YOLOv8 Algorithmus und seine Verbesserungen im Vergleich zu früheren Versionen

Benjamin Albrecht

28. Juni 2024

Inhaltsverzeichnis

1 Aufbau und Funktionsweise des Algorithmus

1.1 YOLOv8 und frühere Versionen

Die Verbesserungen in YOLOv8 im Vergleich zu früheren Versionen beziehen sich auf die Non-Maximum Suppression, Anchor-Free Strategies und Loss Functions.

Non-Maximum Suppression (NMS) ist ein Verfahren, mit dem die Anzahl der überlappender Bounding Boxes reduziert wird. Denn dies wählt die beste Bounding Box aus einer Gruppe überlappender Boxen aus. Ist eine Bounding Box unter einer besimmten Konfidenzschwelle, wird diese verworfen. Verbleibende Bounding Boxes werden nach ihren Konfidenzwerten sortiert. In frühere Versionen wurde das klassische NMS verwendet, bei dem eine Box mit dem höchsten Konfidenzwert beibehalten und andere Boxen, die stark überlappen (über einem bestimmten IoU-Schwellenwert), verworfen werden.

YOLO basiert in früheren Versionen auf vordefinierte Ankerboxen. YOLOv8 hingegen nicht, denn hier werden ankerfreier Mechanismen verwendet, um flexibler auf unterschiedliche Objekte zu reagieren. Dies erspart ebenfalls den Vorverarbeitungsaufwand für das Definieren von Ankerboxen.

Der Loss Function misst die Differenz zwischen den vorhergesagten Ausgaben des Modells und die tatsächlichen Labels in den Trainingsdaten und versucht diese zu minimieren. In YOLOv8 wurde dieser verbessert mit dem Ziel eine höhere Genauigkeit von Objekten zu erreichen [1, 2, 3].

1.2 Mathematischer Hintergrund / Mechanismen im Hintergrund

1.3 Non-Maximum Suppression

In YOLOv8 kommt der Efficient-IoU in Einsatz, wenn es um das Behandeln von Bounding Boxes geht. Und es wurde eine weiche Strategie aus Soft-NMS angenommen, um die Confidence Scores der überlappenden Boxes zu verringern, anstatt sie zu verwerfen. Denn dieser Einsatz ermöglicht die Erhaltung von mehr Bounding Boxes, die überlappende Objekte enthalten. Des Weiteren wird der verbesserte Algorithmus zur Vereinfachung als E-Soft-NMS bezeichnet.
Der E-Soft-NMS wird wie folgt berechnet:

$$s_i = \begin{cases} s_i, & \mathrm{EIoU}(M, b_i) < t \\ s_i e^{-\frac{\mathrm{EIoU}(M,b_i)^2}{\sigma}}, & \mathrm{EIoU}(M, b_i) \geq t \end{cases} \tag{1}$$

S_i steht für den Confidence Score. M für die Box mit maximalem Confidence Score. b_i steht für die andere Bounding Box und t ist der Schwellenwert. Überschreitet der $EIoU$-Wert zwischen b_i und M den Schwellenwert, soll die Punktzahl von b_i mit der Abnahmefunktion multipliziert werden. Wenn nur der Confidence Score gesenkt wird, können redundante Boxes, die dasselbe Ziel vorhersagen, vorhanden sein. Um ihre Entfernung zu verhindern, wird eine Gaußsche Abnahmefunktion verwendet, die mit $EIoU$ zusammenhängt. Detektionsrahmen, die weit von M entfernt sind, sind nicht betroffen, während solche, die sich nahe bei M befinden, stärker bestraft werden. Somit werden Boxes, die unterschiedliche Ziele vorhersagen, erhalten bleiben, redundante Rahmen, die dasselbe Ziel vorhersagen, jedoch entfernt.
Die Formel für die Gaußsche Abnahmefunktion lautet:

$$e^{-\frac{\mathrm{EIoU}(M,b_i)^2}{\sigma}}$$

Und für die EIoU-Berechnung:

$$\text{EIoU} = 1 - \text{IoU} + \frac{\rho^2(o, o^*)}{c^2} + \frac{\rho^2(w, w^*)}{C_w^2} + \frac{\rho^2(h, h^*)}{C_h^2}$$

Wobei **o** für das Zentrum von b_i, o^* für das Zentrum von M, p für den euklidischen Abstand zwischen **o** und o^* stehen. w und w^* in die Breiten von b_i und M. h und h^* sind die Höhen von b_i und M. c ist die diagonale Länge der minimalen Box B, der b_i und M umschließt. Die Breite und Höhe B sind C_w und C_h [4].

1.4 Anchor-Free Strategies

Im Rahmen des ankerfreien Mechanismus von FCOS stellt die folgende Formel die Basis dar, da sie Abstände eines Rasterpunkts zu den Rändern der Bounding Box beschreibt, was eine präzise und flexible Objekterkennung ermöglicht, ohne auf vordefinierte Ankerboxen angewiesen zu sein [5, 6].

$$l^* = x - x_0^{(i)}, \qquad t^* = y - y_0^{(i)},$$
$$r^* = x_1^{(i)} - x, \qquad b^* = y_1^{(i)} - y.$$

x und y stehen für die Koordinaten des Punktes auf dem Feature-Grid, der als Zentrum der Bounding Box verwendet wird. $x_0^{(i)}$ und $y_0^{(i)}$ sind die Koordinaten des oberen linken Eckpunkts der Ground Truth Bounding Box i. l^* steht für den horizontalen Abstand vom Punkt (x, y) zur linken Grenze der Bounding Box. t^* steht für den vertikalen Abstand vom Punkt (x, y) zur oberen Grenze der Bounding Box. Für den horizontalen Abstand vom Punkt (x, y) zur rechten Grenze der Bounding Box steht r^*. Und b^* für den vertikalen Abstand vom Punkt (x, y) zur unteren Grenze der Bounding Box. Das bedeutet, dass anstatt feste Ankerboxen zu verwenden, die Bounding Box durch die Berechnung dieser Distanzen relativ zu einem Punkt auf dem Grid bestimmt wird [5, 6].

1.5 Loss Calculation

Die zwei Teile, aus denen der Verlustberechnungsprozess besteht, sind die Stichprobenzuweisungsstrategie und die Verlustberechnung. In modernen Detektoren werden dynamische Stichprobenzuweisungsstrategien wie simOTA von YOLOX, TaskAlignedAssigner von TOOD und DynamicSoftLAvelAssigner von RTMDet verwendet. Der YOLOv8-Algorithmus integriert die in TaskAlignedAssigner von TOOD verwendete Strategie direkt [7].

Zusammengefasst kann die Matching-Strategie von TaskAlignedAssigner mit einfachen Termen beschrieben werden. Die positiven Stichproben werden basierend auf den gewichteten Bewertungen von Klassifizierung und Regression ausgewählt.

$$t = s^{\alpha} \times u^{\beta},$$

wobei s die Vorhersagebewertung ist, die der Ground-Truth-Kategorie entspricht. u ist die Intersection Over Union (IoU) des Vorhersage-Bounding Boxes und des Ground Truth (gt)-Bounding Boxes.
Der Task-Aligned-Assigner berechnet für jede Ground Truth die Alignment-Metrik für jeden Anker, indem er das gewichtete Produkt zweier Werte nimmt: den vorhergesagten Klassifizierungswert der entsprechenden Klasse und die IoU zwischen der vorhergesagten Begrenzungsbox und der Ground Truth-Begrenzungsbox.
Es werden für jede Ground Truth die größten Top-k-Samples direkt auf der Grundlage der Alignment_metrics-Werte als positiv ausgewählt. Die Verlustberechnung besteht aus der Klassifizierung und der Regression, ohne den „Objectness Loss“ im vorherigen Modell [8].

1.6 Architektur von YOLOv8

YOLOv8 besteht aus einer strukturierten Reihe von Convolutional Neural Networks (CNNs) mit den die Lokalisierung und Klassifizierung von Objekten ermöglicht wird. Die Hauptkomponente der Architektur sind Backbone, Neck und Head.

- **Backbone**: Ein tiefes CNN, das als Feature-Extraktor dient. Wesentliche Merkmale aus dem Eingabebild werden hiermit extrahiert.
- **Neck**: Verarbeitet die extrahierten Merkmale und kombiniert die Informationen, um sie an den Head weiterzugeben.
- **Head**: Empfängt die extrahierten Merkmale und die kombinierten Informationen und gibt Bounding Boxes und Class Scores für jedes erkannte Objekt aus.

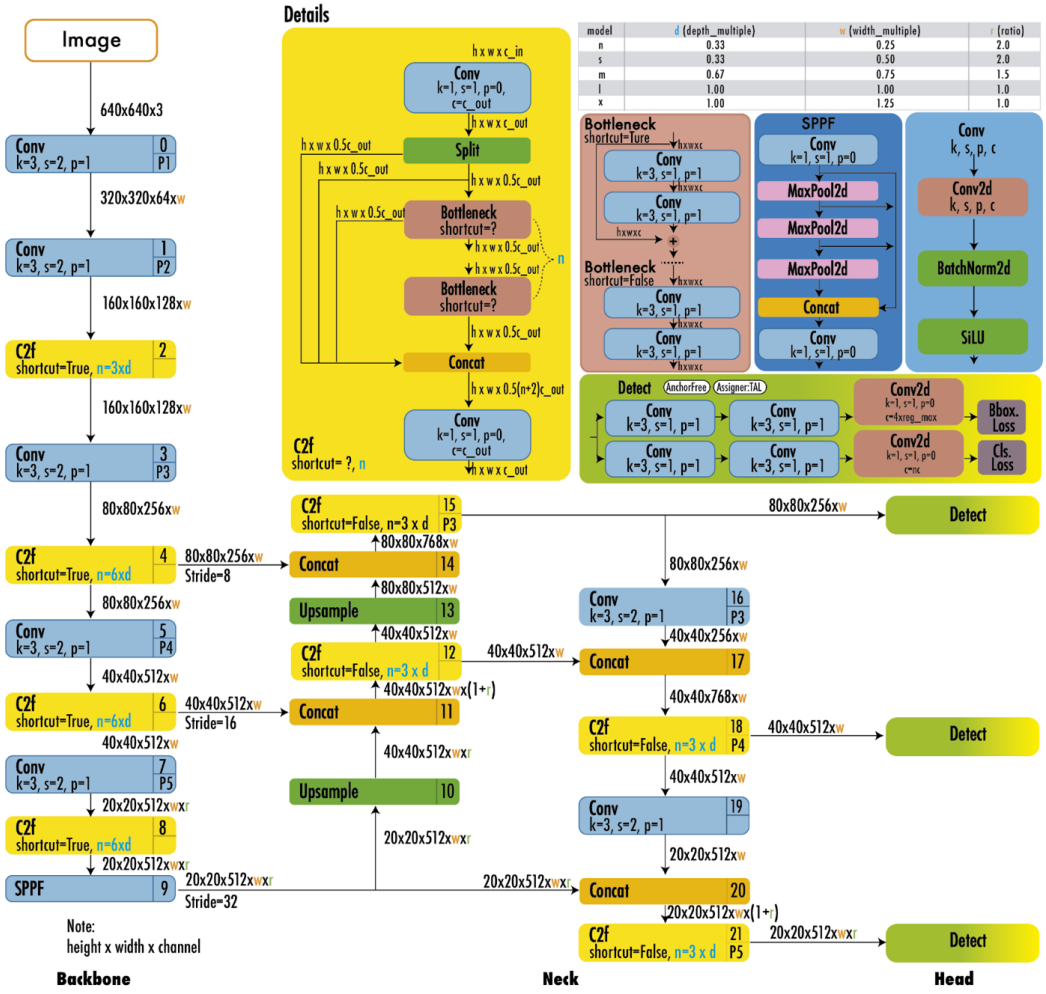

Abbildung 1: YOLOv8 Architektur

Die Tabelle in der Abbildung [Nils muss hier die Nummer der Abbildung einfügen] enthält Modellgröße, -tiefe, -breite und -verhältnis. Es lässt sich bemerken, dass die Tiefe des Modells bei höheren Modellgrößen steigt. Dies gilt ebenfalls für die Tiefe. Jedoch sinkt das Skalierungsverhältnis bei steigenden Modellgrößen, da größere Modelle bereits höhere Tiefe- und Breitewerte haben und somit von Natur aus tiefer und breiter sind. Zusätzliche Skalierung könnte zu hohem Rechenaufwand führen [1].
Die Werte in der Tabelle (bis auf die Skalierungsverhältnisse) orientieren sich an den Werten, die in der Kongigurationsdatei "yolov8.yaml"festgelegt sind. Dort sind ebenfalls die einzelnen Blöcke nummeriert zu sehen. Und Neck wird im Head integriert. Das Bild der Architektur dient der Vereinfachung. Für das Trainieren des Modells wird in dieser Hausarbeit die Modellgröße n

verwendet [9].
Der meistgenutzte Block ist der Convolutional Block. Dieser besteht aus 2D-Convolutional Layer, 2D-Batch Normalization und einer SiLU-Aktivierungsfunktion.
Ein Eingabebild wird zu Beginn durch zwei Convolutional Blöcken bearbeitet. Es wird durch diese Blöcke anhand der fesgelegten Kernel-, Strike- und Paddinggröße verkleinert.
Anstatt von traditionellen Feature Pyramid Networks (FPNs) verwendet YOLOv8 ein neuartiges C2f-Modul, das semantische Merkmale auf hoher Ebene mit räumlichen Informationen auf niedrige Ebene kombiniert. Dies ermöglicht eine effizientere und präzisere Objekterkennung vor allem bei kleinen Objekten [7]. Bottelnecks im CSPDarknet53-Backbone werden für die Reduzierung der Rechenkomplexität ohne Verlust der Genauigkeit genutzt. Spatial Pyramid Pooling Fast (SPPF) erfasst Merkmale auf meheren Skalen und verbessert die Erkennungsleistung weiter. Das Upsample-Layer erhöht die Auflösung der Feature-Maps und das Concat-Layer komibiniert dann die Feature-Maps [1]. Das Detection Head verwendet eine modifizierte Version des YOLO-Heads, die eine dynamische Ankerzuweisung und eine neuartige IoU-Verlustfunktion (Intersection over Union) enthält [7].

Literatur

[1] J. Terven, D.-M. Córdova-Esparza und J.-A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Mach Learn Knowl Extr*, Jg. 5, Nr. 4, S. 1680–1716, 2023. DOI: 10.3390/make5040083.

[2] D. Reis, J. Kupec, J. Hong und A. Daoudi, "Real-Time Flying Object Detection with YOLOv8," Mai 2023, Accessed: 8 June 2024. [Online]. Available: https://arxiv.org/pdf/2305.09972.

[3] M. Safaldin, N. Zaghden und M. Mejdoub, "An Improved YOLOv8 to Detect Moving Objects," *IEEE Access*, Jg. 12, S. 59 782–59 806, 2024. DOI: 10.1109/ACCESS.2024.3393835.

[4] T. Han, T. Cao, Y. Zheng, L. Chen, Y. Wang und B. Fu, "Improving the Detection and Positioning of Camouflaged Objects in YOLOv8," *Electronics (Basel)*, Jg. 12, Nr. 20, S. 4213, Okt. 2023. DOI: 10.3390/electronics12204213.

[5] Z. Tian, C. Shen, H. Chen und T. He, "FCOS: Fully Convolutional One-Stage Object Detection," Apr. 2019.

[6] C. Wang, Z. Luo, S. Lian und S. Li, "Anchor Free Network for Multi-Scale Face Detection," in *2018 24th International Conference on Pattern Recognition (ICPR)*, Aug. 2018, S. 1554–1559. DOI: 10.1109/ICPR.2018.8545814.

[7] Ultralytics, *YOLOv8 Architecture*, https://yolov8.org/yolov8-architecture/, Accessed: 2024-06-28, 2024.

[8] OpenMMLab, *Dive into YOLOv8: How does this state-of-the-art model work?* https://openmmlab.medium.com/dive-into-yolov8-how-does-this-state-of-the-art-model-work-10f18f74bab1, Jan. 2023.

[9] Ultralytics, *YOLOv8 Model Configuration*, Accessed: 2024-06-28, 2023. Adresse: https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/models/v8/yolov8.yaml.