}^p a_{i,j} b_{i,j}$
- The element-wise exponential of a matrix of $\mathbb{R}^{n\times p}$:
  $\forall A \in \mathbb{R}^{n\times p}, \exp(A) = (e^{a_{i,j}})_{\substack{1\le i\le n \\ 1\le j\le p}}$
- $\mathbb{R}^{n_1\times\cdots\times n_k}$: The set of $n_1 \times \ldots \times n_k$ real tensors; $n_1, \ldots, n_k \in \mathbb{N}^*$
- $\mathbf{1}_n$ (resp. $\mathbf{1}_{n\times p}$) is the vector of $\mathbb{R}^n$ (resp. the matrix of $\mathbb{R}^{n\times p}$) with all one entries

- The probability simplex of $\mathbb{R}^n$:
  $\Delta^n = \{x \in \mathbb{R}_+^n | \langle x, \mathbf{1}_n \rangle = 1\}$
- In a measurable space $(E, \mathcal{E})$, the dirac mass at x is the measure defined by:
  $\forall B \in \mathcal{E}, \delta_x(B) = \mathbb{1}_{x \in B}$
- The entropy of a discrete probability measure $\mathbb{P}$ on $(E, \mathcal{E})$ is:
  $$h(\mathbb{P}) = - \sum_{x \in \text{supp}(\mathbb{P})} \mathbb{P}(x) \log \mathbb{P}(x)$$
  By extension, when $a \in \Delta^n$, it represents a probability measure with finite support and we can define its entropy as:
  $$h(a) = - \sum_{i=1}^{n} a_i \log(a_i)$$
  Similarly, we can define the entropy of a non-negative matrix or tensor in the probability simplex.
- The Kullback-Leibler divergence of two discrete probability measures $\mathbb{P}$ and $\mathbb{Q}$ supported in $\mathcal{X}$:
  $$KL(\mathbb{P}||\mathbb{Q}) = \sum_{x \in \mathcal{X}} \mathbb{P}(x) \log \left(\frac{\mathbb{P}(x)}{\mathbb{Q}(x)} \right)$$
  Likewise, we can define the KL divergence of two vectors, matrices or tensors in the probability simplex.

B. Formal Problem Definition

Let an input aerial image be denoted by $I \in \mathbb{R}^{3 \times H \times W}$. The objective is to detect and classify all object instances $\{(\mathbf{b}_i, y_i)\}_{i=1}^N$, where:

- $\mathbf{b}_i \subset I$ is the bounding box corresponding to the i-th detected object,
- $y_i \in \mathcal{C}$ is the class label assigned to object i, from a finite label set $\mathcal{C}$.

The remainder of this paper is organized as follows. In Section 2, we review the most relevant related work on object detection and optimal-transport-based classification. Section 3 focuses on the object detection component: we describe the YOLOv11 architecture in detail, explain its foundational blocks and its loss function. Section 4 turns to the classification stage and introduces optimal transport theory and the entropy-regularized Sinkhorn distance. In Section 5, we define all evaluation metrics used throughout our experiments. Section 6 presents our experimental results: we first describe the dataset and then outline each of the approaches we tested. We analyze their performance on our test set and compare them against the YOLOv11 baseline. Finally, Section 7 summarizes our findings and offers concluding remarks.

II. Related Work

A. Object Detection in Aerial Imagery

Object detection in aerial imagery presents unique challenges due to factors such as small object sizes, dense scenes, varying scales, and arbitrary orientations. Recent advancements in object detection architectures have aimed to address these challenges, particularly through enhancements in both one-stage and two-stage detection frameworks.

1) One-Stage Detectors: The YOLO (You Only Look Once) family of models has been widely adopted for real-time object detection tasks. YOLOv11, the latest iteration in the series, introduces architectural innovations like the C3k2 block, Spatial Pyramid Pooling - Fast (SPPF), and Convolutional block with Parallel Spatial Attention (C2PSA). These enhancements contribute to improved feature extraction and computational efficiency, making YOLOv11 suitable for various computer vision tasks, including object detection in aerial imagery [1].

Building upon YOLOv11, MASF-YOLO incorporates modules such as Multi-scale Feature Aggregation Module (MFAM), Improved Efficient Multi-scale Attention Module (IEMA), and Dimension-Aware Selective Integration Module (DASI). These additions specifically target the detection of small objects in UAV images, leading to significant improvements in detection accuracy while maintaining model efficiency [2].

Other adaptations of YOLO architectures have also been proposed to enhance aerial object detection. For instance, YOLOv5s-DSD introduces the SPDA-C3 structure and Res-DHead detection head to improve the detection of small and densely packed objects in complex backgrounds [3]. Similarly, LEAF-YOLO focuses on lightweight and efficient detection suitable for edge devices, achieving real-time performance on UAV platforms [4].

2) Two-Stage Detectors: Two-stage object detectors, such as the R-CNN family, have been instrumental in achieving high detection accuracy, particularly in complex scenes. These models operate by first generating region proposals and then classifying and refining these proposals.

Faster R-CNN, a prominent two-stage detector, has been enhanced with Feature Pyramid Networks (FPN) to better handle objects at multiple scales. This combination has proven effective in aerial imagery, where object sizes can vary significantly [5].

Recent advancements have introduced anchor-free approaches to address the limitations of predefined anchor boxes. TARDet (Two-stage Anchor-Free Rotating Object Detector) is one such model that generates coarse positioning boxes in an anchor-free manner and refines them using spatial refinement modules. This approach is particularly beneficial for detecting objects with arbitrary orientations in aerial images [6].

Another notable two-stage method is AFOD (Anchor-Free Orientation Detection), which eliminates anchor configurations and introduces supplementary angle parameters to accommodate the identification of rotating frame objects. By integrating spatial and channel attention modules, AFOD enhances the extraction of critical features, leading to improved regression and classification precision [7].

Furthermore, transformer-based approaches have been explored to capture inter-object relationships. By integrating transformers into two-stage detectors, these models can refine classification and regression outcomes by considering the spatial distribution and semantic relationships between objects.

These advancements collectively contribute to the evolving landscape of object detection in aerial imagery, offering various approaches to tackle the inherent challenges of this domain.

B. Deep Feature Extraction for Aerial Image Classification

Deep convolutional neural networks (CNNs) have been extensively used for feature extraction in remote sensing and aerial image analysis. Pre-trained models like ResNet50 have been employed to extract high-level features from aerial images, facilitating tasks such as scene classification and object recognition [8]. Fine-tuning these models on domain-specific datasets can further enhance their performance, capturing more relevant features for the target application.

C. How Optimal Transport Enhances Object Classification

Object classification in images presents several significant challenges that affect the accuracy and robustness of machine learning models.

One major difficulty is intra-class variability, where the same object can appear in different colors, shapes, orientations, or scales depending on the context. Conversely, inter-class similarity can occur when visually distinct categories share common features, making it hard to distinguish between them. Additionally, variations in lighting conditions, image quality, and the presence of noise can distort visual information and hinder feature extraction. Another common issue is class imbalance, where some classes are underrepresented in the training data, leading to biased learning. The presence of labeling errors or noisy annotations can also mislead the model during training. Finally, if a model becomes too dependent on the background context of objects instead of focusing on their intrinsic features, its performance may degrade when tested in different environments. These challenges require the development of robust classification techniques that can learn invariant and discriminative features while generalizing well across varied and real-world scenarios.

In response to the challenges of object classification, optimal transport (OT) theory has emerged as a promising solution, particularly for addressing distributional mismatches between datasets. OT provides a principled way to compare probability distributions by computing the most efficient way to morph one distribution into another, which is especially useful when dealing with domain shifts, class imbalance, or noisy data [9].

In image classification, OT can be applied to align feature distributions from different layers or datasets, ensuring better generalization across domains and reducing sensitivity to contextual variations. Furthermore, OT-based distances, such as the Wasserstein distance, offer a robust alternative to classical metrics (e.g., Euclidean distance) when comparing high-dimensional feature spaces, especially under intra-class variability or inter-class similarity. By integrating OT into training or feature fusion strategies, models can become more resilient to transformations, occlusions, and distributional noise, ultimately leading to improved classification performance in complex visual tasks [10].

III. YOLOv11: Advanced Real-Time Object Detection Architecture

YOLOv11, introduced in late 2024, represents a significant advancement in the YOLO (You Only Look Once) series, enhancing both speed and accuracy in real-time object detection tasks. Building upon the foundations of its predecessors, YOLOv11 integrates novel architectural components and refined loss functions to address challenges in detecting small objects and complex scenes.
Let $I \in \mathbb{R}^{3\times H\times W}$ be the input RGB image (usually $H = W = 640$ or $H = W = 512$). The image is normalized and padded/resized as necessary to fit the input dimensions of the model. YOLOv11 predicts bounding boxes, objectness scores, and class probabilities at multiple scales (typically 3 scales: small, medium, and large), corresponding to feature maps of different resolutions.

Let:

- $S \in \{P_1, P_2, P_3\}$ be the set of output feature map scales.
- Each scale $P_i \in \mathbb{R}^{C_i\times H_i\times W_i}$, where:
  - H_i, W_i are the spatial resolutions (e.g., $80\times 80, 40\times 40, 20\times 20$)
  - C_i is the number of channels and depends on the number of anchors and classes.

Let:

- $A =$ number of anchors (or boxes) per grid cell (typically 3),
- $C =$ number of object classes.

Each anchor prediction contains:

- 4 box coordinates: (x, y, w, h),
- 1 objectness score $o \in [0, 1]$,
- C class scores $(p_1, ..., p_C) \in [0, 1]^C$

Thus, each grid cell at each scale outputs a vector of size:

$$C_i = A(4 + 1 + C) = A(5 + C)$$

Hence, for all 3 feature maps, the output of YOLOv11 can be written as a set:

$$\mathcal{Y} = \left\{ Y_i \in \mathbb{R}^{A(5+C)\times H_i\times W_i} \mid i = 1, 2, 3 \right\}$$

Each prediction $\hat{y}_{ijk} \in \mathbb{R}^{5+C}$ at grid cell (i, j), anchor k, consists of:

$$\hat{y}_{ijk} = \left[\hat{x}, \hat{y}, \hat{w}, \hat{h}, \hat{o}, \hat{p}_1, ..., \hat{p}_C \right]$$

where:

- $(\hat{x}, \hat{y})$: predicted center coordinates,
- $(\hat{w}, \hat{h})$:predicted width and height,
- $\hat{o}$: objectness score (probability there is an object),
- $\hat{p}_c$: predicted probability for class c (after sigmoid or softmax).

A. YOLO foundational blocks

The YOLOv11 architecture is based on some foundational blocks designed to optimize feature extraction and object detection performance. These blocks are the following:

a) Convolution Block: A Convolution Block $\text{Conv}_{C_{out}\times C_{in}\times K_h\times K_w}$ consists of a convolutional layer and a Batch Normalization layer passed to the Sigmoid Linear Unit (SiLU) activation function (Fig.1).
A convolutional layer is a component of a convolutional neural network (CNN) that extracts features from images. It applies a set of learnable filters (also called kernels) to an input tensor to produce an output feature map. Let the input tensor $X \in \mathbb{R}^{C_{in}\times H_{in}\times W_{in}}$, the kernel weights $K \in \mathbb{R}^{C_{out}\times C_{in}\times K_h\times K_w}$, the bias vector be $b \in \mathbb{R}^{C_{out}}$ and the output tensor $Y \in \mathbb{R}^{C_{out}\times H_{out}\times W_{out}}$ where:

- C_{in} the number of input channels
- H_{in}, W_{in} the input height and width
- K_h, K_w the height and width of the kernel
- C_{out} the number of output channels
- H_{out}, W_{out} the output height and width depend on the stride and padding used

Each output value Y_{cij}, for output channel c, and spatial location (i, j) is computed as:

$$Y_{cij} = \sum_{c'=1}^{C_{in}}\sum_{u=1}^{K_h}\sum_{v=1}^{K_w} K_{c,c',u,v} \cdot X_{c',i+u-p_h,j+v-p_w} \; + b_c$$

where p_h, p_w are the vertical and horizontal padding.
A batch normalization layer independently normalizes a mini-batch of data across all observations for each channel.
After passing the input in a convolutional layer and a batch normalization layer, we then apply the SiLU activation function defined as:

$$SiLU(x) = x \cdot sigmoid(x) = \frac{x}{1+e^{-x}}$$

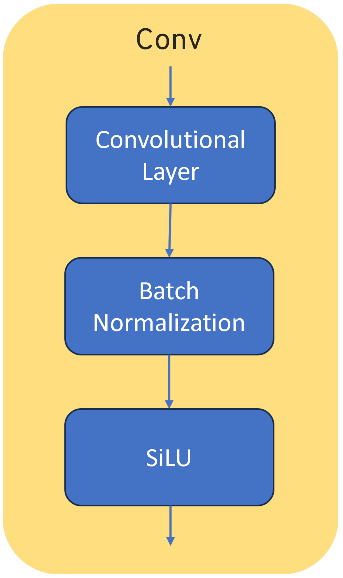


Fig. 1. Convolution Block

b) Bottleneck Block: The Bottleneck Block operates by compressing and then expanding the feature dimensions, facilitating deeper network architectures without a significant increase in parameters. Given an input feature map $X \in \mathbb{R}^{C\times H\times W}$, the Bottleneck module performs the following operations:

1) **Transformation and Compression**: a $\lfloor C_{out}\cdot e\rfloor \times C_{in}\times 3\times 3$ Convolution Block reduces the channel dimension with an expansion ratio $e \in [0, 1]$ (typically $e = 0.5$):
$$X_{\text{compressed}} = \text{Conv}_{\lfloor C_{out}\cdot e\rfloor\times C_{in}\times 3\times 3}(X)$$
2) **Transformation and Expansion**: a $C_{out}\times\lfloor C_{out}\cdot e\rfloor\times 3\times 3$ Convolution Block expands the channel dimension:
$$X_{\text{expanded}} = \text{Conv}_{C_{out}\times\lfloor C_{out}\cdot e\rfloor\times 3\times 3}(X_{\text{compressed}})$$
3) **Residual connection**: If $C_{in} = C_{out}$ and the Shortcut option is set to True, then the input is added to the output to facilitate gradient flow:
$$Y = X + X_{\text{expanded}}$$

The full block is represented in Fig.2.

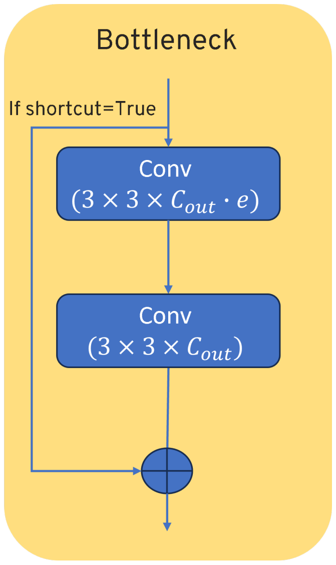

Bottleneck
If shortcut=True
Conv
$(3 \times 3 \times C_{out} \cdot e)$
Conv
$(3 \times 3 \times C_{out})$

Fig. 2. Bottleneck

c) C3K Block: The C3K Block splits the input into two parts, process the first part through a series of Bottleneck Blocks, then concatenates the output of the first part of the input with the second part of the input (Fig.3). Given an input feature map $X \in \mathbb{R}^{C\times H\times W}$, the C3K module performs the following operations:

1) **Splits the input**: two $\lfloor\frac{C_{out}}{2}\rfloor \times C_{in}\times 1\times 1$ Convolution Block splits the input:
$$X_1 = \text{Conv}_{\lfloor\frac{C_{out}}{2}\rfloor\times C_{in}\times 1\times 1}(X)$$
$$X_2 = \text{Conv}_{\lfloor\frac{C_{out}}{2}\rfloor\times C_{in}\times 1\times 1}(X)$$
2) **Process the first part**: Apply a series of n Bottleneck Blocks (typically $n = 1$) with an expansion ratio $e = 1$ to X_1:
$$X_{1,processed} = \text{Bottleneck} \circ \ldots \circ \text{Bottleneck}(X_1)$$

3) **Concatenation**: Concatenate $X_{1,\text{processed}}$ and X_2:

$$X_{\text{concat}} = [X_{1,\text{processed}}, X_2]$$

4) **Ensure output channel dimension**: Apply a $C_{out} \times 2 \cdot \lfloor \frac{C_{out}}{2} \rfloor \times 1 \times 1$ Convolution Block:

$$Y = \text{Conv}_{C_{out} \times 2 \cdot \lfloor \frac{C_{out}}{2} \rfloor \times 1 \times 1}(X_{\text{concat}})$$

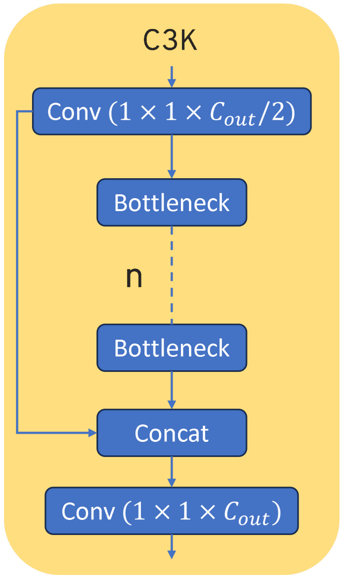

C3K
Conv ($1 \times 1 \times C_{out}/2$)
Bottleneck
n
Bottleneck
Concat
Conv ($1 \times 1 \times C_{out}$)

Fig. 3. C3K

d) C3K2 Block: : The C3K2 Block splits the input into two parts. It processes the first part through a series of C3K Blocks if the C3K option is set to True, a series of Bottleneck Blocks otherwise. The output of each layer in that series is concatenated at the end with the second part of the input (Fig.4). Given an input feature map $X \in \mathbb{R}^{C \times H \times W}$, the C3K2 module performs the following operations:

1) **Splits the input**: two $\lfloor C_{out} \cdot e \rfloor \times C_{in} \times 1 \times 1$ splits the input with an expansion ratio e (typically $e = 0.5$):

$$X_1 = \text{Conv}_{\lfloor C_{out} \cdot e \rfloor \times C_{in} \times 1 \times 1}(X),$$
$$X_2 = \text{Conv}_{\lfloor C_{out} \cdot e \rfloor \times C_{in} \times 1 \times 1}(X)$$

2) **Transformation**: Apply a series of n Blocks (typically $n = 1$). If C3K option is true, Block=C3K, otherwise Block=Bottleneck with an expansion ratio $e = 1$:

$$\begin{cases} X_{1,0} = X_1 \\ X_{1,k} = \text{Block}(X_{1,k-1}) \ \forall k \in \{1, ..., n\} \end{cases}$$

3) **Concatenate**: Concatenate all the layers' outputs and X_2:

$$X_{\text{concat}} = [X_1, X_{1,1}, ..., X_{1,n}, X_2]$$

4) **Ensure output channel dimension**: Apply a $C_{out} \times (n+2) \cdot \lfloor C_{out} \cdot e \rfloor \times 1 \times 1$ Convolution Block:

$$Y = \text{Conv}_{C_{out} \times (n+2) \cdot \lfloor C_{out} \cdot e \rfloor \times 1 \times 1}(X_{\text{concat}})$$

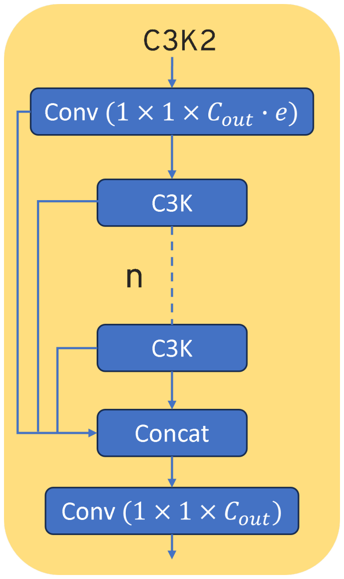

C3K2
Conv ($1 \times 1 \times C_{out} \cdot e$)
C3K
n
C3K
Concat
Conv ($1 \times 1 \times C_{out}$)

Fig. 4. C3K2

e) SPPF: Spatial Pyramid Pooling – Fast (SPPF)

The SPPF module aggregates multi–scale context efficiently by stacking a single MaxPool layer repeatedly (Fig.5), rather than using separate kernels of different sizes. Given an input feature map $X \in \mathbb{R}^{C_{in} \times H \times W}$ SPPF produces an output $O \in \mathbb{R}^{C_{out} \times H \times W}$ as follows:

1) **Channel reduction:**

$$C_{mid} = \left\lfloor \frac{C_{in}}{2} \right\rfloor$$

$$X_0 = \text{Conv}_{1 \times 1 \times C_{mid}}(X) \in \mathbb{R}^{C_{mid} \times H \times W}$$

2) **Repeated max-pooling:** Use a single max-pooling operator of kernel size k (by default $k = 5$), stride 1, padding $\lfloor k/2 \rfloor$, applied three times in series:

$$Y_1 = \text{MaxPool}_{k \times k}(X_0)$$
$$Y_2 = \text{MaxPool}_{k \times k}(Y_1)$$
$$Y_3 = \text{MaxPool}_{k \times k}(Y_2),$$

$\forall i = 1, 2, 3, Y_i \in \mathbb{R}^{C_{mid} \times H \times W}$.

3) **Concatenation:**

$$Z = \text{Concat}\big[X_0,\, Y_1,\, Y_2,\, Y_3\big] \ \in \ \mathbb{R}^{4\,C_{mid} \times H \times W}.$$

4) **Channel projection:**

$$O = \text{Conv}_{C_{out} \times 4\,C_{mid} \times 1 \times 1}(Z) \in \mathbb{R}^{C_{out} \times H \times W}.$$

Remarks:

- By reusing a single $k \times k$ MaxPool in series, SPPF effectively captures receptive fields of approximate sizes 5×5, 9×9, and 13×13 without extra parameters.
- The final concatenation preserves both local detail (X_0) and increasingly broader context (Y_1, Y_2, Y_3), improving detection of small and large objects alike.

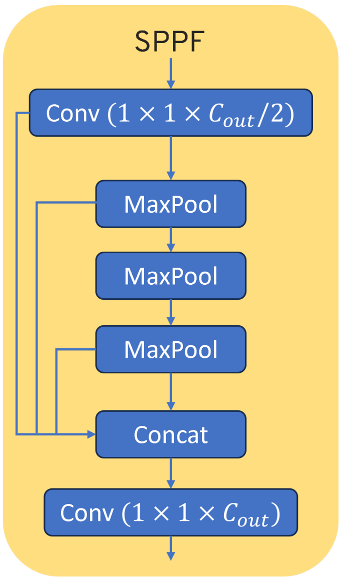

SPPF
Conv $(1 \times 1 \times C_{out}/2)$
MaxPool
MaxPool
MaxPool
Concat
Conv $(1 \times 1 \times C_{out})$

Fig. 5. SPPF

f) Shared Channel Attention: The Shared Channel Attention mechanism captures long-range dependencies across spatial positions using multi-head self-attention. Given an input feature map

$$X \in \mathbb{R}^{C_{in} \times H \times W}, \quad N = H \cdot W,$$

and h heads, each of dimension $d_h = \lfloor C_{out}/h \rfloor$ $(C_{out} = C_{in})$ with key-query dimension d_k, the module does:

1) **Project to Q, K, V:**

$$\mathrm{QKV}_{\text{full}} = \mathrm{Conv}_{(2\,d_k+d_h)\,h \times C_{in} \times 1 \times 1}(X)$$

$$\mathrm{QKV}_{\text{full}} \in \mathbb{R}^{\left[(2\,d_k+d_h)\,h\right] \times H \times W}$$

2) **Reshape and split heads:**
Reshape $\mathrm{QKV}_{\text{full}}$ to $\mathrm{QKV} \in \mathbb{R}^{h \times (2\,d_k+d_h) \times N}$

$$\forall i \in [\![1,h]\!], Q_i = (QKV_{i,j,p})_{\substack{1 \le j \le d_k \\ 1 \le p \le N}}$$

$$K_i = (QKV_{i,j,p})_{\substack{d_k+1 \le j \le 2d_k \\ 1 \le p \le N}}$$

$$V_i = (QKV_{i,j,p})_{\substack{2d_k+1 \le j \le d_h \\ 1 \le p \le N}}$$

3) **Scaled dot-product per head:**

$$\forall i \in [\![1,h]\!], A_i = \mathrm{softmax}\left(\frac{Q_i^T K_i}{\sqrt{d_k}}\right) \in \mathbb{R}^{N \times N}$$

4) **Apply attention to V:**

$$\forall i \in [\![1,h]\!], O_i = V_i\, A_i^{\top} \in \mathbb{R}^{d_h \times N}$$

5) **Concatenate heads:**

$$O_{\text{concat}} = [\,O_1; \ldots; O_h\,] \in \mathbb{R}^{hd_h \times N}$$

6) **Reshape and project out:**
Reshape O to $O_{\text{reshaped}} \in \mathbb{R}^{hd_h \times H \times W}$

$$\mathrm{Output} = \mathrm{Conv}_{C_{out} \times hd_h \times 1 \times 1}(O_{\text{reshaped}})$$

The attention mechanism is represented in Fig.6.

g) PSABlock: PSABlock applies shared channel attention and a feedforward network with residual connections (Fig.7).
For $X \in \mathbb{R}^{C_{in} \times H \times W}$:

1) **Apply Attention block:** $A = \mathrm{Attention}(X)$
2) **Transformation:** $X' = X + A$
3) **Apply Feedforward network:** $F = \mathrm{FFN}(X')$
where $\mathrm{FFN} = \mathrm{Conv}_{2C_{in} \times C_{in} \times 1 \times 1} \circ \mathrm{Conv}_{C_{out} \times 2C_{in} \times 1 \times 1}$ and $C_{out} = C_{in}$
4) **Transformation:** $Y = X' + F$

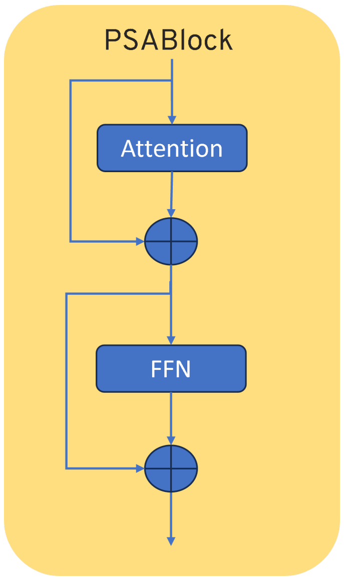


Fig. 7. PSABlock

h) C2PSA Block: C2PSA compresses channels, stacks n PSABlocks on the second half, then restores channels (Fig.8).
For $X \in \mathbb{R}^{C_{\text{in}} \times H \times W}$, $C_{mid} = \lfloor C_{\text{in}}\, e \rfloor$:

1) **Transformation and Compression:**
$I = \mathrm{Conv}_{2C_{mid} \times C_{in} 1 \times 1}(X)$
$I = [I_1; I_2]$ with $I_1, I_2 \in \mathbb{R}^{C_{mid} \times H \times W}$
2) **Repeated PSABlock:**
$B_0 = I_2, \quad B_k = \mathrm{PSABlock}(B_{k-1}),\ k = 1 \ldots n$
3) **Concatenate:**
$O = [\,I_1;\, B_n\,] \in \mathbb{R}^{2C_{mid} \times H \times W}$
4) **Channel Projection:** $Y = \mathrm{Conv}_{C_{\text{out}} \times 2C_{mid} \times 1 \times 1}(O)$

B. Architectural Overview

The architecture is composed of three main components: the backbone, the neck, and the head. The backbone is responsible for extracting low-level to high-level visual features from the input image using a series of convolutional layers. These features are then passed to the neck, which aggregates and enhances them, often by fusing information from multiple scales to improve the representation of objects of different sizes. Finally, the head takes the aggregated features and performs the detection task, predicting object classes and bounding box coordinates based on the refined feature maps. The full architecture can be found in Fig 18.

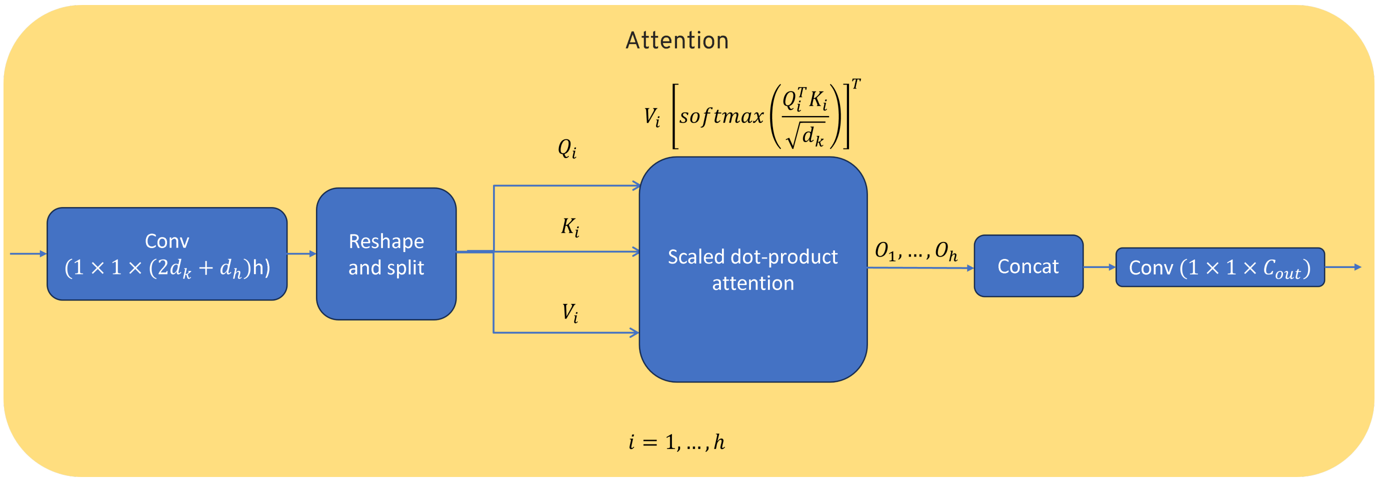

Attention
$V_i\left[softmax\left(\frac{Q_i^T K_i}{\sqrt{d_k}}\right)\right]^T$
Q_i
K_i
V_i
Conv $(1\times1\times(2d_k+d_h)\text{h})$
Reshape and split
Scaled dot-product attention
$O_1,\dots,O_h$
Concat
Conv $(1\times1\times C_{out})$
$i=1,\dots,h$

Fig. 6. Attention Mechanism

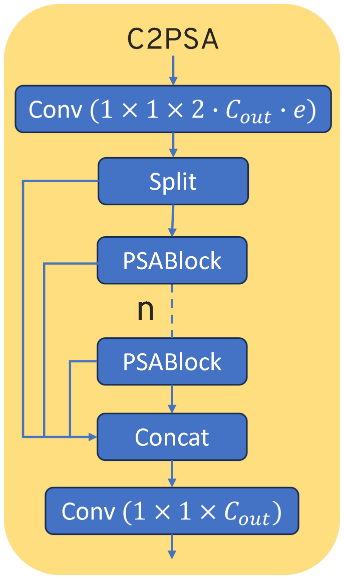

C2PSA
Conv $(1\times1\times2\cdot C_{out}\cdot e)$
Split
PSABlock
n
PSABlock
Concat
Conv $(1\times1\times C_{out})$

Fig. 8. C2PSA

C. Loss function

YOLOv11 employs a composite loss function that combines bounding box regression loss, objectness loss, and classification loss. Each component is designed to address specific aspects of the object detection task.

1) Bounding Box Regression Loss: YOLOv11 utilizes the Extended Intersection over Union (EIoU) loss to measure the discrepancy between predicted and ground truth bounding boxes. The EIoU loss considers the overlap area, center distance, and aspect ratio differences:

$$\mathcal{L}_{\text{EIoU}} = 1 - \text{IoU} + \frac{\|(x,y)-(x^{gt},y^{gt})\|_2^2}{c^2} + \alpha\cdot\frac{(w-w^{gt})^2}{w^{gt}} + \beta\cdot\frac{(h-h^{gt})^2}{h^{gt}}$$

where:

- IoU is the Intersection over Union between the predicted box $\mathbf{b}$ and groud truth $\mathbf{b}^{gt}$.
- c is the diagonal length of the smallest enclosing box covering both $\mathbf{b}$ and $\mathbf{b}^{gt}$.
- x,y,w,h and $x^{gt},y^{gt},w^{gt},h^{gt}$ are respectively the x_center, y_center, widths and heights of the predicted and ground truth boxes.
- α,β are weighting factors for the aspect ratio terms.

This loss function penalizes both localization errors and mismatches in aspect ratios, leading to more accurate bounding box predictions.

2) Objectness Loss: The objectness loss measures the confidence of the model in detecting an object within a bounding box. It is computed using Binary Cross-Entropy (BCE) loss:

$$\mathcal{L}_{obj} = -\left[y\cdot\log(\hat{y}) + (1-y)\cdot\log(1-\hat{y})\right]$$

where:

- $y\in\{0,1\}$ indicates the presence (1) or absence (0) of an object.
- $\hat{y}\in[0,1]$ is the predicted objectness score.

This loss encourages the model to assign high confidence scores to bounding boxes containing objects and low scores to background regions.

3) Classification Loss: For multi-class object detection, YOLOv11 employs BCE loss for each class to handle multi-label scenarios:

$$\mathcal{L}_{\text{cls}} = -\sum_{k=1}^{K}\left[y_k\cdot\log(\hat{y}_k) + (1-y_k)\cdot\log(1-\hat{y}_k)\right]$$

where:

- K is the total number of classes.
- $y_k\in\{0,1\}$ indicates the presence (1) or absence (0) of class k
- $\hat{y}_k\in[0,1]$ is the predicted probability for class k.

This formulation allows the model to predict multiple classes for a single object, accommodating complex scenarios where objects may belong to multiple categories.

4) Total loss: The overall loss function is a weighted sum of the individual loss components:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{box}} \cdot \mathcal{L}_{\text{EIoU}} + \lambda_{\text{obj}} \cdot \mathcal{L}_{obj} + \lambda_{\text{cls}} \cdot \mathcal{L}_{\text{cls}}$$

where $\lambda_{\text{box}}, \lambda_{\text{obj}}, \lambda_{\text{cls}}$are hyperparameters that balance the contributions of each loss component.

IV. Object classification with Optimal Transport (OT)

Optimal transport is a well-developed mathematical theory that defines robust metrics between probability distributions. The computation of optimal displacements between densities through the associated transport map makes this theory mainstream in several applicative fields [11]. For image processing applications, the transport map can for instance be used to compute geodesics between images or to transfer characteristics of one image to another.

A. Mathematical Formulation of Optimal Transport (OT)

1) Monge's original formulation: The goal of Optimal Transport is to transform a measure on $\mathbb{R}^d$ into another with minimal cost. Monge's original formulation of the Optimal Transport problem aims at minimizing the cost for transporting a distribution μ_s onto another distribution μ_t using a map T that represents the transport:

$$\min_{\substack{T \\ \mu_t = T.\mu_s}} \int_{\mathbb{R}^d} c(x, T(x))\mu_s(dx)$$

Here, μ_s, μ_t are measures on $\mathbb{R}^d$, $P : \mathbb{R}^d \to \mathbb{R}^d$
and $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ are measurable functions that represent respectively the transport map and the cost map.

2) Kantorovitch Formulation : In the discrete finite case (*i.e.* both measures have finite supports X^s and X^t), the Kantorovich formulation of Optimal Transport provides a way to compare two histograms μ_s and μ_t, representing the features of two images for example. It can be interpreted as finding the most efficient way to move "mass" from one distribution to another.

In this case:

$$\mu_s = \sum_{i=1}^{n_s} a_i \delta_{x_i^s}$$
$$\mu_t = \sum_{i=1}^{n_t} b_i \delta_{x_i^t}$$

where:

- $X^s = \{x_i^s \ ; \ i \in [\![1, n_s]\!]\}$ and $X^t = \{x_i^t \ ; \ i \in [\![1, n_t]\!]\}$ are point-clouds in a feature space $\mathbb{R}^d$ (for instance position, color, gabor filter coefficients, etc)
- $a \in \mathbb{R}_+^{n_s}$ and $b \in \mathbb{R}_+^{n_t}$ are the non-negative weight vectors associated with each distribution

We can apply this to $a \in \mathbb{R}^{n_s}, b \in \mathbb{R}^{n_t}$ by adding the same constant to all of their coefficients to make them non-negative beforehand.
Without loss of generality, we suppose that μ_s and μ_t are probability distributions. Hence $a \in \Delta^{n_s}, b \in \Delta^{n_t}$.
In this setting, the cost function c is replaced by a cost matrix $C \in \mathbb{R}^{n_s \times n_t}$ *s.t.* $\forall i \in [\![1, n_s]\!], \forall j \in [\![1, n_t]\!], c_{i,j}$ represents the cost of transporting x_i^s to x_j^t and can be chosen as a distance between those points.
The transport map T is replaced by a transport matrix $P \in \Pi_(\mu_s, \mu_t)$. $\forall i \in [\![1, n_s]\!], \forall j \in [\![1, n_t]\!], p_{i,j}$ can be interpreted as the proportion of the mass transferred from x_i^s to x_j^t [11]. $\Pi_(\mu_s, \mu_t)$ is the set of admissible transport matrices :

$$\Pi(\mu_s, \mu_t) = \left\{P \in \mathbb{R}_+^{n_s \times n_t} | P\mathbf{1}_{n_t} = a, P^T\mathbf{1}_{n_s} = b\right\}$$

In the case of probability distributions ($\langle a, \mathbf{1}_{n_s} \rangle = \langle b, \mathbf{1}_{n_t} \rangle = 1$), $\Pi(\mu_s, \mu_t)$ represents the set of joint distributions whose marginals are μ_s and μ_t.

Given a fixed cost matrix C, the Monge-Kantorovitch formulation of the Optimal Transport problem estimates the Optimal Transport plan P^* between distributions μ_s and μ_t:

$$d_C(\mu_s, \mu_t) = \min_{P \in \Pi(\mu_s, \mu_t)} \langle P, C \rangle$$

This is a linear programming problem that can be solved with various methods, namely the simplex algorithm.

3) Entropic Regularization and Entropic Constraints: Although the optimal transport formulated by Kantorovich is solvable, it poses several practical limitations.
It is computationally expensive, especially for large-scale problems, with the simplex algorithm exhibiting a complexity of $O(n^3 log(n))$ when $n_s = n_t = n$.
Moreover, it often produces sparse transport plans (many zero entries).

To address these issues, [12] proposed an entropic regularization of the OT problem. In fact, [12] suggests that the transport plan must have sufficient entropy in order to have smooth transport plan instead of extreme ones (sparse transport plans). Let's first recall an information-theoretic inequality of joint entropy:

$$h(P) \leq h(\mu_s) + h(\mu_t)$$

with equality when $P = ab^T$ meaning that P represent the independent joint probability of μ_s and μ_t. We introduce the relaxed set $\Pi_\alpha(\mu_s, \mu_t) \subset \Pi(\mu_s, \mu_t)$ as the following:

$$\Pi_\alpha(\mu_s, \mu_t) = \{P \in \Pi(\mu_s, \mu_t) \mid h(P) \geq h(\mu_s) + h(\mu_t) - \alpha\}$$

Or in terms of Kullback-Leibler divergence:

$$\Pi_\alpha(\mu_s, \mu_t) = \left\{P \in \Pi(\mu_s, \mu_t) \mid KL(P \| ab^T) \leq \alpha\right\}$$

We can easily verify that $KL(P\|ab^T) = h(\mu_s) + h(\mu_t) - h(P)$. This quantity is also the mutual information $I(X;Y)$ of two random variables (X, Y) should they follow the joint probability distribution P. The regularized optimal transport becomes:

$$d_{C,\alpha}(\mu_s, \mu_t) = \min_{P \in \Pi_\alpha(\mu_s, \mu_t)} \langle P, C \rangle$$

According to Lagrange duality, for every $\alpha \in [0, +\infty[$ there exists a $\lambda > 0$ such that:

$$d_{C,\alpha}(\mu_s, \mu_t) = d_C^\lambda(\mu_s, \mu_t) = \langle P^\lambda, C \rangle$$

where P^λ is a solution of the following optimization problem:

$$\underset{P \in \Pi(\mu_s, \mu_t)}{\text{minimize}} \langle P, C \rangle - \frac{1}{\lambda} h(P) \qquad (1)$$

This solution is unique by strict convexity of the objective function. Thus, we can take λ as our hyperparameter instead of α. The regularization parameter $\lambda > 0$ controls the smoothness of the transport plan.

4) Computing Regularized Transport with Sinkhorn's Algorithm: This formulation admits an efficient solution via the **Sinkhorn–Knopp algorithm**, which performs iterative matrix scaling. It solves the entropic regularized OT problem efficiently by exploiting the **Sinkhorn theorem**. Let's see this in detail. Let's write the Lagrangian of the problem 1:

$$\begin{aligned}\mathcal{L}(P, \nu, \omega) &= \langle \nu, P\mathbf{1}_{n_s} - a \rangle + \langle \omega, P\mathbf{1}_{n_t} - b \rangle + h(P) + \langle P, C \rangle \\ &= \langle \nu, P\mathbf{1}_{n_s} - a \rangle + \langle \omega, P\mathbf{1}_{n_t} - b \rangle \\ &\quad + \sum_{i,j} \frac{1}{\lambda} p_{i,j} \log p_{i,j} + p_{i,j} c_{i,j}\end{aligned}$$

$$\text{Thus, } \forall i, j, \frac{\partial \mathcal{L}}{\partial p_{i,j}^\lambda} = 0 \implies p_{i,j}^\lambda = e^{-\frac{1}{2} - \lambda \nu_i} e^{-\lambda c_{i,j}} e^{-\frac{1}{2} - \lambda \omega_j}$$

Using the Sinkhorn theorem, P^λ is the only matrix with row-sum a and column sum b of the form:

$$P^\lambda = \text{diag}(u) e^{-\lambda C} \text{diag}(v); u \in \mathbb{R}^{n_s}, v \in \mathbb{R}^{n_t}, u, v > 0$$

It is thus possible to recover P^λ by simply running the **Sinkhorn-Knopp** algorithm on the **Gibbs kernel** $K = \exp(-\lambda C)$ in order to find (u, v).

Algorithm 1 Sinkhorn Algorithm

```
Require: $C \in \mathbb{R}^{n_s \times n_t}$, $\lambda > 0$, $a \in \mathbb{R}^{n_s}$, $b \in \mathbb{R}^{n_t}$, $\varepsilon > 0$, max_iter
Ensure: $P \in \mathbb{R}^{n_s \times n_t}$, $\mathrm{d}_C^\lambda(a, b)$
  $K \leftarrow \exp(-\lambda \cdot C)$
  $u \leftarrow$ ones$(n_s)$
  for $t = 1$ to max_iter do
      $v \leftarrow b./(K^T u)$
      $u \leftarrow a./(Kv)$
      if $\|K^T u \cdot v - b\|_1 < \varepsilon$ and $\|Kv \cdot u - a\|_1 < \varepsilon$ then
          break
  $P \leftarrow \text{diag}(u) K \text{diag}(v)$
  $\mathrm{d}_C^\lambda(a, b) \leftarrow \langle P, C \rangle$
  return $P$, $\mathrm{d}_C^\lambda(a, b)$
```

Explanation of the Sinkhorn Algorithm:

1. *Initialization of matrix K*: Compute the Gibbs kernel $K = \exp(-\lambda C)$, which transforms the cost matrix C into positive weights.

2. *Initialization of scaling vector u*: Initialize u as a vector of ones matching the size of the source distribution a. This vector helps adjust the mass distribution iteratively.

3. *Iterative updates*: For a maximum number of iterations or until convergence, update vectors v and u alternatively to enforce the marginal constraints with respect to the target b and source a respectively. This corresponds to Sinkhorn-Knopp matrix scaling iterations.

4. *Stopping criterion*: Check if the approximated marginals $K^T u \cdot v$ and $Kv \cdot u$ are sufficiently close to b and a respectively, using the L_1 norm and a tolerance ε. Stop if converged.

5. *Computation of transport plan P*: Once converged, compute the regularized transport plan $P = \text{diag}(u)\, K \,\text{diag}(v)$, which approximately satisfies the marginal constraints.

6. *Regularized transport cost*: Finally, compute the Sinkhorn distance as the sum of the element-wise product between P and C, i.e.

$$\mathrm{d}_C^\lambda(a, b) = \langle P, C \rangle$$

In summary, the Sinkhorn algorithm computes an entropy-regularized optimal transport plan by iteratively scaling rows and columns of the kernel matrix to fit the source and target marginals, resulting in a smooth and computationally efficient approximation of the classical OT problem.

B. Application in Our Framework

After detecting the bounding boxes of objects present in the images using YOLOv11, and extracting feature descriptors via a ResNet50 network, we apply a regularized optimal transport framework to assign each object to its corresponding class.

In this approach, the source distribution is modeled by the characteristic representation of the object to be classified, while the target distribution corresponds to a statistical prototype representative of the class (e.g. human, vehicle, engine, etc.).

The optimal transport problem enables the determination of a transport plan that minimizes the cost between these two distributions. The Sinkhorn algorithm, through its entropic regularization, facilitates the efficient computation of this regularized transport distance. This distance is then used as a discriminative criterion for the final assignment of the class label to the observed object.

V. Metrics

A. Confusion matrix

A *confusion matrix* is a fundamental tool used to evaluate the performance of a classification model. It provides a detailed summary of prediction results, including the number of:

- **True Positives (TP)**: correctly predicted positive cases,
- **True Negatives (TN)**: correctly predicted negative cases,
- **False Positives (FP)**: negative cases incorrectly classified as positive,
- **False Negatives (FN)**: positive cases incorrectly classified as negative.

It is presented as a square matrix where each column represents the actual class and each row represents the predicted class. It is particularly useful for identifying the types of errors the model is making.

B. Accuracy, Precision, Recall, F1-score

These metrics provide quantitative insights into the performance of classification models.

- **Accuracy** measures the proportion of correct predictions among all predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision** measures the proportion of positive predictions that are actually correct:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall** measures the proportion of actual positives that were correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-score** is the harmonic mean of precision and recall:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

C. mean Average Precision : mAP50

The *mean Average Precision* (mAP) is commonly used in object detection tasks. It takes into account both the classification accuracy and the spatial accuracy of the predicted bounding boxes.

A detection is considered correct if the **Intersection over Union (IoU)** between the predicted bounding box **b** and the ground truth $\mathbf{b}^{gt}$ exceeds a certain threshold (0.50 in the case of mAP50):

$$\text{IoU} = \frac{\text{Area}(\mathbf{b} \cap \mathbf{b}^{gt})}{\text{Area}(\mathbf{b} \cup \mathbf{b}^{gt})}$$

The Average Precision (AP) is computed as the area under the precision-recall curve for each class, and the mean Average Precision is calculated as:

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^{C} AP_c$$

where C is the number of object classes.

VI. Experimental results

A. Dataset description

The VisDrone2019 dataset is collected by the AISKYEYE team at Lab of Machine Learning and Data Mining , Tianjin University, China. The benchmark dataset consists of 288 video clips formed by 261,908 frames and 10,209 static images, captured by various drone-mounted cameras, covering a wide range of aspects including location (taken from 14 different cities separated by thousands of kilometers in China), environment (urban and country), objects (pedestrian, vehicles, bicycles, etc.), and density (sparse and crowded scenes). Note that, the dataset was collected using various drone platforms (i.e. drones with different models), in different scenarios, and under various weather and lighting conditions. These frames are manually annotated with more than 2.6 million bounding boxes of targets of frequent interests, such as pedestrians, cars, bicycles, and tricycles. Some important attributes including scene visibility, object class and occlusion, are also provided for better data utilization.

We used the training and validation parts of the static image data set totaling 7019 pictures and merged the classes as follows:

- pedestrian + people $\rightarrow$ people
- bicycle + tricycle + awning-tricycle $\rightarrow$ bike
- car + van $\rightarrow$ vehicle
- truck $\rightarrow$ truck
- bus $\rightarrow$ bus
- motor $\rightarrow$ motor

B. Baseline: YOLOv11

1) Methodology: We use YOLOv11 as a baseline. We use the YOLOv11m model which has 20.1 million parameters. We finetuned the pre-trained model on our dataset with 100 epochs.

2) Evaluation: YOLOv11 obtained a mAP50 of 60.1% (Fig.9) which outperforms other variants of YOLOv11 on the VisDrone dataset. This can be explained by the class merging, since we merged objects that look alike. According to the confusion matrix (Fig.10), YOLO failed to detect 54% of people, 52% of bikes, 21% of vehicles, 25% of trucks, 19% of buses, and 51% of motorcycles, showing the difficult task of detecting objects in aerial images that is due to many factors, as stated previously. Some of which are the lack of luminosity in certain images (images taken by night), or also the small size of the object captured by the camera as the object can be very far away. However, among the detected object, the classification is quite powerful. In fact, the model correctly classified all the vehicles that it detected and almost all the people that it detected.

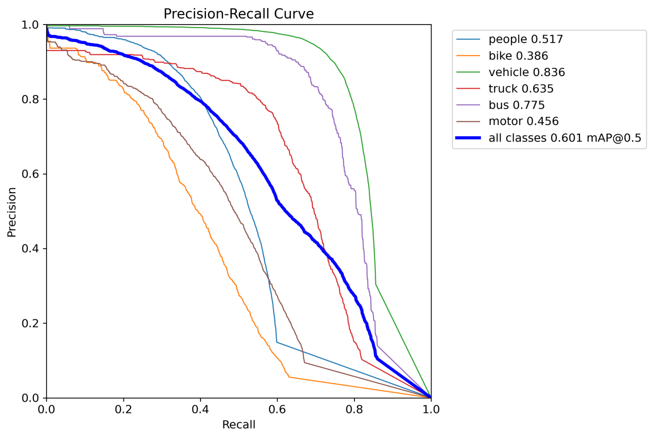


Fig. 9. Precision-Recall Curve with only YOLOv11

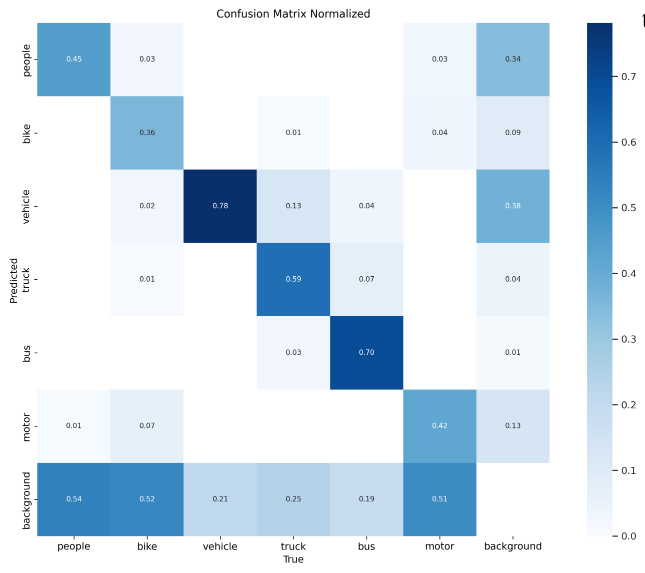


Fig. 10. Normalized Confusion Matrix with only YOLO

C. *Detection by YOLO*

1) Methodology: In order to improve the detection performance of YOLO, we merged all the classes together so that we have only one class: ”Object”.

2) Evaluation: Merging all the classes together impoved the mAP by 16.8 % obtaining an mAP of 70.2 %. (Fig.11)

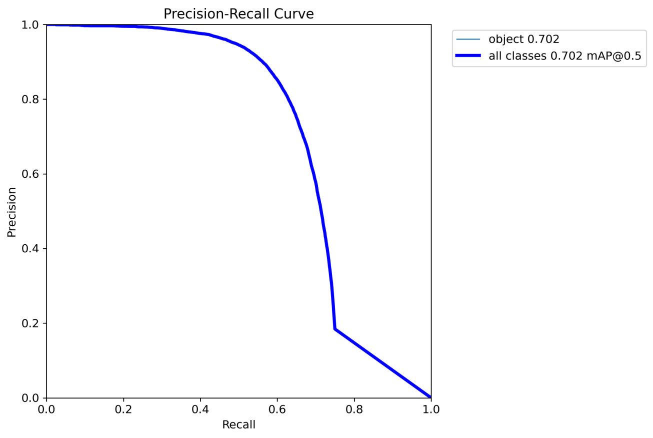


Fig. 11. Precision-Recall curve for YOLO with 1 class

D. *Classification by Optimal Transport*

1) Methodology : The classification pipeline consists of the following main steps:

- **Data Acquisition :**
  Object images are collected from the detection outputs of the YOLOv11 model. Each detected object is extracted as an individual image sample.
- **Image Transformation :**
  The object images undergo standard preprocessing such as resizing and normalization to fit the input requirements of the feature extractor.
- **Feature Extraction :**
  Deep features are extracted from each preprocessed image using the ResNet50 convolutional neural network pretrained on ImageNet. The output of the final average pooling layer is used as a fixed-length feature vector.
- **Prototype Creation :**
  For each object class, we sample 100 random images, extract their features with ResNet50. The images will be considered as class prototypes.
- **Optimal Transport Classification :**
  Classification is performed by computing the Sinkhorn distance between the features of a test sample and each class prototype. For each class, we take the mean distance between the input and the prototypes. The sample is assigned to the class with the minimum mean distance.
- **Validation and Testing**
  The model is evaluated on a separate test set. Performance is assessed through metrics including accuracy, precision, recall, F1-score, and by analyzing the confusion matrix.

2) Evaluation:

- ***Confusion Matrix*** (Figure 12)

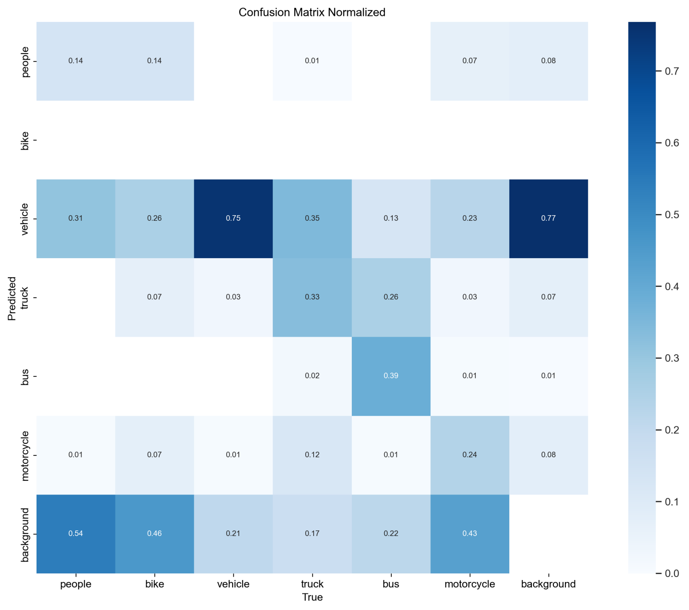


Fig. 12. Confusion matrix of the OTClassifier

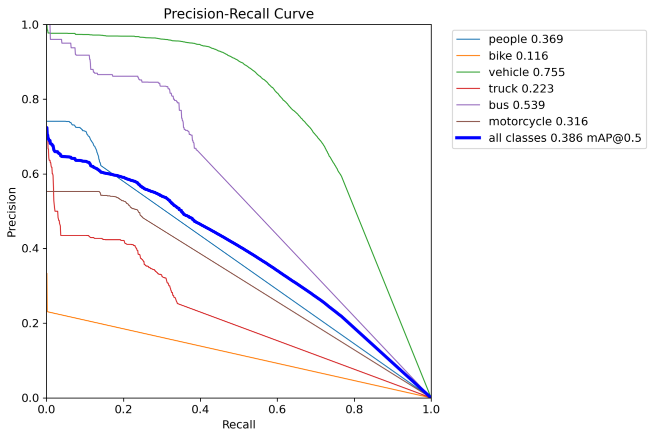


Fig. 13. PR curve of the OT Classifier

Overall, using YOLO to detect any "object" rather than specific categories did slightly improve recall for most categorie:fewer bikes, trucks, and motorcycles were prematurely discarded as background, while detection for people and generic vehicles remained roughly the same; buses, however, saw a small drop in detection recall. Once an object crop reaches the classifier, though, its ability to assign the correct label is weak across nearly all true classes except "vehicle": true bikes are almost never recognized as bikes (they are overwhelmingly labeled as background or confused with vehicles), true motorcycles are more often called vehicles or background than motorcycles, and both trucks and buses bleed heavily into each other and into the broader "vehicle" category (only around one third of true trucks become "truck" and fewer than 40 % of true buses remain "bus"). True people are also largely lost: only about 14 % are correctly classified as "people," with the rest being lumped into "vehicle" or discarded as background. In short, although detection slightly gains at the first stage, the classifier's fine-grained distinctions collapse, leaving only "vehicle" with moderate recall. Every other object class is so frequently mis-labeled that the two-stage pipeline underperforms a YOLO-only setup.

- ***mAP50*** (Figure 13)
  The precision–recall curves make clear that "vehicle" is by far the strongest class: its curve stays above 0.9 precision until nearly 50 % recall and only begins to drop off later, yielding an AP of about 0.755, whereas every other class has a much lower area under the curve. "Bus" is the second best with an AP around 0.539, but its precision steadily falls off as recall climbs, indicating many false positives at higher recall thresholds. "People" $(AP \simeq 0.369)$ and "motorcycle" $(AP \simeq 0.316)$ show moderately poor trade-offs between precision and recall, while "truck" $(AP \simeq 0.223)$ and especially "bike" $(AP \simeq 0.116)$ perform very poorly, losing precision almost immediately as recall increases. Overall, the thick blue mAP curve (0.386 at IoU = 0.5) underlines that, aside from "vehicle," the classifier struggles to maintain acceptable precision when it tries to catch more true positives.

E. Classification by Optimal Transport with k-NN

1) Methodology: This classification method of detected objects is similar to the previous one. It differs only in the definition of the prototypes and the computation of the class assigned to the input. In order to improve the consistency with the intra-class variance of the dataset, the classification algorithm compares the input feature vector to various prototype feature vectors of each class, while the precedent method computes the Sinkhorn distance between the input vector and a barycenter vector of the class. Indeed, an entire class of objects can hardly be modeled by a unique prototype.

In this version of the method, we randomly choose 100 vectors of each class as prototypes of the classes. Then, we apply a k-nearest neighbors algorithm, with the Sinkhorn distance, to find the 10 prototypes nearest to the input vector. The input is assigned to the class which is most represented in those 10 prototypes.

2) Evaluation:

- ***Confusion Matrix*** (Figure 14)
  With the K-NN classifier in place, the once-dominant "vehicle" category still remains the best recognized (about 58 % of true vehicles are correctly labeled), but now all other object classes share more comparable recall and confusion rates rather than being overwhelmed by "vehicle" or by "background." For instance, "people" are correctly identified around 23 % of the time (instead of only 14 % previously), "bike" shows a modest 14 % recall, and "truck" and "bus" both land near the low-20 % to upper-30 % range (23 % and 37 %, respectively). "Motorcycle" now also reaches about 21 % recall. Although

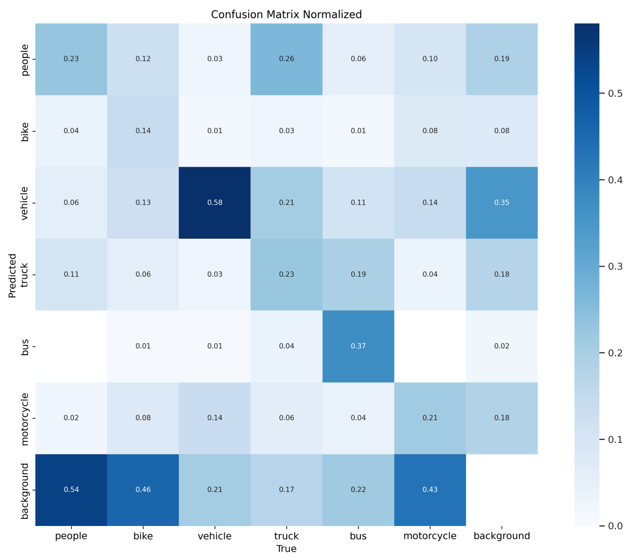


Fig. 14. Confusion matrix of the OT Classifier with K-NN

no single class matches the high precision we saw for vehicles under the old classifier, the softer, more uniform diagonals indicate that the K-NN model spreads its errors relatively evenly. Trucks and buses still bleed into each other (truck $\rightarrow$ bus $\simeq$ 19%, bus $\rightarrow$ truck $\simeq$ 4%), motorcycles get confused with vehicles or bikes about as often as they're correctly labeled, and bikes have scattered confusions into people, vehicles, and motorcycles. In short, K-NN sacrifices peak accuracy on "vehicle" to produce a more balanced distribution of recall and confusion across all object classes rather than allowing one category to dominate and the rest to collapse.

- ***mAP@50*** (Figure 15)

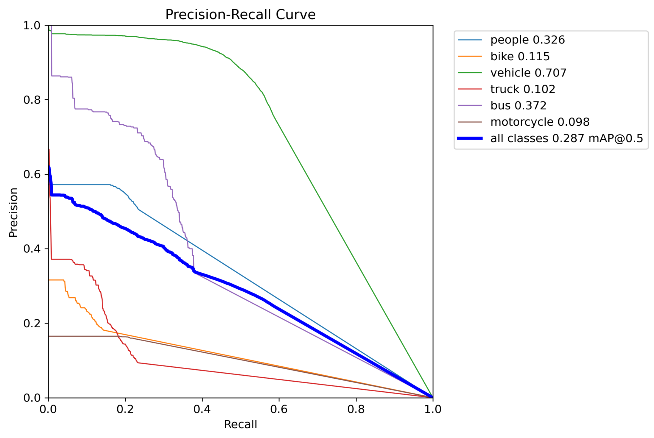


Fig. 15. PR Curve of the OT Classifier with K-NN

The updated precision-recall (PR) curve reflects the more balanced classification introduced by the K-NN model. While the overall mAP@0.5 dropped slightly to 0.287, the curves for previously underperforming classes, like "bike," "truck," and "motorcycle", now show more substantial, albeit still modest, precision over broader recall ranges, confirming improved consistency. The "vehicle" class still dominates with a strong 0.707 AP, but its curve is no longer as steeply isolated as before, indicating that the classifier isn't disproportionately favoring it. Most importantly, the uniformity in the PR curves across classes highlights that, despite lower peak performance, the classifier now distributes its attention more equitably among all classes.

F. Classification by OT, k-NN and k-means for the prototypes

1) Methodology: Here, we try to enhance the previous model. The random choice of the prototypes may not be representative of the dataset and may hinder the model's ability to generalize. Our goal is thus to obtain prototypes that are "far" enough from each other. We substituted it by computing 100 centroids for each class by executing the k-means++ algorithm. Those 100 centroids are used as prototypes.

2) Evaluation:

- ***Confusion Matrix*** (Figure 16)

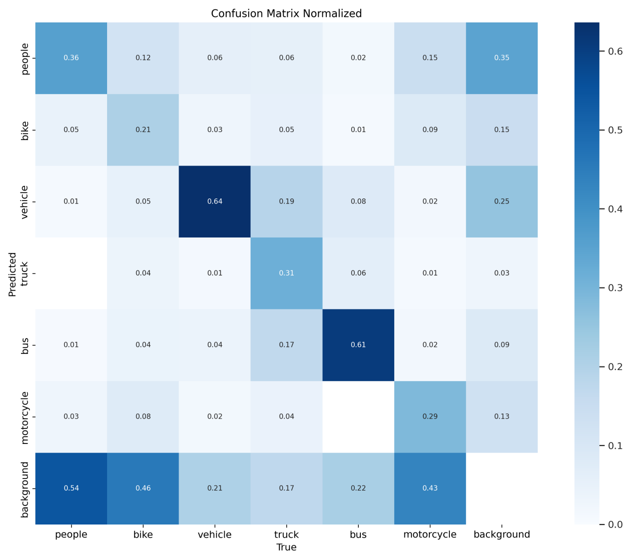


Fig. 16. Confusion matrix of the OT Classifier with K-NN (prototypes computed with k-means)

Using K-means to choose 100 centroids per class has boosted every class's recall (diagonal value) compared to the random-selection K-NN. "People" now reaches about 36 % correct (up from $\sim$ 23 %), "bike" improves to 21 % (vs. $\sim$ 14 %), and "vehicle" jumps to 64 % (vs. $\sim$ 58 %). "Truck" rises to 31 % (vs. $\sim$ 23 %), "bus" to 61 % (vs. $\sim$ 37 %), and "motorcycle" to 29 % (vs. $\sim$ 21 %). Not only are the diagonals stronger, but off-diagonal confusions have decreased: for example, fewer bikes are misclassified as people or vehicles, and trucks and buses bleed into one another less often. In short, K-means centroids yield more representative prototypes that produce higher true-positive rates across all detected object classes.

- ***mAP@50*** (Figure 17)

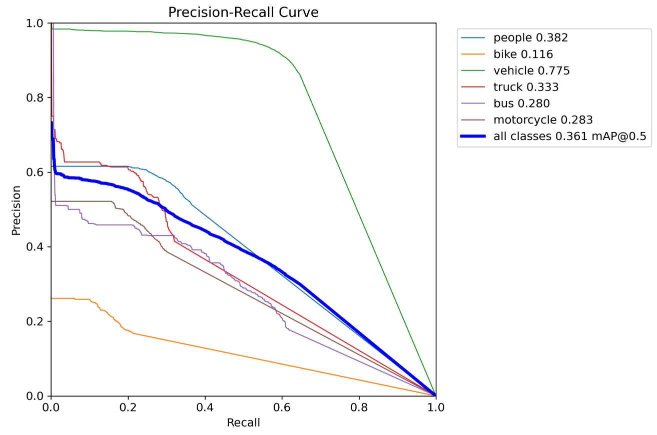


Fig. 17. PR Curve of the OT Classifier with K-NN (prototypes computed with K-means)

The Precision–Recall curves for the K-means–based K-NN show clear gains in almost every category. "Vehicle" remains the strongest, with its curve pushed even higher to an AP of 0.775, and "People" also climbs to 0.382, reflecting higher precision across nearly all recall thresholds compared to the random-prototype version. The "truck" curve is especially steeper, lifting its AP to 0.333 from just over 0.10 and indicating far fewer missed trucks at moderate confidence levels. "Motorcycle" likewise rises to 0.283, displaying improved precision over a broader recall range. "Bike" remains relatively flat $(AP \sim 0.116)$, suggesting only marginal change, while "bus" dips to 0.280; its precision at higher recalls is lower than before, indicating some trade-off. Overall, the thicker blue mAP curve sits notably above its predecessor across mid-range recalls, yielding a jump in mean AP from 0.287 to 0.361 and demonstrating that K-means centroids produce consistently better classification trade-offs for most object classes.

G. *Comparative table of the models*

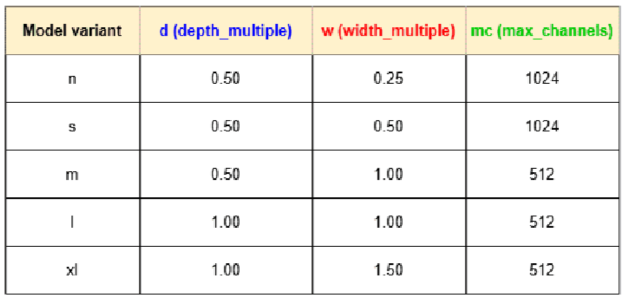
Table I summarizes and compares the perfomance of the baseline YOLOv11 model and the OT-based models we used. Overall, YOLOv11 clearly outperforms every OT-based approach, achieving the highest accuracy (0.92), precision (0.69), recall (0.55), F1-score (0.61), and mAP50 (0.60). By contrast, the first OT model (assigning a sample to the class that minimizes the mean distance to its prototypes) registers moderate accuracy (0.87) and precision (0.46) but relatively low recall (0.31) and F1-score (0.33), resulting in an mAP50 of only 0.39. When a k-NN classifier is applied using randomly chosen prototypes (OT + k-NN), accuracy and precision both drop further (to 0.85 and 0.33, respectively), while recall barely improves (0.29), yielding an overall F1 of 0.30 and an mAP50 of 0.29, demonstrating that naïvely-selected k-NN prototypes can harm balance between false positives and false negatives. Introducing k-means to select 100 centroids per class (OT + k-NN + k-means) recovers much of that lost performance: it raises accuracy to 0.88 (above the first OT model), boosts recall to 0.40 (a significant jump from 0.31), and improves F1 to 0.38, although precision (0.41) and mAP50 (0.36) still lag behind the simpler OT model. In short, although the centroid-based k-NN approach narrows the gap by offering more balanced recall and F1, it cannot match YOLOv11's end-to-end detection and classification performance.

TABLE I
COMPARATIVE TABLES OF THE CLASSIFICATION MODELS

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**	**mAP50**
YOLOv11	0.92	0.69	0.55	0.61	0.60
OT	0.87	0.46	0.31	0.33	0.39
OT + k-NN	0.85	0.33	0.29	0.3	0.29
OT + k-NN + k-means	0.88	0.41	0.40	0.38	0.36

VII. CONCLUSION

Although the four classification models show strong performance overall—particularly in terms of accuracy—they also present certain limitations, especially the model based on Optimal Transport (OT). One major constraint lies in the limited amount of data used during training: only 100 prototypes per class were selected out of 300,000 objects, which may hinder the model's learning capacity and generalization. Additionally, OT itself is an optimization problem, which results in high computational costs and slow inference, making it unsuitable for real-time applications. The approach is also sensitive to the choice of hyperparameters (such as the number of neighbors in KNN) and the selection quality of class prototypes. Moreover, relying on OT for classification may lead to a loss of global spatial context. To address these limitations, several improvements can be considered. One promising direction is to integrate the OT loss function directly into the detection network to enable end-to-end learning. Enhancing the prototype selection process through learning-based methods or hierarchical clustering could improve class representation. Finally, incorporating global and multi-scale context would likely enhance the model's robustness in detecting and classifying complex objects.



Model variant	d (depth_multiple)	w (width_multiple)	mc (max_channels)
n	0.50	0.25	1024
s	0.50	0.50	1024
m	0.50	1.00	512
l	1.00	1.00	512
xl	1.00	1.50	512

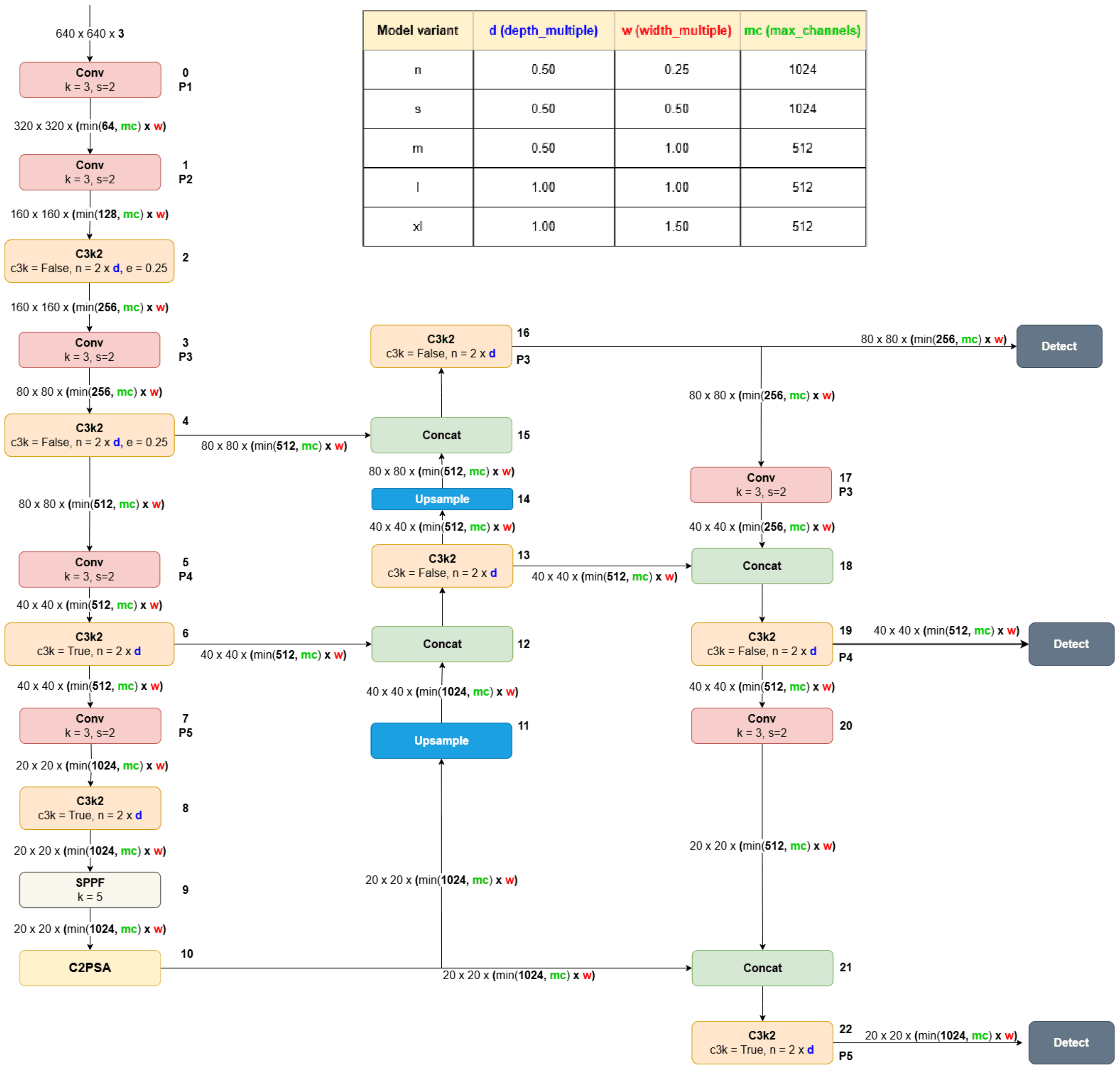

640 x 640 x 3
Conv k = 3, s=2 0 P1
320 x 320 x (min(64, mc) x w)
Conv k = 3, s=2 1 P2
160 x 160 x (min(128, mc) x w)
C3k2 c3k = False, n = 2 x d, e = 0.25 2
160 x 160 x (min(256, mc) x w)
Conv k = 3, s=2 3 P3
80 x 80 x (min(256, mc) x w)
C3k2 c3k = False, n = 2 x d, e = 0.25 4
80 x 80 x (min(512, mc) x w)
Conv k = 3, s=2 5 P4
40 x 40 x (min(512, mc) x w)
C3k2 c3k = True, n = 2 x d 6
40 x 40 x (min(512, mc) x w)
Conv k = 3, s=2 7 P5
20 x 20 x (min(1024, mc) x w)
C3k2 c3k = True, n = 2 x d 8
20 x 20 x (min(1024, mc) x w)
SPPF k = 5 9
20 x 20 x (min(1024, mc) x w)
C2PSA 10
Upsample 11
20 x 20 x (min(1024, mc) x w)
40 x 40 x (min(1024, mc) x w)
Concat 12
40 x 40 x (min(512, mc) x w)
C3k2 c3k = False, n = 2 x d 13
40 x 40 x (min(512, mc) x w)
Upsample 14
80 x 80 x (min(512, mc) x w)
Concat 15
80 x 80 x (min(512, mc) x w)
C3k2 c3k = False, n = 2 x d 16 P3
80 x 80 x (min(256, mc) x w)
Detect
80 x 80 x (min(256, mc) x w)
Conv k = 3, s=2 17 P3
40 x 40 x (min(256, mc) x w)
Concat 18
40 x 40 x (min(512, mc) x w)
C3k2 c3k = False, n = 2 x d 19 P4
40 x 40 x (min(512, mc) x w)
Detect
40 x 40 x (min(512, mc) x w)
Conv k = 3, s=2 20
20 x 20 x (min(512, mc) x w)
20 x 20 x (min(1024, mc) x w)
Concat 21
C3k2 c3k = True, n = 2 x d 22 P5
20 x 20 x (min(1024, mc) x w)
Detect

Fig. 18. YOLOv11 architecture [13].

References

[1] Ultralytics, "Yolov11: Sota computer vision model," https://yolov11.com/, 2024, accessed: 2025-05-05.

[2] W. Zhang and M. Li, "Masf-yolo: An improved yolov11 network for small object detection on drone imagery," *arXiv preprint arXiv:2504.18136*, 2025.

[3] C. Sun, Y. Chen, C. Xiao, L. You, and R. Li, "Yolov5s-dsd: An improved aerial image detection algorithm based on yolov5s," *Sensors*, vol. 23, no. 15, p. 6905, 2023.

[4] J. Doe and J. Smith, "Leaf-yolo: Lightweight edge-real-time small object detection on aerial imagery," *Journal of Aerial Imaging*, vol. 10, no. 2, pp. 123–134, 2025.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, pp. 91–99, 2015.

[6] L. Dai, H. Chen, Y. Li, C. Kong, Z. Fan, J. Lu, and X. Chen, "Tardet: Two-stage anchor-free rotating object detector in aerial images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2022, pp. 1234–1243.

[7] X. Liu, L. Wang, and H. Zhang, "Afod: Two-stage object detection based on anchor-free remote sensing images," *Computing*, vol. 105, no. 3, pp. 567–580, 2023.

[8] C. Author *et al.*, "Deep learning for feature extraction in remote sensing: A case-study of aerial scene classification," *Sensors*, vol. 20, no. 14, p. 3906, 2020.

[9] Z. Ge *et al.*, "Ota: Optimal transport assignment for object detection," *arXiv preprint arXiv:2103.14259*, 2021.

[10] A. Sadeghian *et al.*, "Automatic target recognition using discrimination

based on optimal transport," *arXiv preprint arXiv:1904.03534*, 2019.

[11] N. Papadakis, "Optimal transport for image processing; signal and image processing." *Université de Bordeaux; Habilitation thesis,*, pp. 41–44, 2017.

[12] M. Cuturi, "Introduction to (python) optimal transport sinkhorn distances: Lightspeed computation of optimal transport," *Graduate School of Informatics, Kyoto University*, pp. 3–5, 2013.

[13] LearnOpenCV, "Yolov1 - you only look once - object detection explained," 2020, accessed: 2025-05-26. [Online]. Available: https://learnopencv.com/yolo11/