Penguins vs Turtles,

Image Detection and Classification

**Group Name:**

**Group Member:**

*Abstract*—In this task, we use Yolo and KNN to process the Images. We used the dataset from Kaggle’s Competition to conduct the traning. After the 对于A，我们使用了什么，对于KNN，我们使用了什么。最后我们选择的评估参数是 ，最后得到的结果分别是，我们得到的结论是，并且给出了合理的解释和分析。

Keywords: Yolo, collaborative filtering algorithm, KNN

# **Introduction**

Object detection is a crucial task in the field of computer vision, where various machine learning (ML) and deep learning (DL) models are employed to enhance performance. In the past, two-stage object detectors were widely popular and effective. However, with recent advancements in single-stage object detection and related algorithms, they have shown significant improvements over most two-stage object detectors.

In this task, we need to deal with the image of penguins and turtles. In our group project, we aim to explore and address the object detection and classification tasks on the "Penguins versus Turtles" dataset. This dataset consists of 500 training images and 72 validation images, each containing either a penguin or a turtle, with their positions annotated in the corresponding annotation files.

Furthermore, the emergence of YOLOs (You Only Look Once) has led to remarkable applications of YOLOs in object detection and recognition, outperforming their two-stage counterparts. This motivation drives us to use YOLOs for object detection. And due to the dataset is a small one, we determine to use KNN as an extra method to complete classification task.

# **Literature Review**

## YOLO

YOLO algorithm is a real-time object detection algorithm based on deep learning. Compared to traditional object detection algorithms, YOLO has the key advantage of being highly efficient and fast, enabling simultaneous prediction of object classes and spatial locations in a single forward pass.

The development of Yolo is as follows:YOLO v1: The YOLO algorithm was first proposed in 2015. It divides the input image into a grid and makes predictions within each grid cell, enabling real-time object detection. However, it has certain limitations in detecting small objects and accurately localizing complex-shaped objects.

YOLO v2: YOLO v2 was released in 2016 as an improvement over the original version. It introduced the concept of anchor boxes to enhance object detection for different scales and aspect ratios. This version improved feature extraction by adding more convolutional layers and achieved better localization performance.

YOLO v3: YOLO v3 was further improved upon the previous version and released in 2018. It introduced the Feature Pyramid Network (FPN) to fuse multi-scale features for better detection of objects at different scales. YOLO v3 also increased the number of anchor boxes and incorporated techniques such as multi-label prediction and dynamic anchor box assignment.

YOLO v4: YOLO v4, released in 2020, brought significant breakthroughs to the algorithm. It introduced techniques such as the CSPDarknet53 backbone network, PANet, and various optimization strategies. YOLO v4 achieved the best performance in terms of accuracy, small object detection, and processing speed.

YOLO v5: YOLO v5, released in 2020, focused on improving the algorithm’s speed and accuracy. Compared to the previous version, it introduced a lightweight architecture and adopted technologies such as the EfficientDet-D7 backbone network, focal loss, and various data augmentation methods. YOLO v5 maintains high performance while being suitable for deployment on resource-limited devices.

YOLO v6：YOLO v6 supports the complete industrial application requirements of model training, inference, and multi-platform deployment. It has undergone several improvements and optimizations at the algorithmic level, including network architecture and training strategies. On the COCO dataset, YOLO v6 outperforms other algorithms of similar scale in terms of both accuracy and speed.

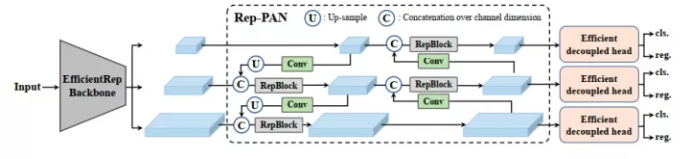


Figure1 The YOLO v6 framework (N and S are shown).

YOLO v7：YOLO v7 was released in 2022.The official version of YOLO v7 achieves higher accuracy and a 120% faster speed (in terms of FPS) compared to YOLO v5 with the same model size. It is 180% faster than YOLOX, 1200% faster than Dual-Swin-T, 550% faster than ConvNext, and 500% faster than SWIN-L. YOLO v7 surpasses all known detectors in terms of both speed and accuracy within the range of 5FPS to 160FPS. Tested on GPU V100, a model with an accuracy of 56.8% AP can achieve a detection rate of over 30 FPS (batch=1). Currently, it is the only detector capable of exceeding 30 FPS at such high accuracy.

YOLO v8：According to the official description, YOLO v8 is described as a state-of-the-art model that builds upon the success of previous YOLO versions and introduces new features and improvements to enhance performance and flexibility further. The specific innovations include a new backbone network, a new anchor-free detection head, and a new loss function. YOLO v8 is designed to run on various hardware platforms, ranging from CPUs to GPUs. However, Ultralytics has not directly named the open-source library as YOLO v8, but instead, it is referred to as “Ultralytics”. This is because Ultralytics positions the library as an algorithm framework rather than a specific algorithm. One notable characteristic is its scalability. The goal of the library is not only to support YOLO series models but also a wide range of tasks such as non-YOLO models, classification, segmentation, and pose estimation.

Two main advantages of the Ultralytics open-source library are:

1.Integration of Current SOTA Technologies

2.Support for Other YOLO Series and Beyond

Since its introduction, the YOLO algorithm has continuously evolved and become a foundational component for real-time object detection. Researchers and developers continue to explore advanced techniques and modifications to enhance the robustness, accuracy, and efficiency of the YOLO algorithm.

## KNN

1. Nearest Neighbors (KNN) algorithm is a commonly used supervised learning algorithm for classification and regression problems. Its basic principle involves calculating the distances between a test sample and the samples in the training set. It identifies the K nearest training samples and then determines the class or prediction value of the test sample based on the labels of these K samples through voting or averaging.

KNN has many advantages.

Simple and easy to understand: The KNN algorithm is relatively simple and does not require many assumptions or tuning.

Non-parametric method: KNN does not make any assumptions about the data distribution, making it suitable for various types of data.

Lazy learning: KNN is a lazy learning method, which means it stores all the training samples in memory and performs calculations only during prediction, without an explicit training process.

Suitable for multi-class problems: KNN can handle multi-class classification problems by using a voting mechanism to determine the final class.

However, KNN also has some drawbacks.

Sensitivity to outliers: KNN is sensitive to outliers because it makes judgments based on the nearest neighbor samples. If there are outliers among the neighbors, it may result in incorrect predictions.

High computational complexity: KNN has a relatively high computational cost on large datasets as it requires calculating distances between samples.

# **Methods**

## Yolo

The YOLO algorithm transforms the object detection problem into a regression problem by partitioning the entire image into a grid and utilizing a convolutional neural network (CNN) for feature extraction. Each grid cell is responsible for predicting the presence of objects in the image. For each grid cell, the algorithm predicts the bounding box coordinates and class label of the object, along with a corresponding confidence score. Finally, non-maximum suppression (NMS) is applied to filter out the most reliable detection results.

The basic steps of the YOLO algorithm are as follows:

1.Input Processing: The input image is divided into fixed-sized grids. Each grid is responsible for detecting objects in the image. Additionally, the image is often resized to a size that can be handled by the model.

2.Feature Extraction: A convolutional neural network (CNN) is used to extract features from the image. Typically, a pre-trained CNN model such as Darknet is used to extract feature maps at different scales of the image.

3.Object Detection: In each grid cell, objects are detected by predicting the position and class of bounding boxes. Each bounding box consists of the object’s location (x, y coordinates) and dimensions (width and height), along with a confidence score indicating the presence of the object.

4.Non-Maximum Suppression (NMS): To reduce redundant detections and improve detection quality, a non-maximum suppression technique is employed to filter and prune different bounding boxes, retaining only the most confident detections.

5.Output Prediction: Finally, the algorithm outputs the detected objects along with their corresponding confidence scores, typically presented in the form of bounding boxes and class labels.

The characteristics of Yolo are as follows:

Real-time Processing: The design philosophy of the YOLO algorithm aims to achieve real-time processing speed while maintaining relatively high detection accuracy. By simplifying the object detection task as a regression problem and performing detection in a single forward pass, YOLO significantly improves detection efficiency.

Global Context: YOLO performs feature extraction and object detection at the whole image level, capturing global contextual information that helps enhance detection accuracy. Compared to traditional methods that detect objects at different scales and locations, YOLO exhibits better global perception capability.

Multi-scale Detection: YOLO utilizes feature maps of different scales for object detection, effectively handling objects of various sizes and predicting different bounding boxes at different scales. This enables better handling of objects with significant scale variations.

## KNN

# Basic steps：

1.Distance Calculation: Calculate the distances between the test sample and each sample in the training set. Common distance metrics used include Euclidean distance, Manhattan distance, Minkowski distance, and others.

2.K Selection: Determine the number K of nearest neighbors to be used for voting or averaging. Choosing an appropriate value for K is important and often requires cross-validation.

3.Neighbor Determination: Select the K samples with the shortest distances as the neighbors of the test sample.

4.Voting or Averaging: For classification problems, the labels of the K neighbors are used for voting to determine the predicted class of the test sample. For regression problems, the average of the labels of the K neighbors is taken as the predicted value of the test sample.

The workflow of KNN is as follows:

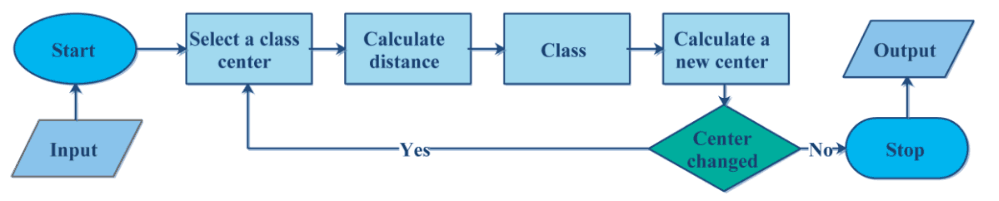


Figure2 The Steps of KNN

# Cosine similarity:

In the KNN algorithm, cosine similarity is a commonly used measure of similarity that evaluates the similarity between two vectors. Cosine similarity measures the cosine value of the angle between two vectors in a multidimensional space. It can be used to compute similarity in domains such as text, images, and more.

The calculation of cosine similarity between two vectors can be done through the following steps:

1.Normalize the two vectors: To eliminate the influence of vector lengths in calculations, it is common to normalize the two vectors to unit vectors. This is achieved by dividing each vector by its length, making them both length.

2.Compute the dot product of the vectors: Multiply the normalized vectors element-wise and sum the results to obtain their dot product.

3.Calculate the cosine similarity: Divide the dot product by the product of the lengths of the vectors.

In the KNN algorithm, cosine similarity can be used to measure the similarity between samples and determine their neighbors. By calculating the cosine similarity between an unlabeled sample and the training samples, the K most similar samples can be identified. The class labels of these neighbors can then be used for voting, possibly with weighting, to determine the class label of the unlabeled sample.

# **Experimental Results**

## Experimental enviroment

C. 实验设置：

* 提供实验设置的描述，包括硬件和软件配置。
* 提及所使用的深度学习或机器学习框架以及其版本。

Ultralytics YOLOv8.0.143

- Python-3.8.10

- torch-2.0.1+cu117

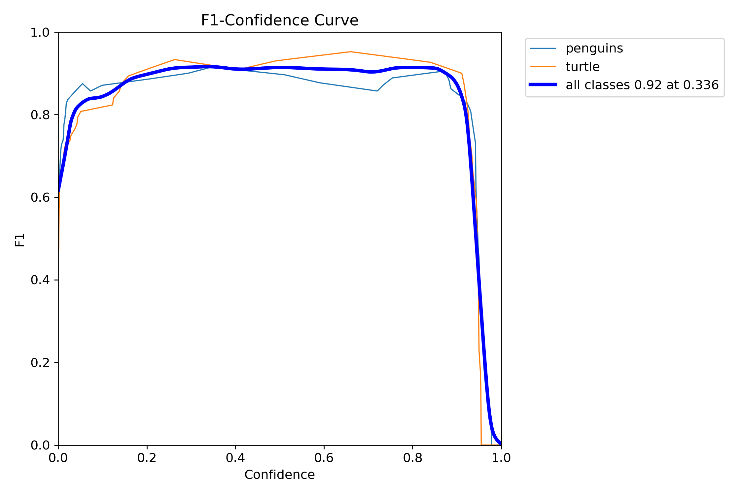
## dataset:

* This dataset consists of 500 training images and 72 validation images, each containing either a penguin or a turtle, with their positions annotated in the corresponding annotation files.
* 包括图像数量、类别数、以及对数据集进行的任何预处理步骤的细节。

## Evaluation Metrics:

## Yolos:

* 1. F1



The graph indicates that, across the entire confidence interval, the F1 score remains above 0.8, maintaining a high level.

2.

D. 基准模型（可选）：

* 如果适用，描述用于比较的基准模型。
* 说明基准模型的架构和参数。

E. 模型架构：

* 展示提出的图像识别与分类模型的详细架构。
* 包括层数、层类型（如卷积层、池化层、全连接层）以及引入的任何新颖组件。

F. 训练过程：

* 描述训练过程，包括优化算法、学习率、批大小以及训练轮数。
* 提及用于改善泛化性能的数据增强技术。

G. 实验结果：

* 展示模型在图像识别与分类任务中的性能结果。
* 包括一个总结各个模型评估指标的表格或图表。
* 讨论结果，突出提出方法的优势和不足。

H. 与基准对比（可选）：

* 如果使用了基准模型，比较提出方法与基准模型的性能。
* 讨论观察到的改进或权衡。

I. 失败案例分析（可选）：

* 如果模型存在失败案例，分析并讨论错误分类或检测性能差的原因。

J. 计算效率（可选）：

* 如适用，讨论提出模型的计算效率，包括推理时间和内存使用情况。

K. 讨论与解释：

* 解释实验结果及其对提出的图像识别与分类模型的影响。
* 讨论实际应用和潜在的未来研究方向

## Authors and Affiliations

**The template is designed for, but not limited to, six authors.** A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

### For papers with more than six authors: Add author names horizontally, moving to a third row if needed for more than 8 authors.

### For papers with less than six authors: To change the default, adjust the template as follows.

#### Selection: Highlight all author and affiliation lines.

#### Change number of columns: Select the Columns icon from the MS Word Standard toolbar and then select the correct number of columns from the selection palette.

#### Deletion: Delete the author and affiliation lines for the extra authors.

## Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is “Heading 5”. Use “figure caption” for your Figure captions, and “table head” for your table title. Run-in heads, such as “Abstract”, will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named “Heading 1”, “Heading 2”, “Heading 3”, and “Heading 4” are prescribed.

## Figures and Tables

#### Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 1”, even at the beginning of a sentence.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Example of a figure caption. (*figure caption*)

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization”, or “Magnetization, M”, not just “M”. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization {A[m(1)]}”, not just “A/m”. Do not label axes with a ratio of quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

# **Discussion**

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

# **Conclusion**

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first ...”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

# **References**

1. Kaggle. Penguins Versus Turtles: Image Classification With Bounding Boxes. 2023. <https://www.kaggle.com/datasets/abbymorgan/penguins-vs-turtles>
2. Redmon, Joseph, et al. “You Only Look Once: Unified, Real-Time Object Detection.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016.
3. Redmon, Joseph, and Ali Farhadi. “YOLO9000: better, faster, stronger.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017.
4. Redmon, Joseph, and Ali Farhadi. “YOLOv3: An Incremental Improvement.” arXiv preprint arXiv:1804.02767 (2018).
5. Bochkovskiy, Alexey, et al. “YOLOv4: Optimal Speed and Accuracy of Object Detection.” arXiv preprint arXiv:2004.10934 (2020).
6. Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In Proceedings of the Twenty-First International Conference on Machine Learning (ICML) (pp. 116-123).
7. Wilson, D. R.& Martinez, T. R. (1997). Improved heterogeneous distance functions. Journal of Artificial Intelligence Research, 6, 1-34.
8. Shiliang Sun.&Rongqing Huang.(2010).An Adaptive k-Nearest Neighbor Algorithm.East China Normal University.