# Analysis of YOLOv8 Model Architecture and Trainable Parameters

**Zhixiong Li[1], Guojin Li[2], Qiannan Li[2]**

[1]Guangxi Financial Vocational and Technical College, Nanning 530007, Guangxi
[2]Central South Architectural Design Institute Co., Ltd., Wuhan 434400, Hubei

**Abstract:** ***With the rapid development of artificial intelligence technology, the complexity and sophistication of neural network models are increasing at an unprecedented rate. Convolutional Neural Networks (CNNs), as one of the key technologies in the field of deep learning, have demonstrated excellent performance in tasks such as image recognition and object detection. Given the significant characteristics of CNNs in terms of depth, breadth, and module diversity, this paper focuses on the cutting-edge open-source object detection project YOLOv8 of ultralytics, deeply analyzes the core components and working principles of its network architecture, and elaborates on the functions of key layers (such as convolutional layers, pooling layers, residual connection layers, upsampling layers, connection layers) and the calculation methods of their trainable parameters in combination with the project source code.***

**Keywords:** Convolutional Neural Networks; YOLOv8; Network Architecture; Trainable Parameters.

## 1. INTRODUCTION

Object detection is the focus and hotspot of research in the field of computer vision. Since the YOLO (You only Look Once) algorithm was first proposed by Joseph Redmon in 2016, after continuous evolution, deep iteration and continuous optimization, it has now developed to the 10th version - YOLOv10. With the upgrade of the version, the model network architecture has been gradually optimized, the number of parameters has gradually increased, the detection speed is faster, the algorithm accuracy is higher, and the structure design is more flexible.

Shi et al. (2026) proposed a multi-stage domain-grounded synthetic data generation approach for fine-tuning large language models in telecommunications [1]. Junxi et al. (2024) introduced GCN-MF, a graph convolutional network based on matrix factorization for recommendation systems [2]. Hu et al. (2024) presented multiscale deep neural networks for text sentiment analysis [3]. Shen et al. (2026) developed a survival risk prediction model for colorectal cancer patients using graph convolutional networks and TCGA multi-omics data integration [4]. Tian et al. (2025) proposed a cross-attention multi-task learning approach for digital advertising recall [5]. Zhang (2026) constructed a hybrid LSTM-GARCH model integrating volatility factors from US financial markets [6]. Han et al. (2026) designed a dual-scale model with collaborative reasoning and multi-feature fusion for robust AI-generated image detection [7]. Liu (2025) systematically evaluated deep learning architectures for plant image classification [8]. Gao (2026) designed an intelligent operations and public relations collaborative management model for corporate image enhancement in smart manufacturing environments [9]. Shen et al. (2025) applied the whale optimization algorithm to financial payment fraud detection [10]. Sun (2025) proposed adaptive interfaces for personalized user experience using machine learning [11]. Zhou (2026) diagnosed bottlenecks in international automotive sales funnels using gradient boosting trees [12].

This article mainly introduces the common modules of CNN: the convolutional layer and the batch normalization layer (Batch Normalization Layer, BN Layer), and takes YOLOv8 as an example to analyze the network structure of YOLOv8 and calculate the trainable parameters of each module in detail.

## 2. CNN KEY LAYER FUNCTION ANALYSIS

The overall architecture of a convolutional neural network is divided into an input layer, a convolutional layer, a pooling layer, and a fully connected layer . The complexity of the neural network determines the setting of its parameters. Some of these parameters will be dynamically adjusted and optimized during the training process to adapt to the requirements of the learning task. For example, the convolutional weight parameters (represented by the Conv1d/2d/3d functions and the batch normalization function in the PyTorch framework). Some parameters are set to fixed values and will not be updated during the training process, including the parameters of components such as the pooling layer, activation function, and loss function, which remain constant.

As the depth and breadth of the neural network expand, its structure becomes increasingly complex, and its generalization ability is significantly improved. In this context, the number of trainable parameters of the model becomes a key indicator for evaluating the performance and complexity of the model. Taking the popular artificial intelligence framework PyTorch as an example, this article will delve into the specific applications, functional implementations, and parameter calculation methods of the convolutional layer and the batch normalization layer to facilitate more efficient model design and optimization.

### 2.1 Conv Convolution

In the Pytorch framework, the convolutional layer is usually implemented using the following functions: nn.Conv1d/2d/3d, nn.ConvTranspose1d/2d/3d, nn.LazyConv1d/2d/3d, nn.LazyConvTranspose1d/2d/3d, nn.Unfold/fold functions. In the field of object detection, nn.Conv2d is the most commonly used. Its function prototype and parameter description are as follows:

torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias= True, padding_mode='zeros', device=None, dtype=None)

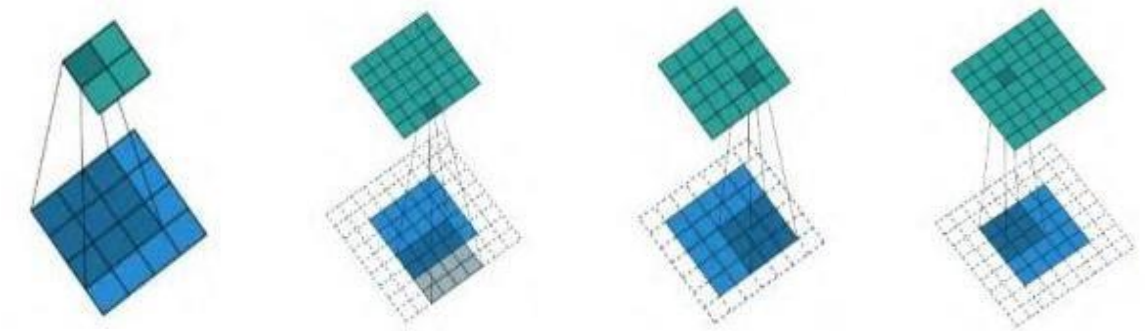
**Table 1:** nn.Conv2d Function Parameter Table

| Serial number | Parameter name | Explanation |
|---|---|---|
| 1 | in_channels | Number of input channels |
| 2 | out_channels | Number of output channels |
| 3 | kernel_size | Convolution kernel size |
| 4 | stride | Stride, default is 1 |
| 5 | padding | Padding amount of the convolution kernel, value is an integer or tuple, default is 0 |
| 6 | dilation | Control point spacing, default is 1 |
| 7 | groups | Number of groups, controlling the connection between input and output, default is 1 |
| 8 | bias | Bias parameter, default is True |
| 9 | padding_mode | Padding method, including zeros, reflect, replicate, circular, default is zeros |
| 10 | device | Computing device, usually CPU or GPU |
| 11 | dtype | Data type |

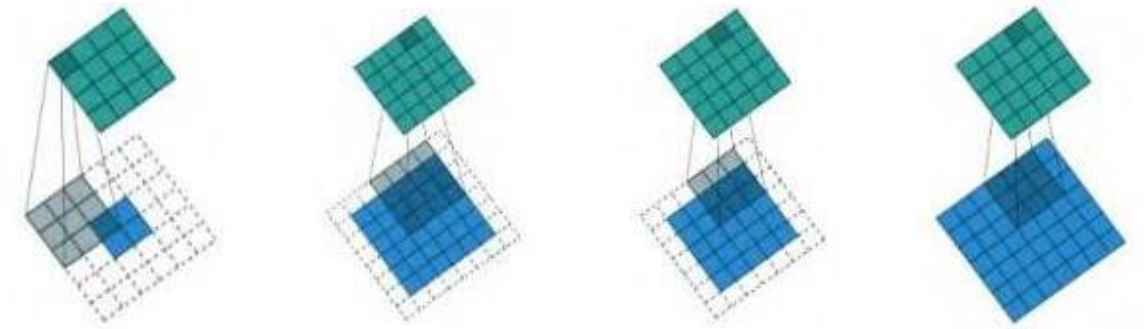
Weight parameter calculation formula of Conv2d:

$$\text{Trainable parameter} = (\text{number of input channels}/\text{number of groups}) \times \text{convolution width} \times \text{convolution height} \times \text{number of output channels} \quad (1)$$

According to different configuration parameters (such as convolution kernel size, stride, padding method, etc.), common convolution patterns are shown in Figure 1.



No padding, no stride | Arbitrary padding, no stride, transposed | Half padding, no stride | Full padding, no stride, transposed



No padding, no stride, transposed | Arbitrary padding, no stride, transposed | Half padding, no stride, transposed | Full padding, no stride, transposed

**Figure 1:** Common Convolution Methods

### 2.2 BN Batch Normalization

Batch normalization is a technique for accelerating the training of deep networks. It improves the training speed and model performance by reducing internal covariate shift, enhances gradient propagation, and strengthens the generalization performance of the model. Four normalization techniques, BatchNorm, Layer-Norm, InstanceNorm, and GroupNorm, are derived from the normalization technology. Yuxin Wu, Kaimin He[6] conducted an in-depth analysis of the principles of these four normalization techniques, as shown in Figure 2. In the Pytorch framework, these functions are implemented as nn.BatchNorm1d/2d/3d, nn.LazyBatchNorm1d/2d/3d, nn.InstanceNorm1d/2d/3d, nn.GroupNorm, etc.

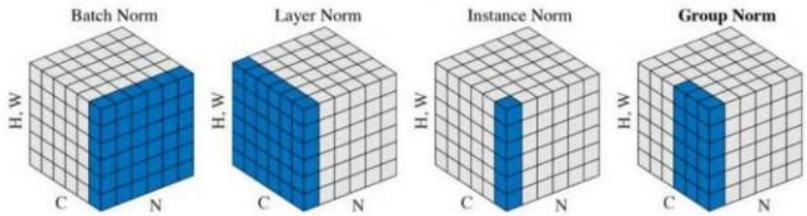



**Figure 2:** Four BN Modes

In YOLOv8, the nn.BatchNorm2d function is mainly used, and its calculation formula is as follows.

$$y = \frac{x - E[x]}{\sqrt{\mathrm{Var}[x] + \varepsilon}} \times \gamma + \beta \tag{2}$$

The function prototype and parameter description are as follows:

torch.nn.BatchNorm2d(num_features, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True, device=None, dtype=None)

**Table 2:** nn.BatchNorm2d Function Parameter Table

| Serial number | Parameter name | Explanation |
|---|---|---|
| 1 | num_features | Number of features |
| 2 | eps | Value added to the denominator for computational stability |
| 3 | momentum | Update parameter for mean and variance |
| 4 | affine | Whether affine transformation is needed for normalization. When it is True, coefficient matrices g$\gamma$ and $\beta$ need to be given. |
| 5 | track_running_stats | Whether the mean and variance of the model need to be updated |
| 6 | device | Computing device, usually CPU or GPU |
| 7 | dtype | Data type |

BatchNorm2d contains trainable parameters, and the parameter calculation formula is:

$$\text{Weight parameter} = \text{Input parameter} \times 2 \tag{3}$$

## 3. YOLOV8 MODEL ARCHITECTURE

According to the network architecture diagram of YOLOv8 drawn by RangeKing@github, its overall structure is

divided into three parts: Backbone, Neck, and Head. The Backbone is responsible for feature extraction and is mainly composed of three types of modules: Convolution Module (Con-Module), CSPLayer_2Conv, and SPPF. The Neck is mainly responsible for the fusion of multi-scale features, connecting the outputs of Stage Layer2/3/4 of the Backbone part through the Shotcut layer to enhance the feature representation ability. The Head part adopts a decoupled head structure (Decoupled-Head), which consists of a convolution module and Conv2d, and is responsible for calculating the losses of bounding box regression and classification tasks.

## 4. NETWORK STRUCTURE PARAMETER EXPLANATION

The network framework of YOLOv8n contains a total of 23 modules, 17 trainable parameter modules, 6 non-trainable parameters, and 225 layers of neural network. When the number of object detections is 80, the total number of parameters is 3,157,200, and the number of trainable parameters is 3,157,184.

According to the configuration file provided by the official, taking YOLOv8.yaml as an example, the file provides four types of configuration parameters: nc, scales, backbone, and head. The specific meanings are as follows:

> nc: Number of object detections

scales: Model depth and width scaling factors. The first column controls the model depth, the third column controls the model width, and the third column controls the maximum number of channels of the model. Among them, there are n → YOLOv8n model, s → YOLOv8s model, m → YOLOv8 m model, l → YOLOv8 l model, x → YO - LOv8x model.

backbone: Parameters of the backbone network skeleton, which explains the four types of data formats in the file.

**Table 3:** [-1,1,Conv,[64,3,2]] Data Format Description

| Serial number | Value (column) | Meaning |
|---|---|---|
| 1 | -1 | Default value, the number of input channels of this layer comes from the previous layer |
| 2 | 1 | Number of repetitions |
| 3 | Conv | Convolution module |
| 4 | [64, 3, 2] | [Number of output channels, kernel size, stride] |

**Table 4:** [-1,3,C2f,[128,True]] Data Format Description

| Serial number | Value (column) | Meaning |
|---|---|---|
| 1 | -1 | Default value, the number of input channels of this layer comes from the previous layer |
| 2 | 3 | Number of repetitions |
| 3 | C2f | Convolution module |
| 4 | [128, True] | [Number of output channels, whether to perform residual connection] |

**Table 5:** [-1, 1, SPPF, [1024, 5]] Data Format Explanation

| Serial number | Value (column) | Meaning |
|---|---|---|
| 1 | -1 | Default value, the number of input channels for this layer comes from the previous layer |
| 2 | 1 | Number of repetitions |
| 3 | SPPF | SPPF module |
| 4 | [1024, 5] | Convolution parameters |

Head: Components of the neural network in the head part. It consists of five components: nn.Upsample, Concat, C2f, Conv, and Detect. Among them, the parameter interpretations of C2f and Conv are the same as those in the Backbone. The interpretations of the other three are as follows.

**Table 6:** [-1, 1, nn.Upsample, [None, 2, "nearest"]] Data Format Explanation

| Serial number | Value (column) | Meaning |
|---|---|---|
| 1 | -1 | Default value, the number of input channels of this layer comes from the previous layer |
| 2 | 1 | Number of repetitions |
| 3 | nn.Upsample | Upsampling layer |
| 4 | [None, 2, "nearest"] | Convolution parameters |

**Table 7:** [[-1, 6], 1, Concat, ] Data Format Explanation

| Serial number | Value (column) | Meaning |
|---|---|---|
| 1 | [-1,6] | Receive data from the previous layer and the sixth layer |
| 2 | 1 | Number of repetitions |
| 3 | Concat | Connection module |
| 4 | | Dimension |

**Table 8:** [[15, 18, 21], 1, Detect, [nc]] Data Format Explanation

| Serial number | Value (column) | Meaning |
|---|---|---|
| 1 | [15, 18, 21] | Receive data from layers 15, 18, and 21 |
| 2 | 1 | Number of repetitions |
| 3 | Detect | Detection module |
| 4 | [nc] | Number of classifications |

## 5. TRAINING PARAMETER CALCULATION

The main modules of YOLOv8 include six types: Conv, C2f, SPPF, Upsample, Concat, and Detect, all of which inherit from the nn.Module class. Among them, the Upsample and Concat modules have no trainable parameters. The Conv module contains a two-dimensional convolutional layer, a batch normalization layer, and an activation function. The C2f module contains two Conv objects and a multi-layer Bottleneck structure, the number of layers of which corresponds to the repetition times in the backbone and head parameters. The Bottleneck module consists of two Conv objects and performs residual connection when the shotcut value is True and the input parameters c1 and c2 are equal. SPPF consists of two Conv objects and a two-dimensional max pooling layer. The detailed calculation methods of the trainable parameters for each layer are shown in Table 9.

**Table 9:** Statistical Table of Trainable Parameters for Each Layer of YOLOv8

| Number of layers | Module name | Sub-module | Calculation expression | Number of parameters in sub-module | Total quantity |
|---|---|---|---|---|---|
| 0 | conv.Conv | nn.Conv2d | $16 \times 32 \times 3 \times 3$ | 400 | 464 |
| | | BN | $32 \times 2$ | 64 | |
| 1 | conv.Conv | nn.Conv2d | $16 \times 32 \times 3 \times 3$ | 4608 | 4672 |
| | | BN | $32 \times 2$ | 64 | |
| 2 | block.C2f | cv1 | $32 \times 32 \times 1 \times 1 + 32 \times 2$ | 1088 | 7360 |
| | | cv2 | $(2 + 1) \times 16 \times 32 \times 1 \times 1 + 32 \times 2$ | 1600 | |
| | | Bottleneck | $1 \times 2 \times (16 \times 16 \times 3 \times 3+16\times2)$ | 4672 | |
| 3 | conv.Conv | nn.Conv2d | 32×64×3×3 | 18432 | 18560 |
| | | BN | 64×2 | 128 | |
| 4 | block.C2f | cv1 | 64×64×1×1+ 64×2 | 4224 | 49664 |
| | | cv2 | (2+2)×128×64× $1 \times 1 + 64 \times 2$ | 8320 | |
| | | Bottleneck | $2 \times 2 \times (32 \times 32 \times 3 \times 3+32\times2)$ | 37120 | |
| 5 | conv.Conv | nn.Conv2d | $64 \times 128 \times 3 \times 3$ | 73 728 | 73 984 |
| | | BN | $128 \times 128 \times 1 \times 1 +$ | 256 | |
| 6 | block.C2f | cv1 | $128 \times 128 \times 1 \times 1 + 128\times2$ | 16640 | 197 632 |
| | | cv2 | $(2 + 2) \times 64 \times 128 \times 1 \times 1 + 128 \times 2$ | 33 024 | |
| | | Bottleneck | $2 \times 2 \times (64 \times 64 \times 3 \times 3 + 64 \times 2)$ | 147 968 | |
| 7 | conv.Conv | nn.Conv2d | $128 \times 256 \times 3 \times 3 +$ | 294 912 | 295 424 |
| | | BN | $256 \times 2$ | 512 | |
| 8 | block.C2f | cv1 | $256 \times 256 \times 1 \times 1 + 256\times2$ | 66048 | 460 288 |
| | | cv2 | (2+1)×128× 256×1×1+ 256×2 | 98816 | |
| | | Bottleneck | $1 \times 2 \times (128 \times 128 \times 3 \times 3 + 128 \times 2)$ | 295 424 | |
| 9 | SPPF | cv1 | $256 \times 128 \times 1 \times 1 + 128 \times 2$ | 33024 | 164 608 |
| | | cv2 | $128 \times 4 \times 256 \times 1 \times 1 + 256 \times 2$ | 131 584 | |
| 10 | upsampling. Upsample | | | | 0 |
| 11 | conv.Concat | | | | 0 |
| 12 | block.C2f | cv1 | $384 \times 128 \times 1 \times 1 + 128 \times 2$ | 49408 | 148 224 |
| | | cv2 | $(2 + 1) \times 64 \times 128 \times 1 \times 1 + 128 \times 2$ | 24832 | |
| | | Bottleneck | $1 \times 2 \times (64 \times 64 \times 3 \times) 3+64\times2)$ | 73 984 | |
| 13 | upsampling. Upsample | | | | 0 |
| 14 | conv.Concat | | | | 0 |
| 15 | block.C2f | cv1 | 192×64×1×1+ $64 \times 2$ | 12416 | 37 248 |
| | | cv2 | $(2 + 1) \times 32 \times 64 \times 1 \times 1 + 64 \times 2$ | 6272 | |
| | | Bottleneck | $1\times2\times(32\times32\times3\times 3 + 32 \times 2)$ | 18 560 | |
| 16 | conv.Conv | nn.Conv2d | 64×64×3×3 | 36 864 | 36 992 |

| | | | | | |
|---|---|---|---|---|---|
| | | BN | 64×2 | 128 | |
| 17 | conv.Concat | | | | 0 |
| 18 | block.C2f | cv1 | 192×128×1×1+ 128×2 | 24 832 | 123 648 |
| | | cv2 | (2+1)×6 4×128× 1×1+128×2 | 24 832 | |
| | | Bottleneck | $1 \times 2 \times (64 \times 64 \times 3 \times 3+64\times2)$ | 73 984 | |
| 19 | conv.Conv | nn.Conv2d | $128 \times 128 \times 3 \times 3$ | 147 456 | 147712 |
| | | BN | $128 \times 2$ | 256 | |
| 20 | conv.Concat | | | | 0 |
| 21 | block.C2f | cv1 | $384 \times 256 \times 1 \times 1 + 256\times2$ | 98816 | 493 056 |
| | | cv2 | $(2 + 1) \times 128 \times 256 \times 1 \times 1 + 256 \times 2$ | 98816 | |
| | | Bottleneck | $1\times2\times(128\times128\times 3 \times 3 + 128 \times 2)$ | 295 424 | |
| 22 | head.Detect | cv2 | Omitted | 381888 | 897 664 |
| | | cv3 | Omitted | 515760 | |
| | | dfl | Omitted | 16 | |

## 6. SUMMARY

This article analyzes the neural network architecture of YOLOv8 in detail and introduces the calculation methods of the trainable parameters of 22 modules. By calculating the trainable parameters of each module, it provides guidance for the architecture optimization, parameter adjustment, and lightweight deployment of YOLO. As a typical representative of one-stage detection algorithms. The YOLO series of algorithms has been continuously developed and widely used in industrial fields such as object detection, image segmentation, pose estimation, and visual tracking. In the future, with the further development of technology, the YOLO algorithm will show great potential and value in more fields.

## REFERENCES


[1] Shi, C., Macdonald, G., Jalli, B., Lei, W., Zou, J., Jain, M., & Philip, J. (2026, May). Think less, label better: Multi-stage domain-grounded synthetic data generation for fine-tuning large language models in telecommunications. In ICC 2026-IEEE International Conference on Communications (pp. 1-6). IEEE.

[2] Junxi, Y., Wang, Z., & Chen, C. (2024). GCN-MF: A graph convolutional network based on matrix factorization for recommendation. Innovation & Technology Advances, 2(1), 14–26. https://doi.org/10.61187/ita.v2i1.30

[3] Hu, H., Zhang, J., & Sun, Y. (2024). The Multiscale Deep Neural Networks: Unveiling New Directions in Text Sentiment Analysis. Innovation & Technology Advances, 2(2), 34–45. https://doi.org/10.61187/ita.v2i2.65

[4] Shen, Z., Lin, S., Wang, Y., Saunders, E., & Dai, Y. (2026, May). Survival Risk Prediction Model for Colorectal Cancer Patients Based on Graph Convolutional Network and TCGA Multi-Omics Data Integration. In 2026 7th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT) (pp. 370-373). IEEE.

[5] Q. Tian, D. Zou, Y. Han and X. Li, "A Business Intelligence Innovative Approach to Ad Recall: Cross-Attention Multi-Task Learning for Digital Advertising," 2025 IEEE 6th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), Shenzhen, China, 2025, pp. 1249-1253, doi: 10.1109/AINIT65432.2025.11035473.

[6] Zhang, X. (2026, March). A Hybrid LSTM-GARCH Model Integrating Volatility Factors from the US Financial Markets. In Proceedings of the 2026 International Conference on AI Decision-Making and Management (pp. 183-189).

[7] Han, J., Zhong, P. & Sun, L. Dual-scale model collaborative reasoning with multi-feature fusion for robust AI-generated image detection. Multimedia Systems 32, 363 (2026). https://doi.org/10.1007/s00530-026-02425-4

[8] Liu, Y. (2025). The Deep Learning Paradigm for Plant Image Classification: A Systematic Evaluation of Architectural Efficacy. International Journal of Advance in Applied Science Research, 4(8), 73-79.

[9] Gao, W. (2026). Intelligent Operations and Public Relations Collaborative Management Model for Corporate Image Enhancement in the Smart Manufacturing Environment. Journal of Computer Technology and Applied Mathematics, 3(1), 28-34.

[10] Shen, Zepeng, et al. "Research on Application of Whale Optimization Algorithm in Financial Payment Fraud Detection." 2025 4th International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID). IEEE, 2025.

[11] Sun, L. (2025, November). Adaptive Interfaces for Personalized User Experience: A Machine Learning Approach. In Proceedings of the 2025 International Conference on Artificial Intelligence and Sustainable Development (pp. 457-462).
[12] Zhou, Z. (2026). Bottleneck Diagnosis in International Automotive Sales Funnels Using Gradient Boosting Trees: Evidence from Cross-Regional Team Efficiency Evaluation. Journal of Computer Technology and Applied Mathematics, 3(1), 11-18.