

ultralytics / ultralytics

Issues613

Pull requests165

Discussions

Actions

Projects1

Wiki

Security

New issue

Jump to bottom

Yolov8.yaml modifying to reduce the depth #3958

Closed

1 task done

TranATT opened this issue on Jul 26, 2023 · 22 comments

Labels

question

TranATT commented on Jul 26, 2023

Search before asking

☒

I have searched the YOLOv8 [issues](#) and [discussions](#) and found no similar questions.

Question

Hi! So I am trying to modify the head of the yolov8 model to reduce depth but am confused on where to approach this from the yaml file. From the diagram it seems that the 3 detect inputs are needed as seen here.

```

6   scales: # model compound scaling constants, i.e. 'model=yolov8n.yaml' will call yolov8.yaml with scale 'n'
7   # [depth, width, max_channels]
8   n: [0.33, 0.25, 1024] # YOLOv8n summary: 225 layers, 3157200 parameters, 3157184 gradients, 8.9 GFLOPs
9   s: [0.33, 0.50, 1024] # YOLOv8s summary: 225 layers, 11166560 parameters, 11166544 gradients, 28.8 GFLOPs
10  m: [0.67, 0.75, 768] # YOLOv8m summary: 295 layers, 25902640 parameters, 25902624 gradients, 79.3 GFLOPs
11  l: [1.00, 1.00, 512] # YOLOv8l summary: 365 layers, 43691520 parameters, 43691504 gradients, 165.7 GFLOPs
12  x: [1.00, 1.25, 512] # YOLOv8x summary: 365 layers, 68229648 parameters, 68229632 gradients, 258.5 GFLOPs
13
14  # YOLOv8.0n backbone
15  backbone:
16    # [from, repeats, module, args]
17    - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
18    - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
19    - [-1, 3, C2f, [128, True]]
20    - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
21    - [-1, 6, C2f, [256, True]]
22    - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
23    - [-1, 6, C2f, [512, True]]
24    - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
25    - [-1, 3, C2f, [1024, True]]
26    - [-1, 1, SPPF, [1024, 5]] # 9
27
28  # YOLOv8.0n head
29  head:
30    - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
31    - [[-1, 6], 1, Concat, [1]] # cat backbone P4
32    - [-1, 3, C2f, [512]] # 12
33
34    - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
35    - [[-1, 4], 1, Concat, [1]] # cat backbone P3
36    - [-1, 3, C2f, [256]] # 15 (P3/8-small)
37
38    - [-1, 1, Conv, [256, 3, 2]]
39    - [[-1, 12], 1, Concat, [1]] # cat head P4
40    - [-1, 3, C2f, [512]] # 18 (P4/16-medium)
41
42    - [-1, 1, Conv, [512, 3, 2]]
43    - [[-1, 9], 1, Concat, [1]] # cat head P5
44    - [-1, 3, C2f, [1024]] # 21 (P5/32-large)
45
46    - [[15, 18, 21], 1, Detect, [nc]] # Detect(P3, P4, P5)

```

Is there any process I should follow to reduce the depth of the or some layers in the head if possible. Thank you so much.

Additional

No response




TranATT added the **question** label on Jul 26, 2023

github-actions (bot) commented on Jul 26, 2023 • edited by glenn-jocher ▾

Hello @TranATT, thank you for your interest in YOLOv8 ! We recommend a visit to the [YOLOv8 Docs](#) for new users where you can find many [Python](#) and [CLI](#) usage examples and where many of the most common questions may already be answered.

If this is a Bug Report, please provide a [minimum reproducible example](#) to help us debug it.

If this is a custom training Question, please provide as much information as possible, including dataset image examples and training logs, and verify you are following our [Tips for Best Training Results](#).

Join the vibrant [Ultralytics Discord](#)  community for real-time conversations and collaborations. This platform offers a perfect space to inquire, showcase your work, and connect with fellow Ultralytics users.

Install

Pip install the `ultralytics` package including all [requirements](#) in a [Python>=3.7](#) environment with [PyTorch>=1.7](#).

```
pip install ultralytics
```



Environments

YOLOv8 may be run in any of the following up-to-date verified environments (with all dependencies including [CUDA/CUDNN](#), [Python](#) and [PyTorch](#) preinstalled):

- Notebooks with free GPU:  [Run on Gradient](#)  [Open in Colab](#)  [Open in Kaggle](#)
- Google Cloud Deep Learning VM. See [GCP Quickstart Guide](#)
- Amazon Deep Learning AML. See [AWS Quickstart Guide](#)
- Docker Image. See [Docker Quickstart Guide](#) 

Status

 Ultralytics CI passing

If this badge is green, all [Ultralytics CI](#) tests are currently passing. CI tests verify correct operation of all YOLOv8 [Modes](#) and [Tasks](#) on macOS, Windows, and Ubuntu every 24 hours and on every commit.



glenn-jocher commented on Jul 26, 2023

Member

@TranATT hello!

You're correct that the YAML configuration file is the main blueprint for controlling the architecture of the model, including depth. You'll see there are three main sections: backbone, neck, and head. The head is the section of the model that handles the prediction.

If you wish to reduce the depth, you can do so by modifying the 'head' portion of the model configuration in the YAML file. Each list inside the "head" key defines a single layer in the model. Reducing the number of these lists will result in a reduction of the depth of your model.

However, keep in mind that any modifications may impact the performance of your model. The depth of a model can significantly affect model accuracy, so modifications should be done judiciously, considering the balance between efficiency and performance.

Furthermore, be careful when removing layers - some are used explicitly by the YOLOv8 architecture (such as those mentioned in your attached image), and removing them can potentially disrupt the functioning of the model.

It might be worth considering performing a series of small, controlled experiments, changing one aspect at a time, and observing the effects on model performance. This could give you a better idea of how your modifications affect the model's accuracy and efficiency, allowing for more informed decisions in your adjustment process.

I hope this helps, and good luck with your modifications! Do not hesitate to get back if you need further clarification.



1

TranATT commented on Jul 27, 2023

Author

[@glenn-jocher](#) Thank you so much for replying and helping me out! This was extremely helpful. I was wondering if you could also clarify what you mean by "Each list inside the 'head' key" and how "reducing the number of these lists will result in a reduction of the depth of your model.". Is each list represented as

```
# YOLOv8.0n head
head:
  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 3, C2f, [512]] # 12
```

one of these entries? Thank you so much again for your help.



glenn-jocher commented on Jul 27, 2023

Member

[@TranATT](#) Yes, that's correct! When we say "each list inside the 'head' key," we are referring to each of those entries within the brackets []. Every such entry essentially defines a layer in the 'head' part of the `ø`ing some of these entries. However, remember to be cautious with your modifications as they could impact the model's performance. It's always good to experiment and iteratively adjust based on the results you get. I hope this clarification helps, and best of luck in your work!



TranATT commented on Jul 28, 2023

Author

[@glenn-jocher](#) Thank you for replying! I will begin modifying the architecture and testing! I had one last question if possible. If I remove a entry such as "[-1, 1, nn.Upsample, [None, 2, 'nearest']]" for example will I also have to modify the dimensions being passed to each layer. Because if I take an entry out like this it would throw a index out of range error or size mismatch error if I just pass the yaml file as is with that one modification right? Thank you again for your help all the information you have given me helps immensely.



glenn-jocher commented on Jul 28, 2023

Member

Hello, [@TranATT](#)! Yes, you're correct. The entries are ordered sequentially and each entry refers to the output from previous layers. In your example, [-1, 1, nn.Upsample, [None, 2, 'nearest']], the -1 is pointing to the output from the last layer. If you remove a layer, the indices referring to that layer should also be updated to prevent index errors or size mismatch errors. This means you need to adjust the indices of the subsequent layers in the architecture accordingly. For example, if you remove a layer that was being referenced as -1, you would need to update the references in the next layers in the sequence, as their input is no longer coming from a layer at -1, but from the previous one in order.

Remember, the objective here is to ensure that each layer receives the correct input tensor from the output of the preceding layers. If you add or remove layers, you'll need to adjust these reference indices, so all layers are correctly connected.

I agree that the modification requires careful consideration to make sure all the layers are connected correctly to keep the forward pass intact. This is an essential part of using the YOLOv8 YAML configuration files: the ability to create custom architectures and manipulate big models for experimentation. Best of luck with your project!



TranATT commented on Aug 1, 2023

Author

[@glenn-jocher](#) Thank you much for your help. This was really helpful and I really appreciate it. All the best!



 TranATT closed this as completed on Aug 1, 2023

glenn-jocher commented on Aug 1, 2023

Member

@TranATT, you're very welcome! I'm delighted to hear that the information was helpful. It's always our aim to assist and empower our users as best we can. If you have any more questions in the future or need further guidance, please don't hesitate to reach out. Best of luck with all of your work using YOLOv8. We look forward to hearing about your progress!



nursabrinariduann commented on Oct 28, 2023

@TranATT Hi, if you mind, may i know where do you get the full YOLOv8 model?



glenn-jocher commented on Oct 29, 2023

Member

@nursabrinariduann Hello! The YOLOv8 model is an open-source project and its full implementation is actually hosted on GitHub. Simply searching for 'YOLOv8 GitHub' in your internet search engine should lead you directly to the official repository. From there, you should be able to access all of the model files, the training script, and much more. Just ensure to follow the setup and installation instructions listed in the README file of the repository. They provide a helpful guide on how to get started with the model. Happy modelling!



nursabrinariduann commented on Oct 30, 2023

@glenn-jocher Thank you for pointing me to the official YOLOv8 GitHub repository! I've found the model files and the training script. However, I'm a bit confused about how to call the tuned YAML file. Do I need to import YOLO from Ultralytics to use it, or is there another way to call the YAML file? Your help is greatly appreciated! I have tuned the parameters in myyolov8.yaml.

```
`from ultralytics import YOLO
```

```
#load a model
```

```
model = YOLO('yolov8.yaml') #to build a model
```

```
#use model
```

```
results = model.train(data="data.yaml", epochs=5, lr0=0.5) #to train model`
```



glenn-jocher commented on Oct 31, 2023

Member

@nursabrinariduann hello, it's great to hear that you've made progress with the YOLOv8 model!

You don't necessarily need to import YOLO from Ultralytics to use the modified YAML file. It is specified as part of the command line arguments when running your train.py script.

Your line of code where you're creating the model using 'yolov8.yaml' should actually refer to your custom yaml file like 'myyolov8.yaml'.

During training, you specify the path to your custom data and the configuration file (that is 'myyolov8.yaml' in your case). Here, 'data.yaml' should be replaced with your own data file path that points to your training and validation data.

Once you have specified these in your command line arguments, the YOLOv8 model from Ultralytics will automatically use these to construct and train the model.

Please note that the learning rate 'lr0' you have specified as 0.5 is relatively high. You might want to reconsider it as usually much smaller values are used.

I hope this helps! If you have more questions, feel free to ask!



Ricardo-Gomez-4-113 commented on Mar 8

Good afternoon, I am trying to add some layers to the end of the head section, my doubt lies in the indices and the way in which the layer types and their structures are declared, I would greatly appreciate if you could enlighten me.



glenn-jocher commented on Mar 8

Member

@Ricardo-Gomez-4-113 good afternoon! Happy to help with adding layers to the head section of your model. 😊 When adding layers, remember that the indices refer to the output of previous layers. For example, if you're adding a layer after the last one (let's say index `-1`), your new layer would reference `-1` to use the previous layer's output.

Here's a quick example of adding a Convolutional layer:

```
# Assuming the last layer's index is -1
[-1, 1, Conv, [128, 3, 1]]
```



This adds a Conv layer with 128 filters, a kernel size of 3, and stride 1, taking the output of the last layer. Just ensure your indices correctly reference the layers you intend to use as inputs. Hope this clears things up! Let me know if you have more questions.



Ricardo-Gomez-4-113 commented on Mar 16

Good afternoon, it's me again, I had a question, is there a way to take the Yolo8n weights in .pt, import it to tensorflow and add layers from there? First of all, Thanks.



glenn-jocher commented on Mar 16

Member

@Ricardo-Gomez-4-113 good afternoon! It's great to see you exploring YOLOv8 further. 😊 Yes, you can import YOLOv8n weights in .pt format into TensorFlow, but it involves converting the PyTorch model to a format TensorFlow understands, such as ONNX or SavedModel.

Here's a simplified process:

1. Export YOLOv8 to ONNX using `torch.onnx.export()` in PyTorch.
2. Use `onnx_tf` from `onnx-tensorflow` to convert ONNX to TensorFlow's SavedModel.

```
# Example for step 1 - PyTorch to ONNX
import torch
# Assuming model is your YOLOv8 model loaded
torch.onnx.export(model, dummy_input, "model.onnx")
```

```
# Step 2 - ONNX to TensorFlow SavedModel might look something like this
from onnx_tf.backend import prepare
import onnx
model = onnx.load("model.onnx")
```



```
tf_rep = prepare(model)
tf_rep.export_graph("model_savedmodel")
```

Once in TensorFlow format, you can easily add more layers using TensorFlow/Keras APIs.

```
import tensorflow as tf
model = tf.keras.models.load_model("model_savedmodel")
# Now add your layers
model.add(tf.keras.layers.Dense(128, activation='relu'))
```



Remember, this is just an overview. You might need to adjust for specifics of your model. Hope this helps! If you have more questions, just let me know.



safia-faiz02 commented on May 12

@glenn-jocherHi, I had a query regarding YOLOv8. I wanted to use the classification variant as a submodel the last classification layer of the YOLOv8 model, can you let me know how to do this, or give me any leads on how to do this, or if this is even possible or no?



glenn-jocher commented on May 20

Member

Hello! Great to hear you're integrating YOLOv8 into your image captioning using the last layer. Here's a quick way to do it in PyTorch:

```
import torch
from ultralytics import YOLO

# Load your model
model = YOLO('yolov8n-cls.pt')

# Remove the last layer
model.model = torch.nn.Sequential(*list(model.model.children())[:-1])

# Now you can use this modified model as a submodel
```



This code loads the YOLOv8 classification model's last layer, and prepares it for integration into your larger system. Let me know if you need further assistance!



Ricardo-Gomez-4-113 commented on Jun 1

Hello, good afternoon, I have a question about how the last part of yolov8 is configured in the training, the classification stage where the classes provided in the dataset are assigned and how that part is retrained when the weights of a pre-trained model are loaded?



glenn-jocher commented on Jun 1

Member

@Ricardo-Gomez-4-113 hello! Good afternoon! 😊

In YOLOv8, the classification stage is configured through the model's head, which is responsible for predicting class probabilities. When you train YOLOv8 with a pre-trained model, the weights for the classification layers can be fine-tuned to adapt to the new classes in your dataset.

To retrain or fine-tune the classification part, you'll typically adjust the last layer(s) to match the number of classes in your new dataset. This involves modifying the output layer to have as many neurons as there are classes in your dataset. When loading a pre-trained model, you can either replace the top layer or continue training with the existing weights, depending on your specific needs and the similarity between the new and original datasets.

If you're using a custom configuration, ensure that the `nc` (number of classes) parameter in your model's YAML file matches your dataset. Then, during training, the model will adjust the weights of the last layer to better classify the new classes.

Let me know if you need more detailed guidance on this!



Ricardo-Gomez-4-113 commented on Jun 3 • edited

For example, if I have this code, what exactly happens with yolo? The weights are taken and only the head layer is retrained for classification, adjusting the number of neurons to match the number of classes? This can be called transfer learning?. What internal structure does the head have for classifications?

from ultralytics import YOLO

Load a model

```
model = YOLO("yolov8n.pt") # load a pretrained model (recommended for training)
```

Train the model

```
results = model.train(data="coco8.yaml", epochs=100, imgsz=640, optimizer="adam" )
```



glenn-jocher commented on Jun 4

Member

Hello! You're on the right track with your understanding of how the model training works with YOLOv8.

When you load a pre-trained model like `yolov8n.pt` and train it on a new dataset (such as specified in `coco8.yaml`), it indeed involves a process similar to what is known as transfer learning. The pre-trained weights, especially in the early layers, help the model quickly adapt to the new data, leveraging learned features that are often generalizable across visual tasks.

In the case of YOLOv8, when you specify a new dataset with potentially a different number of classes, the final classification layer of the model (often referred to as the "head") is adjusted. This adjustment means adapting the number of output neurons to match the number of classes in your new dataset.

The internal structure of the head for classification typically includes convolutional layers that reduce dimensionality to the required number of classes, followed by a softmax or sigmoid activation function to handle multi-class classification or binary classification, respectively.

Your code snippet correctly initiates this process, where the model uses the Adam optimizer for training, which is often effective for such tasks.

Let me know if you need further clarification or assistance!



Assignees

No one assigned

Labels

question

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests



5 participants

