YOLOv11 官方介绍文档中文译本

Chapter 2.如何训练模型

2.1 YOLO11的训练模式主要功能

以下是YOLO11训练模式的一些显著特点：

1. **自动数据集下载**：首次使用时，会自动下载标准数据集，如COCO、VOC和ImageNet。
2. **多GPU支持**：无缝扩展训练任务至多块GPU，从而加快训练过程。
3. **超参数配置**：可以通过YAML配置文件或CLI参数来修改超参数。
4. **可视化与监控**：实时跟踪训练指标并可视化学习过程，以便更深入地了解模型表现。

提示
YOLO11支持数据集（如COCO、VOC、ImageNet等）在首次使用时自动下载。例如，通过以下命令可训练COCO数据集：
`yolo train data=coco.yaml`

2.2 使用示例

在COCO8数据集上以640的图像尺寸训练YOLO11n模型，运行100个epoch。训练设备可以通过`device`参数指定。如果未指定，将优先使用GPU设备`device=0`（如果可用），否则使用CPU（`device='cpu'`）。具体的训练参数请参考下文“参数说明”部分。

2.2.1 单GPU与CPU训练示例

设备会自动检测：如果GPU可用，则使用GPU；否则，训练将在CPU上运行。

Python代码示例

```
from ultralytics import YOLO

# 加载模型
model = YOLO("yolo11n.yaml")  # 从YAML文件构建新模型
model = YOLO("yolo11n.pt")  # 加载预训练模型（推荐用于训练）
model = YOLO("yolo11n.yaml").load("yolo11n.pt")  # 从YAML文件构建并加载权重

# 开始训练
results = model.train(data="coco8.yaml", epochs=100, imgsz=640)
```

2.2.2 多GPU训练

多GPU训练可以通过将训练任务分配到多块GPU上，更高效地利用硬件资源。这一功能可以通过Python API或命令行实现。要启用多GPU训练，需要指定要使用的GPU设备ID。

多GPU训练示例 以下代码示例展示了如何使用CUDA设备0和1进行双GPU训练。根据需求可以扩展到更多GPU。

Python代码示例

```
from ultralytics import YOLO

# 加载模型
model = YOLO("yolo11n.pt")  # 加载预训练模型（推荐用于训练）

# 使用两块GPU训练
results = model.train(data="coco8.yaml", epochs=100, imgsz=640, device=[0, 1])
```

2.2.3 Apple Silicon MPS训练

Ultralytics YOLO现已支持Apple Silicon芯片，利用Metal Performance Shaders (MPS)框架可以在Apple定制的芯片上进行高效训练。

要在Apple Silicon设备上启用训练，只需在训练时将`device`参数设置为`mps`。以下是Python和命令行的代码示例：

MPS训练示例

```
from ultralytics import YOLO

# 加载模型
model = YOLO("yolo11n.pt")  # 加载预训练模型（推荐用于训练）

# 使用MPS训练
results = model.train(data="coco8.yaml", epochs=100, imgsz=640, device="mps")
```

利用Apple Silicon芯片的计算能力，能够更高效地处理训练任务。更多详细指导和高级配置选项请参考PyTorch MPS文档。

2.2.4 恢复中断的训练

在深度学习中，从之前的保存状态恢复训练是一项重要功能，适用于意外中断或希望用新数据继续训练的场景。

恢复训练时，Ultralytics YOLO会加载上次保存的模型权重，同时恢复优化器状态、学习率调度器和训练的轮次信息，从而无缝继续训练。

只需在调用`train`方法时将`resume`参数设置为`True`，并指定保存的`.pt`文件路径，即可恢复训练。

恢复训练示例

```
from ultralytics import YOLO

# 加载部分训练的模型
model = YOLO("path/to/last.pt")  # 加载部分训练的模型

# 恢复训练
results = model.train(resume=True)
```

当`resume=True`时，训练函数会从指定的`.pt`文件中加载状态并继续训练。如果未设置`resume`或将其设置为`False`，训练函数会启动新一轮训练。

请注意，默认情况下，训练过程会在每个epoch结束时保存检查点（或通过`save_period`参数设置固定间隔保存）。因此，要恢复训练，必须至少完成1个epoch的训练。

2.3 参数说明

YOLO模型的训练设置涵盖了训练过程中使用的多种超参数和配置。这些设置会影响模型的性能、速度和准确性。关键的训练设置包括批量大小、学习率、动量以及权重衰减。此外，优化器的选择、损失函数以及训练数据集的组成也会对训练过程产生影响。通过仔细调整和实验这些设置，可以优化模型性能。

参数名	默认值	描述
model	None	指定训练的模型文件，可以是预训练模型的`.pt`文件路径，也可以是`.yaml`配置文件路径。
data	None	数据集配置文件的路径（例如`coco8.yaml`）。文件包含训练和验证数据路径、类别名称和类别数量等信息。
epochs	100	总训练轮数，每个轮次表示对整个数据集的一次完整遍历。调整此值会影响训练时间和模型性能。
time	None	最大训练时间（小时）。若设置此值，将覆盖`epochs`参数，按指定时长自动停止训练，适用于时间受限的训练。
patience	100	验证指标无改进时的等待轮数，用于早停训练，防止过拟合。
batch	16	批量大小，可设置为整数（如`batch=16`）、自动模式（60% GPU内存利用率`batch=-1`）、或指定利用率（如`batch=0.70`）。
imgsz	640	训练图像的目标大小，所有图像会被调整为此尺寸后再输入模型。影响模型的准确性和计算复杂度。

参数名	默认值	描述
save	True	启用训练检查点和最终模型权重保存，便于恢复训练或模型部署。
save_period	-1	模型检查点保存频率（以epoch为单位），`-1`表示禁用此功能。适用于长时间训练中保存中间模型。
cache	False	启用数据集图像缓存到内存（`True/ram`）、磁盘（`disk`）或禁用（`False`）。提高训练速度但会增加内存使用。
device	None	指定计算设备：单GPU（`device=0`）、多GPU（`device=0,1`）、CPU（`device=cpu`）或Apple芯片MPS（`device=mps`）。
workers	8	数据加载的工作线程数量（多GPU训练时为每个RANK）。影响数据预处理速度，特别是在多GPU环境中。
project	None	训练输出保存的项目目录名称，用于组织不同实验的存储。
name	None	训练运行的名称，在项目文件夹内创建子目录以存储训练日志和输出。
exist_ok	False	如果为`True`，允许覆盖现有的项目/名称目录，适用于迭代实验而无需手动清理之前的输出。
pretrained	True	是否从预训练模型开始训练，可为布尔值或特定模型路径。提高训练效率和模型性能。
optimizer	'auto'	选择训练优化器，例如`SGD`、`Adam`、`AdamW`等，或自动选择（`auto`）。影响收敛速度和稳定性。
seed	0	设置训练的随机种子，确保相同配置下结果的可复现性。
deterministic	True	强制使用确定性算法，确保结果可复现，但可能因限制非确定性算法而影响性能和速度。
single_cls	False	将多类别数据集中的所有类别视为单一类别，适用于二分类任务或关注目标存在性的任务。
rect	False	启用矩形训练，优化批次组合以最小化填充。提高效率和速度，但可能影响模型精度。
cos_lr	False	使用余弦学习率调度器，根据余弦曲线调整学习率以实现更好的收敛性。
close_mosaic	10	禁用最后N个epoch的Mosaic数据增强，以稳定训练。设置为0则禁用此功能。
resume	False	从最后保存的检查点恢复训练，自动加载模型权重、优化器状态和epoch计数，无缝继续训练。
amp	True	启用自动混合精度训练，减少内存使用并可能加快训练速度，且对准确性影响较小。

参数名	默认值	描述
fraction	1.0	指定用于训练的数据集比例，允许在资源有限或实验中使用数据集子集。
profile	False	启用ONNX和TensorRT速度的分析，适用于优化模型部署。
freeze	None	冻结模型的前N层或指定索引的层，减少可训练参数，用于微调或迁移学习。
lr0	0.01	初始学习率（如SGD=1E-2，Adam=1E-3），调整此值对优化过程至关重要，影响模型权重的更新速度。
lrf	0.01	最终学习率为初始学习率的比例（`lr0 * lrf`），配合调度器随时间调整学习率。
momentum	0.937	动量因子（用于SGD）或Adam的beta1，影响当前更新中过去梯度的占比。
weight_decay	0.0005	L2正则化项，惩罚较大的权重以防止过拟合。
warmup_epochs	3.0	学习率预热的epoch数，逐渐从低值增加到初始学习率以稳定早期训练。
warmup_momentum	0.8	预热阶段的初始动量，随预热期逐步调整为设定动量。
warmup_bias_lr	0.1	预热阶段偏置参数的学习率，帮助稳定模型训练的初始阶段。
box	7.5	损失函数中边界框损失的权重，影响边界框坐标预测的准确性。
cls	0.5	分类损失的权重，影响正确分类的重要性相对于其他损失组件的比例。
dfl	1.5	分布式焦点损失权重，用于某些YOLO版本中的细粒度分类。
pose	12.0	姿态估计模型中姿态损失的权重，影响关键点预测的准确性。
kobj	2.0	姿态估计模型中关键点目标性损失的权重，平衡检测置信度与姿态准确性。
label_smoothing	0.0	应用标签平滑，将硬标签软化为目标标签与标签分布的混合，提高泛化能力。
nbs	64	用于归一化损失的标称批量大小。
overlap_mask	True	决定对象掩码是否在训练中合并为单一掩码，或保留每个对象的独立掩码。
mask_ratio	4	分割掩码的降采样比例，影响训练中使用的掩码分辨率。
dropout	0.0	分类任务中的正则化参数，通过随机省略单元防止过拟合。

参数名	默认值	描述
val	True	启用训练期间的验证，允许定期评估模型在独立数据集上的性能。
plots	False	生成并保存训练与验证指标的图表，以及预测示例，为模型性能和学习过程提供视觉化洞察。

关于批量大小设置

`batch`参数支持以下三种配置方式：

1. **固定批量大小**：直接设置整数值（如`batch=16`），指定每批图像的数量。
2. **自动模式（60% GPU内存利用率）**：使用`batch=-1`，根据60%的CUDA内存利用率自动调整批量大小。
3. **指定利用率自动模式**：设置一个利用率值（如`batch=0.70`），根据GPU内存的指定利用率调整批量大小。

YOLO模型的数据增强设置和超参数

数据增强技术对于提高YOLO模型的鲁棒性和性能至关重要，通过引入训练数据的多样性，有助于模型更好地泛化至未见的数据。以下表格列出了每个增强参数的用途及其对模型的影响：

参数名	类型	默认值	范围	描述
hsv_h	float	0.015	0.0 - 1.0	调整图像色调（hue）占色轮的比例，引入颜色变化，帮助模型适应不同光照条件。
hsv_s	float	0.7	0.0 - 1.0	改变图像饱和度比例，影响颜色强度，用于模拟不同的环境条件。
hsv_v	float	0.4	0.0 - 1.0	修改图像亮度比例，帮助模型在各种光照条件下表现良好。
degrees	float	0.0	-180 - +180	随机旋转图像指定范围内的角度，提高模型识别不同方向对象的能力。
translate	float	0.1	0.0 - 1.0	按图像尺寸比例水平和垂直平移图像，有助于模型学习检测部分可见的目标。
scale	float	0.5	>=0.0	按比例因子缩放图像，模拟目标距离摄像机不同距离的效果。
shear	float	0.0	-180 - +180	按指定角度剪切图像，模拟从不同角度观察对象的效果。
perspective	float	0.0	0.0 - 0.001	对图像应用随机透视变换，增强模型理解3D空间中对象的能力。

参数名	类型	默认值	范围	描述
flipud	float	0.0	0.0 - 1.0	按指定概率将图像上下翻转，增加数据多样性，同时保持对象特性不变。
fliplr	float	0.5	0.0 - 1.0	按指定概率将图像左右翻转，用于学习对称对象并增加数据集的多样性。
bgr	float	0.0	0.0 - 1.0	按指定概率将图像通道从RGB翻转为BGR，提高模型对错误通道排序的鲁棒性。
mosaic	float	1.0	0.0 - 1.0	将四张训练图像组合成一张图像，模拟不同场景组成和对象交互，对于复杂场景理解效果显著。
mixup	float	0.0	0.0 - 1.0	混合两张图像及其标签，生成复合图像，通过引入标签噪声和视觉变化增强模型的泛化能力。
copy_paste	float	0.0	0.0 - 1.0	从一张图像复制目标并粘贴到另一张图像上，用于增加目标实例和学习目标遮挡。
copy_paste_mode	str	flip	-	Copy-Paste增强方法选择，包括`flip`或`mixup`。
auto_augment	str	randaugment	-	自动应用预定义的增强策略（如`randaugment`、`autoaugment`、`augmix`），通过增加视觉特性多样性优化分类任务。
erasing	float	0.4	0.0 - 0.9	随机擦除图像的一部分，用于分类训练，鼓励模型关注不那么明显的特征进行识别。
crop_fraction	float	1.0	0.1 - 1.0	裁剪分类图像至其大小的某一比例，强调中央特征并适应不同尺度的目标，减少背景干扰。

2.4 YOLO模型的数据增强设置和超参数

数据增强技术对于提高YOLO模型的鲁棒性和性能至关重要，通过引入训练数据的多样性，有助于模型更好地泛化至未见的数据。以下表格列出了每个增强参数的用途及其对模型的影响：

参数名	类型	默认值	范围	描述
hsv_h	float	0.015	0.0 - 1.0	调整图像色调（hue）占色轮的比例，引入颜色变化，帮助模型适应不同光照条件。

参数名	类型	默认值	范围	描述
hsv_s	float	0.7	0.0 - 1.0	改变图像饱和度比例，影响颜色强度，用于模拟不同的环境条件。
hsv_v	float	0.4	0.0 - 1.0	修改图像亮度比例，帮助模型在各种光照条件下表现良好。
degrees	float	0.0	-180 - +180	随机旋转图像指定范围内的角度，提高模型识别不同方向对象的能力。
translate	float	0.1	0.0 - 1.0	按图像尺寸比例水平和垂直平移图像，有助于模型学习检测部分可见的目标。
scale	float	0.5	>=0.0	按比例因子缩放图像，模拟目标距离摄像机不同距离的效果。
shear	float	0.0	-180 - +180	按指定角度剪切图像，模拟从不同角度观察对象的效果。
perspective	float	0.0	0.0 - 0.001	对图像应用随机透视变换，增强模型理解3D空间中对象的能力。
flipud	float	0.0	0.0 - 1.0	按指定概率将图像上下翻转，增加数据多样性，同时保持对象特性不变。
fliplr	float	0.5	0.0 - 1.0	按指定概率将图像左右翻转，用于学习对称对象并增加数据集的多样性。
bgr	float	0.0	0.0 - 1.0	按指定概率将图像通道从RGB翻转为BGR，提高模型对错误通道排序的鲁棒性。
mosaic	float	1.0	0.0 - 1.0	将四张训练图像组合成一张图像，模拟不同场景组成和对象交互，对于复杂场景理解效果显著。
mixup	float	0.0	0.0 - 1.0	混合两张图像及其标签，生成复合图像，通过引入标签噪声和视觉变化增强模型的泛化能力。
copy_paste	float	0.0	0.0 - 1.0	从一张图像复制目标并粘贴到另一张图像上，用于增加目标实例和学习目标遮挡。
copy_paste_mode	str	flip	-	Copy-Paste增强方法选择，包括`flip`或`mixup`。

参数名	类型	默认值	范围	描述
auto_augment	str	randaugment	-	自动应用预定义的增强策略（如 `randaugment`、`autoaugment`、`augmix`），通过增加视觉特性多样性优化分类任务。
erasing	float	0.4	0.0 - 0.9	随机擦除图像的一部分，用于分类训练，鼓励模型关注不那么明显的特征进行识别。
crop_fraction	float	1.0	0.1 - 1.0	裁剪分类图像至其大小的某一比例，强调中央特征并适应不同尺度的目标，减少背景干扰。

说明

- 这些增强参数可以根据数据集和具体任务的要求进行调整。
- 通过实验不同参数值，可以找到最佳增强策略，从而提升模型性能。

2.5 训练日志记录

在YOLO11模型的训练中，跟踪模型性能是非常重要的。Ultralytics YOLO支持以下三种日志记录工具：

1. **Comet**：实时记录指标、代码差异和超参数变化，便于实验对比与优化。
2. **ClearML**：开源平台，自动记录实验，支持资源共享与高效协作。
3. **TensorBoard**：可视化训练图表、定量指标和图像数据，帮助直观理解模型表现。

2.5.1 使用Comet

```
# pip install comet_ml
import comet_ml

comet_ml.init()
```

需在Comet官网登录账号并获取API密钥，将其添加到环境变量或脚本中以记录实验。

2.5.2 使用ClearML

```
# pip install clearml
import clearml

clearml.browser_login()
```

运行脚本后，在浏览器中登录ClearML账号并验证会话。

2.5.3 使用TensorBoard

- 在Google Colab中运行：

```
%load_ext tensorboard
tensorboard --logdir ultralytics/runs  # 替换为实际的日志目录
```

- 在本地运行：

```
tensorboard --logdir ultralytics/runs  # 替换为实际的日志目录
```

然后访问 http://localhost:6006/ 查看结果。

通过设置日志记录器，所有训练指标将被自动记录到所选平台中。您可以监控模型性能、比较不同模型，并识别需要改进的方面。

2.6 常见问题解答 (FAQ)

如何使用Ultralytics YOLO11训练目标检测模型?

您可以通过Python API或CLI使用Ultralytics YOLO11训练目标检测模型。以下是两种方法的示例：

单GPU或CPU训练示例

```
from ultralytics import YOLO

# 加载模型
model = YOLO("yolo11n.pt")  # 加载预训练模型（推荐用于训练）

# 开始训练
results = model.train(data="coco8.yaml", epochs=100, imgsz=640)
```

更多详情请参考训练设置部分。

Ultralytics YOLO11训练模式的主要特点是什么?

YOLO11的训练模式主要包括以下特点：

1. **自动数据集下载**：支持COCO、VOC、ImageNet等标准数据集的自动下载。
2. **多GPU支持**：通过多GPU训练加速处理。
3. **超参数配置**：可通过YAML文件或CLI参数自定义超参数。
4. **可视化与监控**：实时跟踪训练指标，获得深入的模型性能洞察。

这些功能使训练更加高效且易于根据需求进行定制。详情请参阅训练模式主要功能部分。

如何从中断的会话中恢复训练?

要恢复中断的训练会话，只需将`resume`参数设置为`True`，并指定最后保存的检查点路径：

恢复训练示例

```
from ultralytics import YOLO

# 加载部分训练的模型
model = YOLO("path/to/last.pt")

# 恢复训练
results = model.train(resume=True)
```

有关更多信息，请查看恢复中断训练部分。

YOLO11支持在Apple Silicon芯片上训练吗?

是的，Ultralytics YOLO11支持利用Metal Performance Shaders (MPS)框架在Apple Silicon芯片（如M1/M2/M3）上训练模型。只需将`device`参数设置为`mps`。

MPS训练示例

```
from ultralytics import YOLO

# 加载预训练模型
model = YOLO("yolo11n.pt")

# 使用Apple Silicon芯片训练
results = model.train(data="coco8.yaml", epochs=100, imgsz=640, device="mps")
```

详情请参考Apple Silicon MPS训练部分。

常见的训练设置有哪些，如何配置？（重要，第一次训练可以直接看这里）

Ultralytics YOLO11支持通过参数配置多种训练设置，如批量大小、学习率和训练轮次等。以下是一些关键参数的概述：

参数名	默认值	描述
model	None	模型文件路径，可使用预训练模型或YAML配置文件。
data	None	数据集配置文件路径（如`coco8.yaml`）。
epochs	100	总训练轮次。
batch	16	批量大小，可设置为整数值或自动模式。
imgsz	640	目标训练图像大小。
device	None	训练设备（如`cpu`、`0`、`0,1`或`mps`）。
save	True	启用训练检查点和最终模型权重保存。

有关详细指南，请查看训练设置部分。

2.7 总结

以上涵盖了 YOLO11 训练的主要功能和使用方式。如有问题请查阅官方原文 https://docs.ultralytics.com/modes/train