期末复习大纲

一、单项选择题（每小题 2 分，本题共 40 分）

TensorFlow语法基础与构建模型过程

```
模型构建过程：
用TensorflowAPI：tf.keras搭建网络八股
import
train, test
model = tf.keras.models.Sequential
model.compile
model.fit
model.summary3
（第三个ppt）
```

语法：（与ppt一样）

[tensorflow基础语法讲解_tensorflow 2.13.1语法讲解-CSDN博客]

图像分类/目标检测项目的基本流程

1. 数据集收集与清洗
2. 数据分析与可视化
3. 模型训练
4. 模型评估及指标
5. 模型部署

图像分类减少过拟合的方法

- 数据清洗
- 增大训练集
- 采用正则化
- 增大正则化参数

labelme文件格式

```
labelme文件格式是一种常用的图像标注数据格式，用于存储图像及其对应的标签信息。该格式通常由两个文件组成：一个是图像文件，另一个是标签文件。

图像文件可以是任意格式的图像文件，例如JPEG、PNG等。标签文件是一个JSON格式的文件，其中包含了图像的标签信息。
---------------------------------------
标签文件的内容包括以下几个部分：

“version”：标签文件的版本号。
“flags”：一些标志位，用于标识标签文件的特殊属性。
“shapes”：一个数组，包含了图像中的所有标签信息。
    “label”：标签的名称。
    “points”：标签的标注起点和终点的坐标。
    “shape_type”：标签的形状类型，例如矩形、多边形等。


-----------------------------------------------------------------
{
  "version": "4.5.6",
```

```
  "flags": {},
  "shapes": [
    {
      "label": "person",
      "points": [[10, 20], [30, 40]],
      "shape_type": "rectangle"
    },
    {
      "label": "car",
      "points": [[50, 60], [70, 80]],
      "shape_type": "rectangle"
    }
  ]
}
```

这个示例中包含了两个标签，一个是"person"，另一个是"car"。它们分别是矩形形状，起点和终点的坐标分别为(10, 20)和(30, 40)，(50, 60)和(70, 80)

YOLO框架常用命令及含义

```
eg:
# yolov8n-pose模型，迁移学习微调
!yolo pose train data=Triangle_215.yaml model=yolov8n-pose.pt pretrained=True
project=Triangle_215 name=n_pretrain epochs=50 batch=16 device=0

# yolov8n-pose模型，随机初始权重，从头重新学习
!yolo pose train data=Triangle_215.yaml model=yolov8n-pose.pt
project=Triangle_215 name=n_scratch epochs=50 batch=16 device=0
```

model:传入的model.yaml文件或者model.pt文件，用于构建网络和初始化；
data:训练数据集的配置yaml文件；
pretrained:是否加载预训练的权重；
project：该项目的文件夹名称；
name：用于保存训练文件夹名；
epochs：训练轮次；
batch:训练批次；
device:设备

模型轻量化的策略（剪枝、知识蒸馏、量化）

模型轻量化的原因：减少模型的大小，提高推理速度，让模型能够成功的部署在各个平台上运作

模型剪枝：

细粒度剪枝(fine-grained)：即对连接或者神经元进行剪枝，它是粒度最小的剪枝。

向量剪枝(vector-level)：它相对于细粒度剪枝粒度更大，属于对卷积核内部(intra-kernel)的剪枝。

核剪枝(kernel-level)：即去除某个卷积核，它将丢弃对输入通道中对应计算通道的响应。

滤波器剪枝(Filter-level)：对整个卷积核组进行剪枝，会造成推理过程中输出特征通道数的改变。例如Dropout就是典型的模型剪枝；

模型量化：

量化，即将网络的权值，激活值等从高精度转化成低精度的操作过程，例如将32位浮点数转化成8位整型数int8，同时我们期望转换后的模型准确率与转化前相近。

优势：

1.更小的模型尺寸；

2.更低的功耗；

3.更快的计算速度；

知识蒸馏：

一般地，大模型往往是单个复杂网络或者是若干网络的集合，拥有良好的性能和泛化能力，而小模型因为网络规模较小，表达能力有限。利用大模型学习到的知识去指导小模型训练，使得小模型具有与大模型相当的性能，但是参数数量大幅降低，从而可以实现模型压缩与加速，就是知识蒸馏与迁移学习在模型优化中的应用。

二、判断题（每小题 1 分，本题共 15 分）

目标检测算法的类型（单阶段和双阶段）

单阶段目标检测算法：

速度快，有更快的推理速度，适用于实时应用或对速度要求较高的场景。（快 实时性好）；

虽然单阶段算法在速度方面具有优势，但它们通常在准确性上略逊于双阶段算法。然而，一些高级单阶段算法已经在准确性方面取得了显著进展。（准确性略差）

代表算法：YOLO,SSD等

双阶段目标检测算法：

双阶段算法通常在目标检测任务的准确性方面表现出色。它们可以提供高质量的目标检测结果，特别适用于复杂场景和需要高精度的应用。（适用于复杂场景和高精度应用）；

由于需要两个独立的阶段，双阶段算法通常需要更多的计算资源和时间。因此，它们的推理速度相对较慢。（慢）；

代表算法：Faster R-CNN、Mask R-CNN、Cascade R-CNN等

labelme文件格式的理解（选择部分详解---》）

网络宽度与网络深度的训练难度

网络宽度通常在训练集上产生较好的性能，但需要更多的数据来防止过拟合。 如果你的数据有限，可以考虑减小网络宽度，以减少过拟合的风险。 通常，增加宽度是一种增加模型复杂度的方式，适用于任务较复杂的问题，如大规模图像分类或自然语言处理。

网络深度： 网络深度指的是神经网络中的层数。 增加深度可以增加模型的抽象能力，使其能够学习更高级的特征。 然而，深度也会增加训练时间和梯度消失/爆炸的问题。

对于某些任务，深度网络可能不是最佳选择，因为训练深层网络可能会很困难，需要大量的数据和计算资源。

YO LO框架相关（模型选择）

目标检测、目标追踪、实例分割、关键点检测

detect、segment、classification，pose

三、填空题（本大题共 15 空，共 15 分）

爬虫图片数据处理

- 删除非三通道图片
- 删除GIF图片；

图片预处理步骤：缩放、裁剪、转Tensor和归一化

正向传播一般会得到两个数据结果：置信度和所对应的索引

评价图像分类任务的四个指标

```
1 Accuracy（准确率/精度）
2 Precision（精确率/查准率）
3 Recall（召回率/查全率）
4 F1-Score
```

四、简答题（本大题共 30 分，每小题 6 分，共 5 小题）

三角板关键点检测yolo数据格式及其含义

```
0 0.50449 0.21040 0.60022 0.42080 0.79634 0.07651 2 0.21588 0.01185 2 0.34734 0.40733 2
0 0.51365 0.74353 0.61063 0.43427 0.81286 0.62392 2 0.21875 0.53879 2 0.34231 0.95151 2
```

格式为txt
含义：
第一个数据为0，表示该目标的类别，（钝角，直角，锐角其中的一个）
接下来的四个数据表示了矩形框的坐标
后边的坐标表示了关键点的坐标，以及是否可见，其中 0.00000 表示没有显露出不可见，1.00000 表示被遮挡不可见，2.00000 表示可见

图像分类任务：只训练最后一层：全参数微调、全框架微调，三种训练方式的区别

一：只微调训练模型最后一层（全连接分类层）

```
In [22]: model = models.resnet18(pretrained=True) # 载入预训练模型

         # 修改全连接层，使得全连接层的输出与当前数据集类别数对应
         # 新建的层默认 requires_grad=True
         model.fc = nn.Linear(model.fc.in_features, n_class)

In [23]: model.fc
Out[23]: Linear(in_features=512, out_features=30, bias=True)

In [24]: # 只微调训练最后一层全连接层的参数，其它层冻结
         optimizer = optim.Adam(model.fc.parameters())
```

二：微调训练所有层

```
In [51]: model = models.resnet18(pretrained=True) # 载入预训练模型

         model.fc = nn.Linear(model.fc.in_features, n_class)

         optimizer = optim.Adam(model.parameters())
```

三：随机初始化模型全部权重，从头训练所有层

```
In [52]: model = models.resnet18(pretrained=False) # 只载入模型结构，不载入预训练权重参数

         model.fc = nn.Linear(model.fc.in_features, n_class)

         optimizer = optim.Adam(model.parameters())
```

区别：
数据量少，但数据相似度非常高，在这种情况下之训练最后一层；
如果相似度低，就使用全框架微调；
如果数据量大，采用全框架微调，只载入模型结构，不载入预训练权重参数

关键点检测任务数据集划分的实现流程

yolo命令

目标检测、图像分割和关键点检测三大计算机视觉研究任务的研究内容