模型 2：CAN 运行说明

官方代码在 GitHub 上

为了在训练的 I/O 耗时和各位同学的电脑内存不会爆炸之间取得一个平衡，我们重新设计了数据预处理结果—— `.npy` 格式的数据集文件。但这样子就需要修改一些 CAN 官方的代码才能够正确跑起来

各位的任务就是将预处理得到的 `.npy` 文件作为输入，给模型跑起来。参考的思路为在 `dataset.py` 中增加 `MyDataset` 类，自定义一个我们的数据集。

> PS: 建议实现该类的时候按需加载各个 `.npy` 文件，以免内存爆炸

如何训练?

确保装好了一些额外的包：

```
pip install -r requirements.txt
```

在 `vocab.txt` 开头加上（假如没有的话）：

```
eos
sos
```

确保 `config.yaml` 中的参数设置符合实际情况。除了路径之外，还要确保这里符合 `vocab.txt` 里的行数：

```
counting_decoder:
  in_channel: 684
  # vocab.txt 的行数
  out_channel: 415
```

训练，启动!

```
python train.py
```

训练完了要保存什么?

最重要的：`checkpoints` 文件夹下的子文件夹里有模型的训练结果 `.pth` 权重文件

最终提交要用到的

- 模型在测试集上输出的标签
- 模型的复杂度

  CAN 训练阶段用的是 `CAN` 类，而测试阶段用的是 `Inference` 类。**我们这里测试的是 `CAN` 类在不加载模型权重的时候测得的复杂度**

  单位：Mac。**测试复杂度时输入图像维度是 (1, 1, 64, 64), 标签输入维度是 (1, 16)**，即调用参数为：

```
def test_params_flop(model,x_shape):
    """
    You need to give default value to inputs in model.forward(), the following code can only
    """
    from ptflops import get_model_complexity_info
    with torch.cuda.device(0):
        macs, params = get_model_complexity_info(
            model.cuda(),
            x_shape,
            as_strings=True,
            print_per_layer_stat=False
        )
        print('{:<30}  {:<8}'.format('Computational complexity: ', macs))
        print('{:<30}  {:<8}'.format('Number of parameters: ', params))

test_params_flop(model, (1, 64, 64))
```

  本模型的原始复杂度参考为 5.94e8 Mac (即 593.51 MMac)

请尽力在不损失太多性能的情况下，尝试优化模型的结构以减小模型复杂度，以获得较好的最终成绩。

如何提高模型的最终表现?

评价的指标不仅有模型的精度，还有新增了模型的复杂度。所以**照抄原本的模型会导致分数很低**，各位同学还需要想办法优化模型的复杂度。

- 模型精度：

  多摸索，尝试更改 `MyDataset.yaml` 中的一些参数，例如：

  - `epochs`：训练的轮数
  - `lr`：学习率

- 模型复杂度：

  需要尝试优化模型当中的部分结构，观察模型复杂度变化和结果精度的变化，取得一个平衡