



how much difference between architecture of a cnn and yolo is there?

Sources

YOLO Object Detection Explained: A Beginner's Guide - Encord

encord

Convolutional Neural Networks, Explained | by Mayank Mishra

towardsdatascience

YOLO Object Detection Algori

basic

Answer

While both Convolutional Neural Networks (CNNs) and YOLO (You Only Look Once) architectures share some similarities, there are significant differences in their structure and approach to object detection. Here's a comparison of their architectures:

CNN Architecture

CNNs typically consist of:

1. Convolutional layers
2. Pooling layers
3. Fully connected layers

The basic CNN architecture follows this pattern:

- Input layer
- Multiple convolutional and pooling layers
- One or more fully connected layers
- Output layer

CNNs are designed to extract features from images through convolution operations and reduce spatial dimensions through pooling ⁶ ₈.

YOLO Architecture

YOLO, on the other hand, is specifically designed for object detection and has a more complex architecture:

- 24 convolutional layers
- 4 max-pooling layers
- 2 fully connected layers ⁷

YOLO's architecture is similar to GoogleNet and works as follows:

1. Resizes input image to 448x448
2. Applies 1x1 convolution to reduce channels
3. Follows with 3x3 convolution
4. Uses ReLU activation for all layers except the final one
5. Employs batch normalization and dropout for regularization ⁷

Key Differences

1. **Purpose:** CNNs are general-purpose feature extractors, while YOLO is specifically designed for real-time object detection ¹ ₃.
2. **Output:** CNNs typically output class probabilities, while YOLO predicts bounding boxes and class probabilities simultaneously ³.
3. **Speed:** YOLO is designed for real-time processing, making it faster than traditional CNNs for object detection tasks ⁵.
4. **Approach:** YOLO divides the image into a grid and predicts objects within each grid cell, whereas CNNs process the entire image at once ³.

In summary, while YOLO builds upon CNN principles, its architecture is specifically optimized for fast, efficient object detection, making it distinct from general-purpose CNNs.

Share Rewrite

...

What are attention layers?

Sources

[D] - Why do Attention layers work so well? Don't weights in DNNs ...

How Attention works in Deep Learning: understanding ... - AI Summer

12. Attention Layers - deep le...

reddit

theaisummer

dmol

Answer

Attention layers are a key component in modern deep learning architectures, particularly in sequence modeling tasks like natural language processing. They allow neural networks to focus on specific parts of input data when processing information.

Structure and Function

Attention layers typically consist of three main components:

1. Query (Q)
2. Key (K)
3. Value (V)

The goal of an attention layer is to determine the relevance of different values based on the relationship between a given query and the corresponding keys ¹. This mechanism enables the network to learn which data is most relevant for a given input.

How Attention Works

1. **Attention Mechanism:** The layer computes an attention vector by measuring the relevance between a query element and all other elements in the input sequence ⁵.
2. **Weighted Summation:** Using the computed attention scores, the layer performs a weighted sum over the values to create a context vector that emphasizes important input elements ⁵.

Types of Attention

There are two main types of attention mechanisms:

1. **Additive Attention:** Computes attention scores using a feed-forward neural network on concatenated query and key vectors.
2. **Dot-Product Attention:** Measures attention scores using the dot product between query and key vectors ⁵.

Advantages

Attention layers offer several benefits:

- They allow models to focus on specific sections of input while performing a task ⁵.
- They help capture long-range dependencies in data, particularly useful for sequential or structured information ⁵.
- They improve performance by enabling selective processing of the most relevant information ⁵.

Applications

Attention mechanisms have proven effective in various domains:

- Natural Language Processing (NLP)

- Speech Recognition

For example, in machine translation tasks, attention helps align words between source and target languages, improving translation quality ⁴.

Learning Process

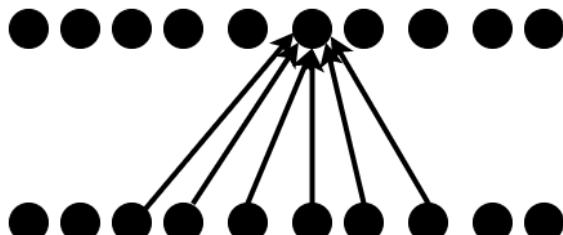
Attention weights are learned through backpropagation, similar to other neural network parameters ⁴. This allows the model to adaptively determine which parts of the input to focus on for different tasks or inputs.

In summary, attention layers provide a powerful mechanism for neural networks to selectively focus on relevant information, leading to improved performance across a wide range of deep learning applications.

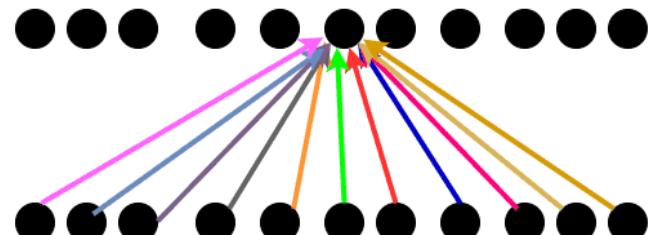
[Share](#) [Rewrite](#)

Like Dislike Save ...

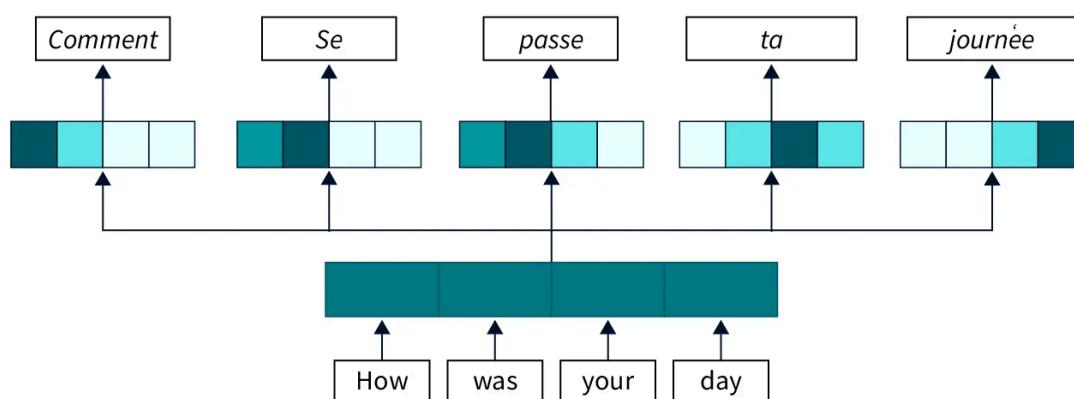
Convolution



Global attention

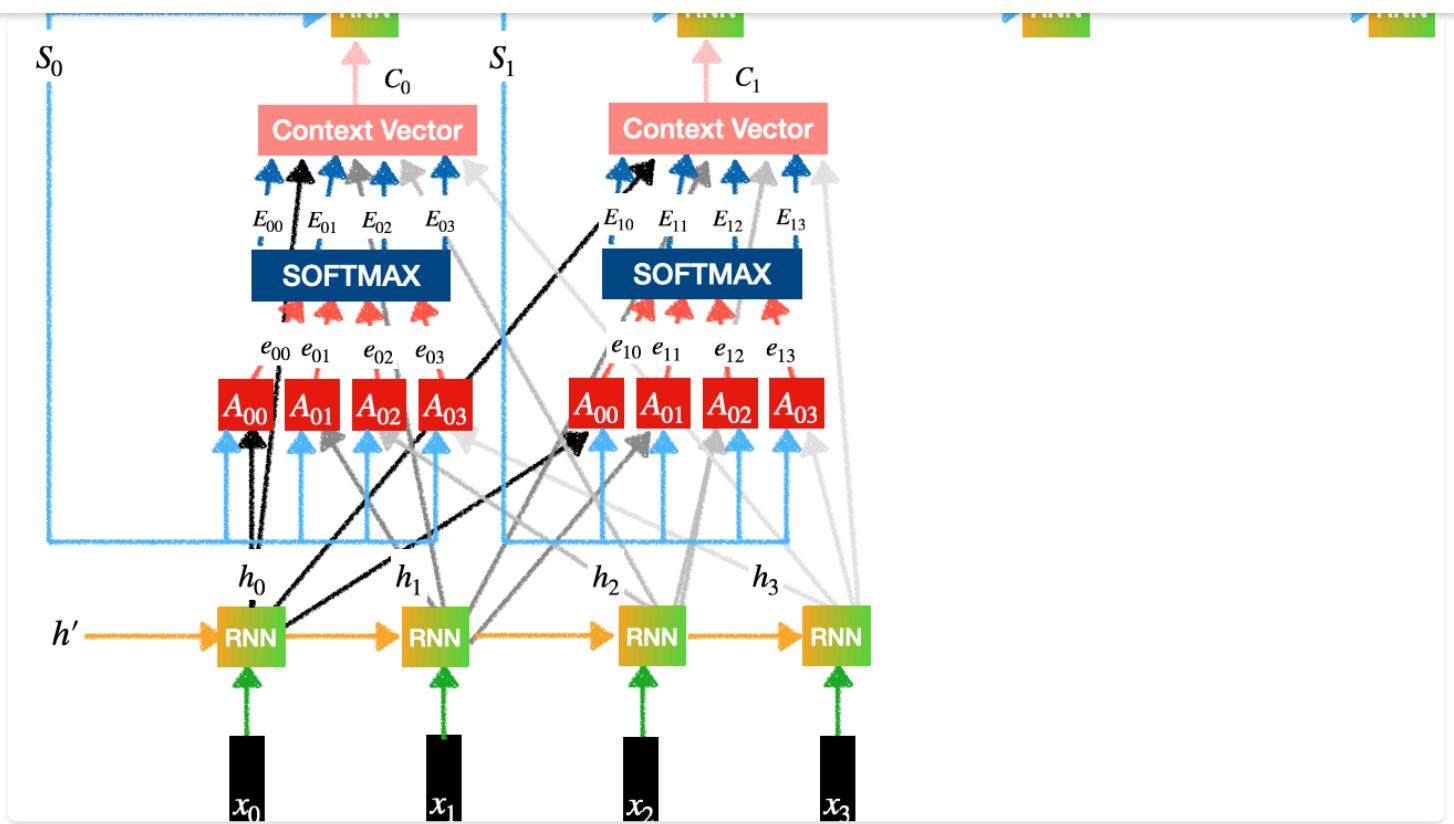


Important

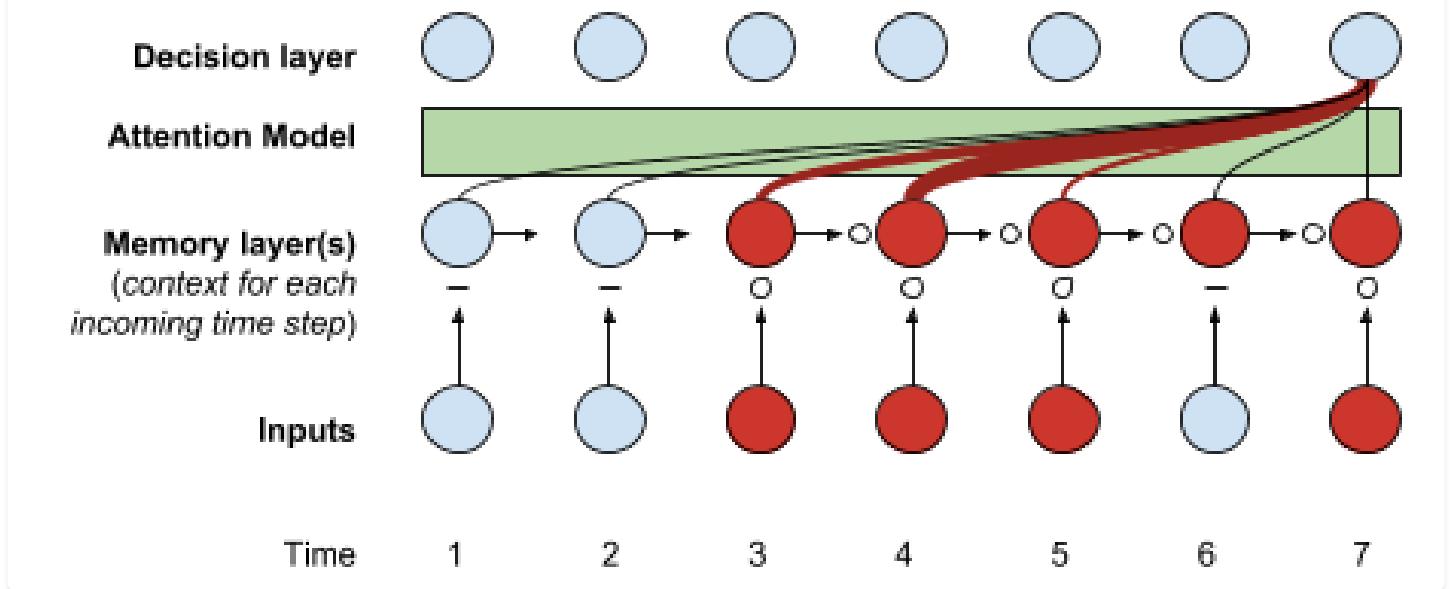


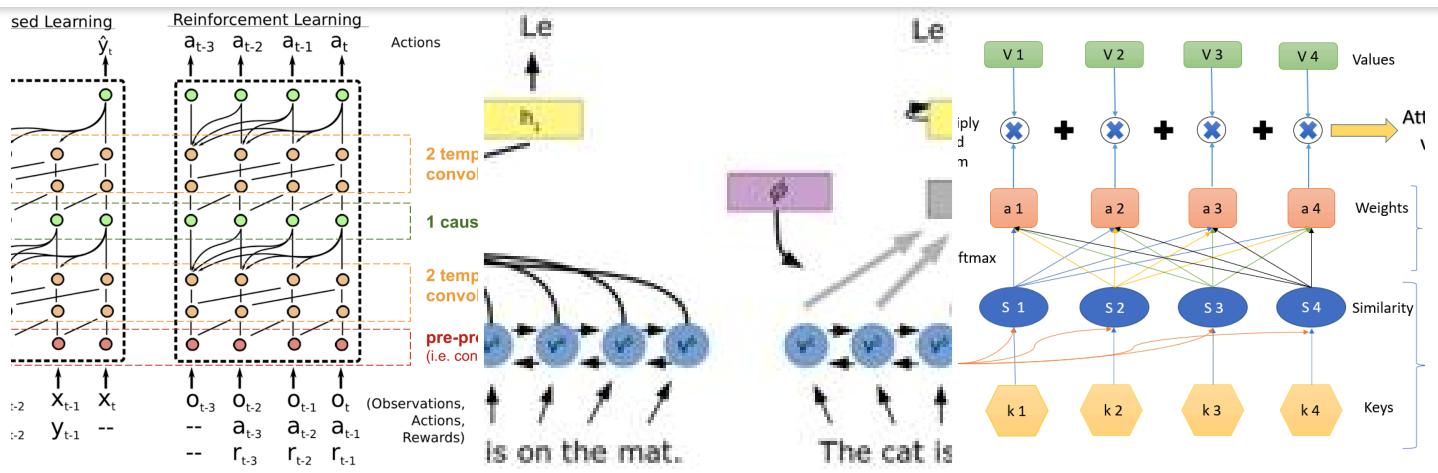
Unimportant

SCALER
Topics



Attention Mechanism





View More

Are transformers ever used in CNN-like networks?

Sources

[D] Transformers are basically CNNs? : r/MachineLearning - Reddit

A combination network of CNN and transformer for interference ...

Vision Transformers vs CNNs



Answer

Yes, transformers are increasingly being used in combination with CNN-like networks, creating hybrid architectures that leverage the strengths of both approaches. These hybrid models aim to combine the local feature extraction capabilities of CNNs with the global context modeling of transformers.

Hybrid Architectures

Several studies have explored the integration of transformers and CNNs:

1. ConvFormer: This architecture combines CNN and transformer elements for medical image segmentation ⁴. It uses a hierarchical CNN and transformer hybrid structure to capture both local and global representations.
2. CNNTF: This model combines CNNs and transformers for interference identification in communication signals ². It replaces the word embedding layer of the transformer with CNNs to better handle 1D signal data.
3. Vision Transformers with CNN elements: Some approaches use CNN-like components within Vision Transformer architectures to enhance their ability to capture local features ³.

Advantages of Combining CNNs and Transformers

The integration of these two architectures offers several benefits:

1. Complementary strengths: CNNs excel at capturing local features, while transformers are better at modeling long-range dependencies ^{1 3}.

3. Flexibility: These combined approaches allow for more adaptable architectures that can be tailored to specific tasks and data types ².

Considerations

While hybrid models show promise, it's important to note:

1. Increased complexity: Combining CNNs and transformers can lead to more complex models that may require more computational resources ⁵.
2. Task-specific design: The optimal balance between CNN and transformer elements may vary depending on the specific task and dataset ⁶.

In conclusion, the integration of transformers into CNN-like networks is an active area of research, with various hybrid architectures being developed to leverage the strengths of both approaches for improved performance across different computer vision tasks.

Share Rewrite



Do transformers ever directly operate on images?

Sources

[Vision Transformers \(ViT\) in Image Recognition: Full Guide - viso.ai](#)

[Vision transformer - Wikipedia](#)

[Image Processor - Hugging Face](#)

viso

en.wikipedia

huggingface

Answer

Yes, transformers can directly operate on images through Vision Transformers (ViT) and similar architectures. Vision Transformers process images by:

1. Splitting the input image into fixed-size patches
2. Flattening these patches
3. Creating linear embeddings from the flattened patches
4. Adding positional embeddings
5. Feeding the resulting sequence to a transformer encoder ^{1 5}

This approach allows transformers to work directly with image data without the need for convolutional layers. The ViT model, for example, treats image patches as if they were tokens in a sequence, enabling the transformer to apply self-attention mechanisms directly to visual information ^{2 4}.

Some key advantages of this direct operation on images include:

- Capturing global context effectively by processing images as sequences of patches ⁵
- Eliminating the need for traditional convolutional layers in some architectures ¹
- Achieving competitive or superior performance compared to CNNs in various image recognition tasks ^{1 2}

However, it's worth noting that some hybrid approaches combine CNN-like elements with transformers to leverage the strengths of both architectures ¹.

Share Rewrite



Is there any transformer real-time equivalent of a YOLO?

Answer

Yes, there is a transformer-based real-time equivalent to YOLO called RT-DETR (Real-Time Detection Transformer). RT-DETR, developed by Baidu, is designed to perform real-time object detection while maintaining high accuracy ^{1 5}.

Key features of RT-DETR include:

1. Efficient Hybrid Encoder: It processes multi-scale features by decoupling intra-scale interaction and cross-scale fusion, enabling real-time performance ^{1 8}.
2. IoU-aware Query Selection: This improves the initialization of object queries, enhancing detection accuracy ⁸.
3. Adaptable Inference Speed: RT-DETR allows flexible adjustment of inference speed through different decoder layers without requiring retraining ^{1 8}.

Performance-wise, RT-DETR outperforms YOLO detectors in both speed and accuracy:

- RT-DETR-L achieves 53.0% AP on COCO val2017 at 114 FPS
- RT-DETR-X achieves 54.8% AP at 74 FPS ¹

RT-DETR addresses the limitations of both YOLO models and traditional transformer-based detectors:

1. It eliminates the need for Non-Maximum Suppression (NMS), which negatively affects the speed and accuracy of YOLO models ³.
2. It overcomes the high computational cost typically associated with transformer-based detectors, making it suitable for real-time applications ^{3 5}.

In summary, RT-DETR represents a significant advancement in real-time object detection, combining the strengths of transformer architectures with the speed requirements of real-time applications, effectively challenging the dominance of YOLO models in this domain ^{1 3 8}.

 Share  Rewrite



Watch

Is RT-DETR open source?

• Sources

How to Train RT-DETR on a Custom Dataset with Transformers

pranavdurai10/rtdetr-pytorch - GitHub

RT-DETR (Realtime Detection)

blog.roboflow

github

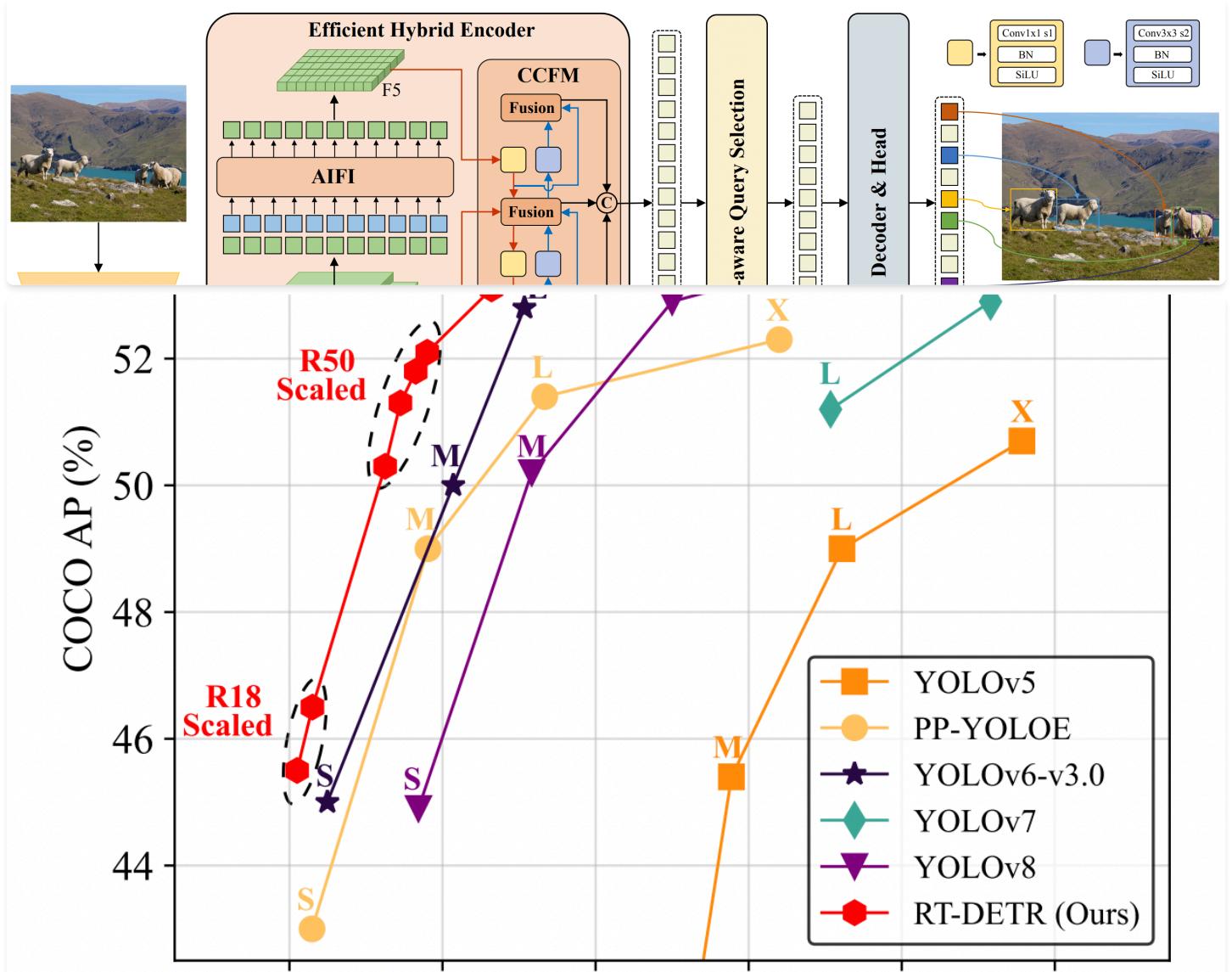
docs.ultralytics

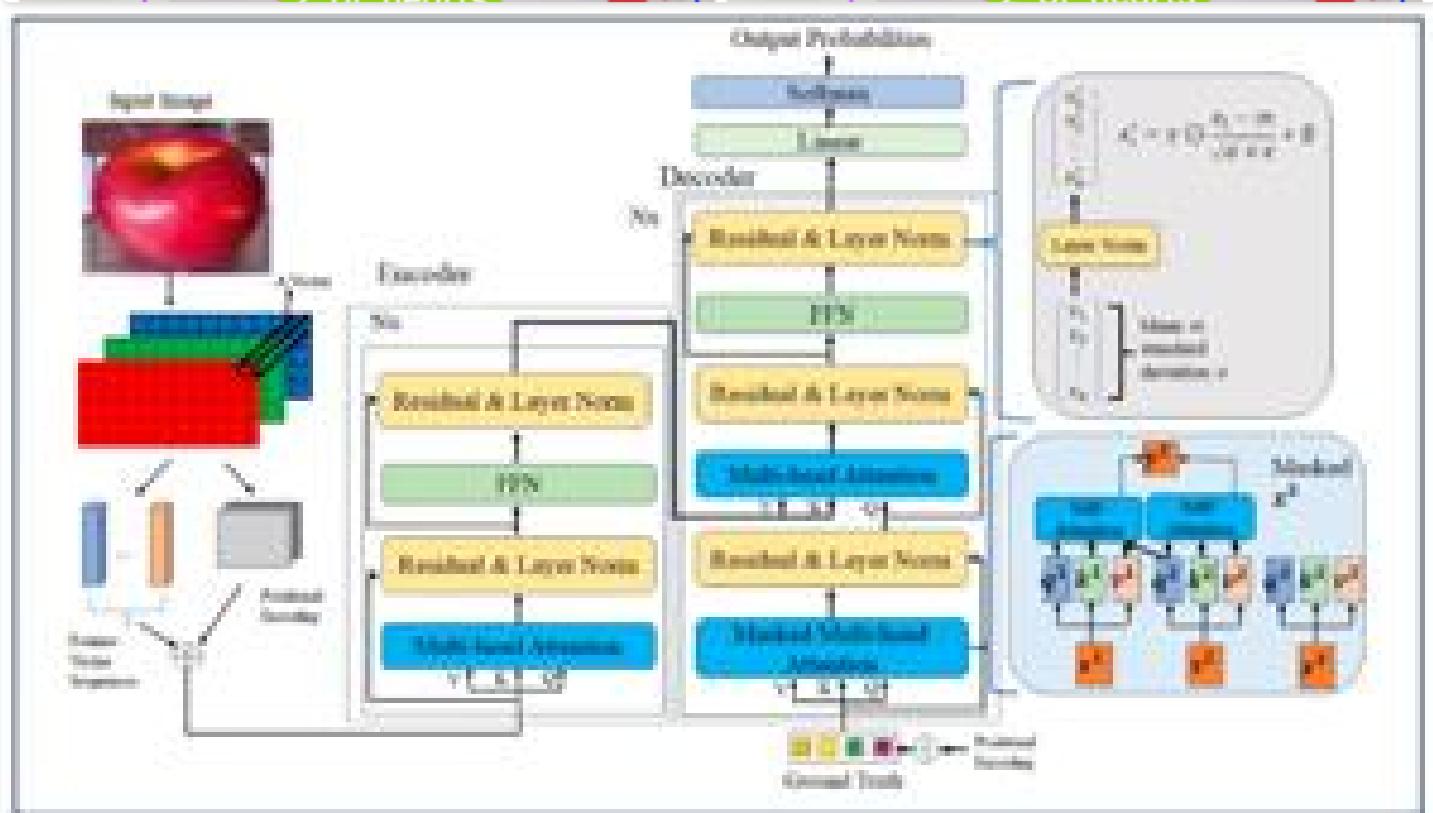
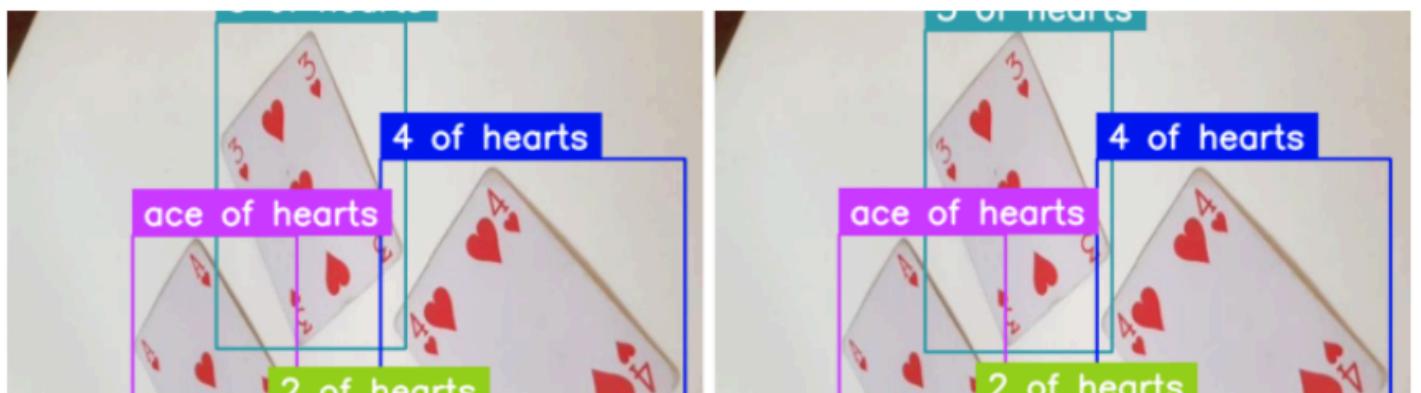
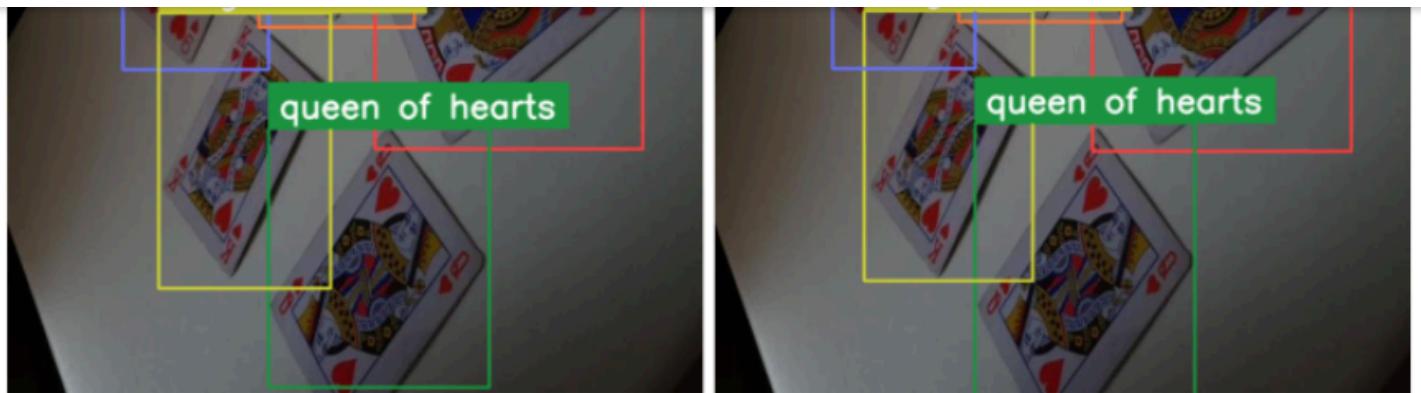
Answer

Yes, RT-DETR (Real-Time Detection Transformer) is open source. The model has been released under the Apache 2.0 license, making it freely available for use, especially in enterprise projects ¹⁻⁷. The official implementation of RT-DETR is available on GitHub, maintained by the original authors ⁵. Additionally, there are other open-source implementations and adaptations of RT-DETR, such as PyTorch versions and integrations with popular deep learning libraries like Ultralytics ²⁻⁶.

Share Rewrite

Like Dislike

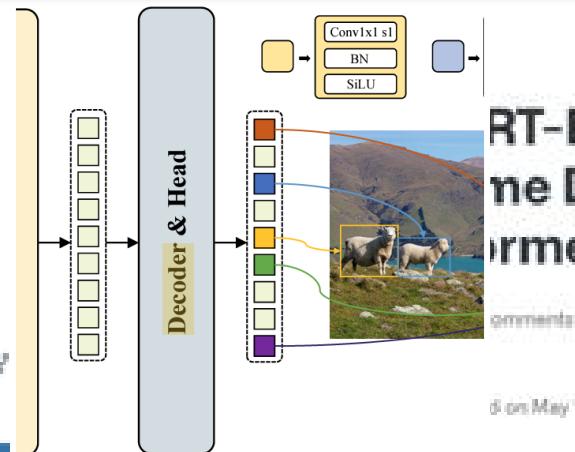




nyu/RT-DETR

Official RT-DETR (RTDETR paddle
Time DEtection TRansformer, DETRs
Real-time Object Detection. 🔥🔥

315 Issues 2 Discussions 3k Stars



RT-DETR one Detection Transformer) -...

Created on May 11, 2023

View More

What is the license for the Ultralytics version?

Sources

[How can ultralytics bypass AGPL 3.0 open source requirement](#)

[A Comprehensive Guide to Implementing Baidu's RT-DETR](#)

[RT-DETR \(Realtime Detection](#)

reddit

digitalocean

docs.ultralytics

Answer

The Ultralytics version of RT-DETR is licensed under the AGPL-3.0 (GNU Affero General Public License version 3.0) ¹ ⁶. This license has significant implications for commercial use:

1. It requires that any applications using the AGPL-3.0 licensed code must also be open-sourced.
2. For commercial applications that cannot be open-sourced due to proprietary data or other reasons, Ultralytics offers a paid commercial license option to bypass the open-source requirement ¹.

This licensing model has been described by some as a "scammy business model" ¹, as it effectively requires companies to either open-source their entire application or pay for a commercial license to use the Ultralytics implementation of RT-DETR.

It's worth noting that there are alternative implementations of RT-DETR available under more permissive licenses:

1. The original Baidu implementation of RT-DETR is available under the Apache 2.0 license, which is more permissive for commercial use ¹.
2. Other frameworks like MMDetection and RTMDet offer similar functionalities under more flexible licensing terms ¹.

For developers concerned about licensing issues, it may be advisable to explore these alternative implementations or other object detection models with more permissive licenses for commercial applications.

Share Rewrite

Home

Discover

Spaces

Library

What kind of adaptive thresholding or other feature detectors are generally used by researchers who train neural network architectures to detect stars?

Sources

Adaptive dynamic adjustment star identification threshold estimation

...

 pubmed.ncbi.nlm.nih

adaptthresh - MathWorks

 mathworks

[PDF] Using adaptive threshol
gray ...

 scholarsmine.mst

Answer

Researchers training neural network architectures to detect stars often employ various adaptive thresholding techniques and feature detectors as part of their preprocessing or initial detection steps. While there isn't a single universally adopted method, several approaches are commonly used:

Adaptive Thresholding Techniques

1. **Local Mean-Based Thresholding:** This method calculates thresholds based on the local mean intensity in the neighborhood of each pixel. It's particularly useful for images with varying background illumination ².
2. **Skewness-Corrected Adaptive Thresholding:** This technique addresses the issue of histogram skewness in star images, which can distort color detection. It adapts the threshold based on the skewness of the histogram distribution, improving segmentation of gray areas in star images ³.
3. **Otsu's Method:** While not strictly adaptive, Otsu's method is often used as a baseline or in combination with other techniques. It selects an optimal threshold by maximizing the separability of resultant classes in gray levels ³.

Feature Detection Methods

1. **Gaussian Curve Fitting:** After initial detection, the exact center of a star is often computed by fitting a Gaussian curve to the luminance profile ⁴.
2. **Round Object Detection:** Some algorithms consider a star as a round object with regularly decreasing luminance in all directions, typically with a radius of up to 50 pixels ⁴.

Additional Techniques

1. **Hot Pixel Detection:** Many star detection algorithms incorporate methods to identify and exclude hot pixels, which could otherwise be mistaken for stars ⁴.
2. **Adaptive Filtering:** Used to enhance star signals and reduce noise, particularly in images affected by light pollution ⁵.
3. **Adaptive Morphology:** This technique can help in detecting star trajectories and improving overall detection accuracy ⁵.
4. **Wavelet-Based Methods:** Some researchers use spatially adaptive wavelet thresholding for star detection, which can be effective in handling varying image characteristics ³.

It's important to note that the choice of method often depends on the specific characteristics of the star images being processed, such as background noise, star density, and image quality. Researchers may combine multiple techniques or develop hybrid approaches to achieve optimal results for their particular datasets and neural network architectures.

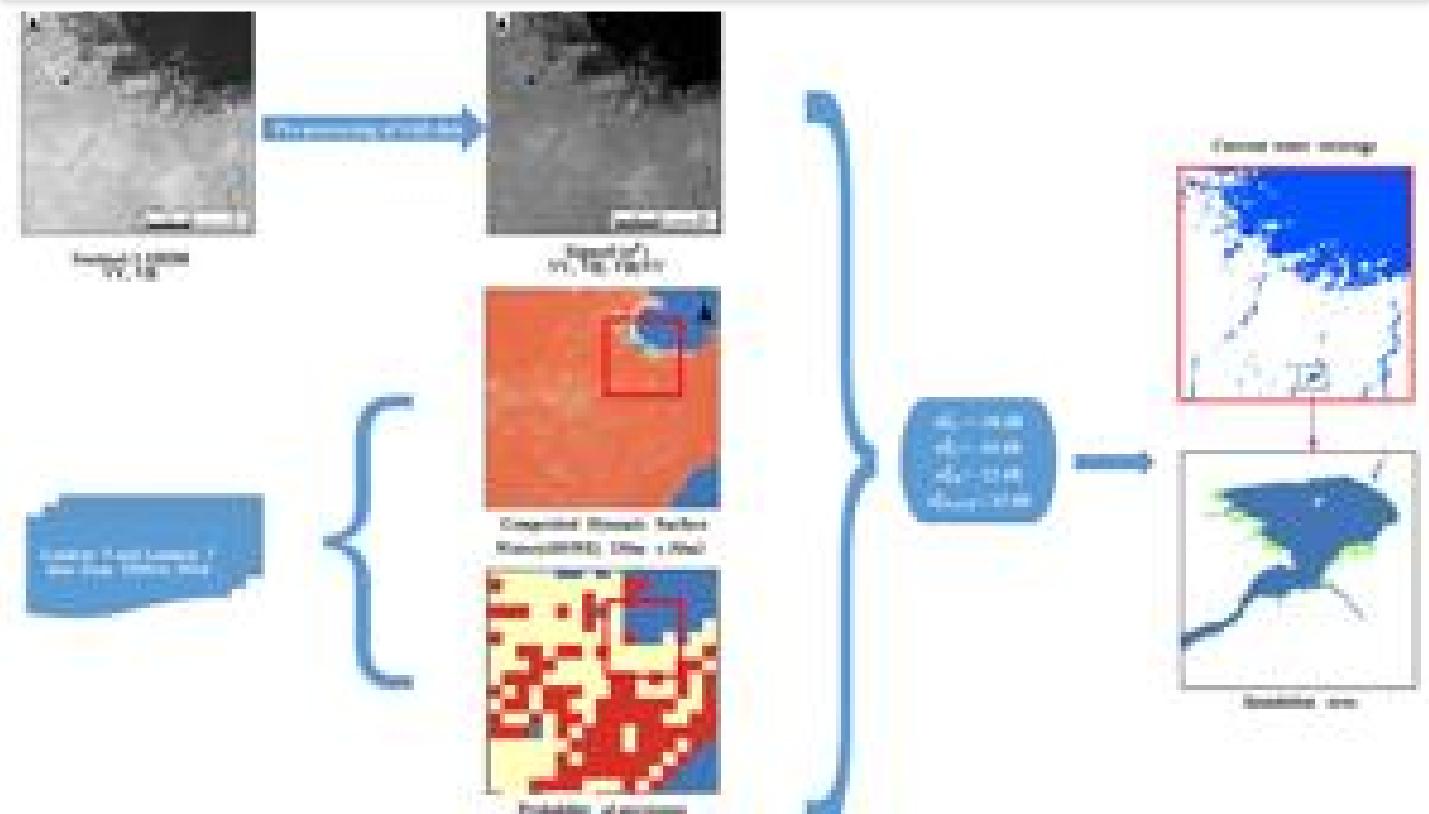
 Share  Rewrite

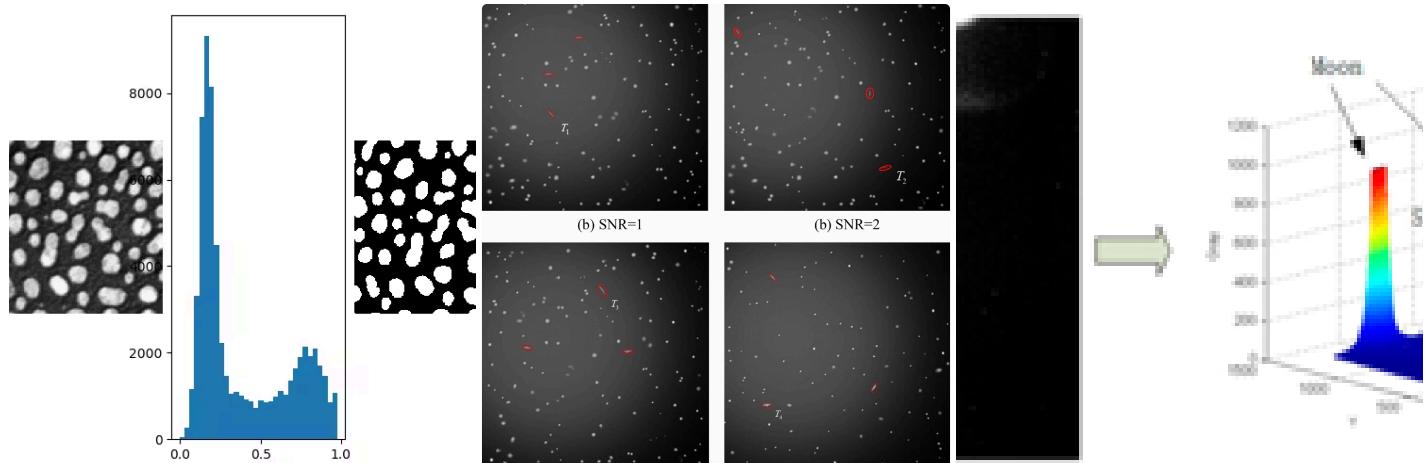
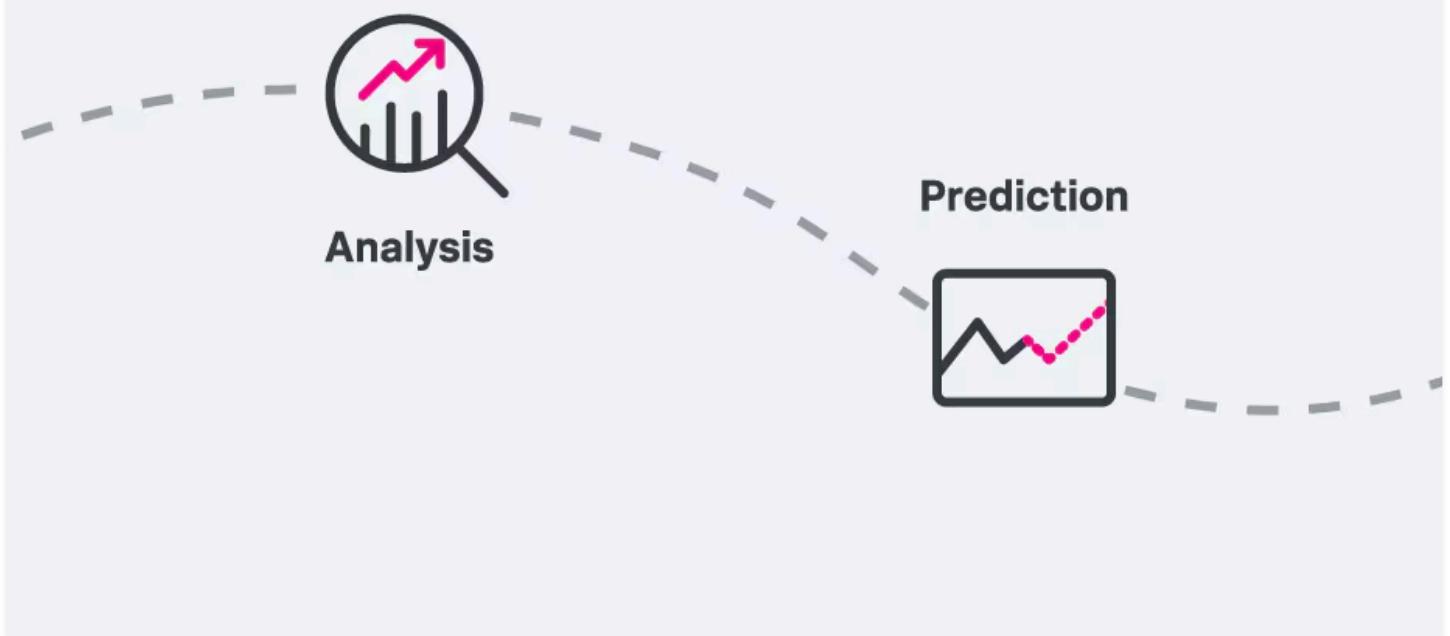


Sponsored

Watch



4 Steps of ITSI



View More

Can you write a simple python script that is a template for local mean-based thresholding?

Answer

python

```
import cv2
import numpy as np

def local_mean_threshold(image, block_size, c):
    # Convert image to grayscale if it's not already
    if len(image.shape) > 2:
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    else:
        gray = image

    # Calculate local mean
    local_mean = cv2.boxFilter(gray, -1, (block_size, block_size), normalize=True)

    # Apply thresholding
    threshold = local_mean - c
    binary = np.where(gray > threshold, 255, 0).astype(np.uint8)

    return binary

# Load an image
image = cv2.imread('path_to_your_image.jpg')

# Apply local mean-based thresholding
block_size = 51 # Size of the local neighborhood for calculating mean
c = 2 # Constant subtracted from the mean
result = local_mean_threshold(image, block_size, c)

# Display results
cv2.imshow('Original Image', image)
cv2.imshow('Thresholded Image', result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

This script does the following:

1. Defines a `local_mean_threshold` function that takes an image, block size, and a constant `c` as parameters.
2. Converts the input image to grayscale if it's not already.
3. Calculates the local mean using a box filter.
4. Applies thresholding by subtracting the constant `c` from the local mean.
5. Returns a binary image where pixels above the threshold are set to 255 (white) and below to 0 (black).
6. Loads an image, applies the thresholding, and displays the results.

You can adjust the `block_size` and `c` parameters to fine-tune the thresholding for your specific images. The `block_size` determines the size of the neighborhood for calculating the local mean, and `c` is a constant subtracted from the mean to adjust the threshold.

This template provides a starting point for implementing local mean-based thresholding, which you can further modify or integrate into a larger star detection pipeline.

 Share  Rewrite

Sources

Remote Control- Telescopius to SC - SharpCap Forums

 forums.sharpcap.co

Stellarium/stellarium-unity: Bridge tools to combine ... - GitHub

 github

stevejcl/dwarfium: App to con

 github

Answer

To get star information for pixels in your camera image while using Stellarium, you can follow these steps:

1. Align your camera and Stellarium:

Ensure that your outdoor camera and Stellarium are pointed in the same direction and set to your local time and location.

2. Perform image thresholding:

Apply local mean-based thresholding or other suitable techniques to identify potential stars in your camera image.

3. Establish a coordinate system:

Create a mapping between your camera's pixel coordinates and celestial coordinates (right ascension and declination).

4. Use Stellarium's API:

Leverage Stellarium's remote control plugin to query star information. The plugin starts a web server that allows access to Stellarium data ³.

5. Query star data:

For each identified star in your thresholded image, send a request to Stellarium's API using the mapped celestial coordinates.

6. Parse the response:

Extract relevant star information from Stellarium's API response, which may include the star's name, magnitude, and other details.

7. Match and annotate:

Associate the retrieved star information with the corresponding pixels in your camera image.

To implement this workflow, you'll need to:

- Develop a coordinate transformation algorithm to convert between pixel and celestial coordinates.
- Set up communication with Stellarium's remote control plugin, typically accessible via HTTP requests to `http://<localhost or IP>:<port>/api/main/status` ³.
- Create a system to efficiently query and process star data for multiple points in your image.

Keep in mind that precise alignment and calibration between your camera and Stellarium will be crucial for accurate results. You may need to implement additional calibration steps to account for any discrepancies between the real sky and Stellarium's representation.

 Share  Rewrite

Related

[How can I link my camera images directly to Stellarium for real-time star identification](#) 

[What tools can I use to synchronize my camera's field of view with Stellarium](#) 

[Is it possible to automate the process of matching camera pixels with star positions in Stellarium](#) 

[Can I use ASCOM to integrate my camera with Stellarium for accurate star tracking](#) 

[How do I set up Stellarium to receive coordinates from my camera for real-time star identification](#) 

