实验项目2：TinySeg 图像分割模型对比与优化研究

该实验围绕图像分割模型的核心架构展开对比研究，基于小型数据集TinySeg（128×128像素），可以使用PyTorch、MindSpore等成熟深度学习框架，要求学生实现并优化三种主流模型——PSPNet、DeepLabv3（含ASPP模块）和CCNet（交叉注意力机制），通过数据增强、长周期训练（≥50 epoch）及多维度评估（mIoU、Dice系数、边缘F1分数、混淆矩阵），探究不同解码器结构对分割精度、计算效率和长程依赖建模能力的影响，最终通过可视化错误样本和参数对比，深化对语义分割模型设计原理的理解，同时培养工业级深度学习任务的工程实践能力，为真实场景中的实时分割任务提供选型参考。

1. 数据集与预处理

1.1 数据集说明

- **TinySeg 数据集**
  - 图像数量：6,624张（训练集6,000张，验证集624张）
  - 图像尺寸：128×128 RGB图像
  - 标注格式：单通道像素级掩码（类别索引值）
  - 下载链接：百度网盘 提取码：`mw3l`
  - 下载数据集之后，请可视化该数据集，熟悉数据格式；注意，实验只能使用这个数据集，不能采用别的数据集

1.2 数据增强策略

- **基础增强**（必须实现）：

```
from albumentations import (
    HorizontalFlip, RandomRotate, RandomBrightnessContrast
)
base_aug = Compose([
    HorizontalFlip(p=0.5),
```

```
    RandomRotate(limit=15, p=0.5),
    RandomBrightnessContrast(brightness_limit=0.2, contrast_limit=0.2, p=0.3)
])
```

- **高级增强**（至少选择2种）：

```
advanced_aug = Compose([
    ElasticTransform(alpha=1, sigma=50, alpha_affine=50, p=0.3),
    GridDistortion(num_steps=5, distort_limit=0.3, p=0.3),
    RandomCrop(width=96, height=96, p=0.5),  # 上采样回128×128
])
```

- **要求**：对比以下三种配置的mIoU差异：

配置	增强组合
无增强	仅归一化
基础增强	base_aug
完整增强	base_aug + advanced_aug

---

2. 模型实现要求

2.1 基准模型：PSPNet

- **架构**：ResNet18（骨干） + PSPNet（解码器）
  - PSP模块需包含4级金字塔池化（1×1, 2×2, 3×3, 6×6）
  - 最终特征图通过双线性上采样恢复至128×128

2.2 DeepLabv3

- **ASPP模块设计要求**：

```
# 4个并行分支
branches = [
    Conv2d(in_channels, out_channels, 1),                          # 1×1卷积
    Conv2d(in_channels, out_channels, 3, padding=6, dilation=6),   # 空洞率6
    Conv2d(in_channels, out_channels, 3, padding=12, dilation=12), # 空洞率12
    Sequential(GlobalAvgPool2d(), Conv2d(...), Upsample(...))      # 全局池化
]
```

- **输出融合**：使用1×1卷积将4分支特征相加后输出

2.3 CCNet（Criss-Cross Attention）

- **核心模块实现步骤**：
  a. **特征映射**：输入特征图 $X \in \mathbb{R}^{C×H×W}$
  b. **生成Query/Key**：

```
query = Conv2d(C, C//8, 1)(X)  # [C//8, H, W]
key = Conv2d(C, C//8, 1)(X)    # [C//8, H, W]
```

  c. **注意力计算**：
    - 对每个位置 (i,j)，计算水平和垂直方向的关联权重
    - 两次交叉注意力（Criss-Cross）迭代
  d. **输出融合**：$Y = \text{Conv}(X + \text{Attention}(X))$
- **约束条件**：
  - 注意力头数 ⩾ 2
  - 支持GPU并行计算（禁止使用for循环实现）

---

3. 训练与评估配置

3.1 训练参数

- **训练周期**：至少50个epoch（早停条件：验证集mIoU连续10个epoch无提升）
- **优化器**：AdamW（lr=0.001, weight_decay=1e-4）
- **学习率调度**：CosineAnnealingLR（T_max=20, eta_min=1e-5）
- **损失函数**：CrossEntropyLoss + DiceLoss（权重比1:1）

3.2 评估指标

1. **基础指标**：
   - **Pixel Accuracy**: $\frac{\text{正确像素数}}{\text{总像素数}}$
   - **mIoU**: $\frac{1}{C} \sum_{c=1}^C \frac{TP_c}{TP_c + FP_c + FN_c}$
   - **Dice系数**: $\frac{2 \times TP}{2 \times TP + FP + FN}$

2. **进阶指标**：
   - **类别平衡mIoU**：对每个类别的IoU进行平均（避免大类主导）
   - **边缘F1分数**：通过Sobel算子提取边缘区域后计算F1

```
# 边缘掩码生成
sobel_x = cv2.Sobel(mask, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(mask, cv2.CV_64F, 0, 1, ksize=3)
edge_mask = (np.abs(sobel_x) > 0) | (np.abs(sobel_y) > 0)
```

---

4. 可视化与分析要求

4.1 训练过程可视化

- **训练曲线**：绘制损失和mIoU随epoch的变化曲线（训练集 vs 验证集）
- **混淆矩阵**：

```
import seaborn as sns
cm = confusion_matrix(y_true.flatten(), y_pred.flatten(), normalize='true')
```

```
sns.heatmap(cm, annot=True, fmt=".2f", xticklabels=CLASSES,
yticklabels=CLASSES)
```

4.2 预测结果可视化

- **样本对比**：
  - 每个类别选择3张代表性验证集样本
  - 展示：原图、真实掩码、PSPNet预测、DeepLabv3预测、CCNet预测
- **错误分析**：
  - 用红色标注假阳性（FP），蓝色标注假阴性（FN）区域
  - 重点分析小目标（如交通灯）和边界模糊类（如植被边缘）

---

5. 提交材料

5.1 代码要求

- **模块化结构**：

```
project/
├── models/          # 模型实现（pspnet.py, deeplabv3.py, ccnet.py）
├── dataloader.py    # 数据加载与增强
├── train.py         # 训练脚本（支持早停、学习率调度）
└── evaluate.py      # 指标计算与混淆矩阵生成
```

- **可复现性**：提供完整的 `requirements.txt` 和随机种子设置（如 `torch.manual_seed(42)`）

5.2 实验报告

- **必选内容**：

a. 模型结构图（PSPNet、DeepLabv3、CCNet的核心模块）

b. 数据增强消融实验（表格对比mIoU和边缘F1）

c. 模型对比表格（参数量、mIoU、Dice、FPS）

d. 混淆矩阵热力图（标注高频错误如“天空→建筑”）

- **可选内容**：
  - 不同骨干网络（如ResNet34）的对比
  - 注意力权重可视化（CCNet的交叉注意力热力图）
- **注意**：实验报告中请写清楚姓名和学号

---

6. 参考文献

1. **PSPNet**: Paper | Code
2. **DeepLabv3**: Paper | TorchVision实现
3. **CCNet**: Paper | 官方代码
4. **Albumentations**: 文档

实验设计说明

1. **系统性对比**：通过统一的数据增强、训练策略和评估指标，确保模型对比的公平性。
2. **工程实践导向**：引入工业级工具链（如Albumentations、PyTorch Lightning），提升代码规范性。
3. **深度分析要求**：通过混淆矩阵和边缘F1分数，引导学生从像素级错误中挖掘模型弱点。
4. **扩展性**：模块化代码结构便于后续添加新模型（如UNet、Transformer）。