同学可以根据自己电脑情况，选择 kaggle 平台在线运行代码，或本地配置环境，本地运行代码。建议采用后者，学习深度学习中环境配置的流程。
Kaggle 平台代码：https://www.kaggle.com/code/xiaofang12138/ml-course3
本地环境配置与运行代码示例如下：

第一步：环境配置

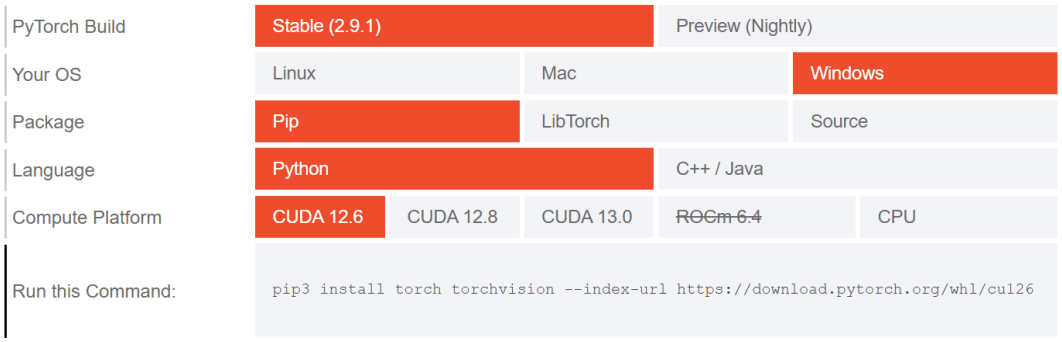
若你的电脑没有独立显卡，请直接跳转到第五步
若你的 pytorch 已安装成功，请直接跳转到

1. **如果你的电脑未安装 anconda，建议安装 anconda：**
   下载链接：https://www.anaconda.com/download
   （选择 Windows 版本，一般选 64 位）
   安装时勾选 **“Add Anaconda3 to my PATH environment variable”**（方便后续调用），其余默认下一步。
   注：成功安装 anaconda 后，打开 anaconda prompt 会显示”(base) 路径>”，可以自查以下 anaconda 是否安装成功
2. **创建虚拟环境：**
   在 win 搜索框搜索 Anaconda Prompt，打开并输入：
   conda create -n pytorch_env python=3.9
   conda activate pytorch_env
3. **查看系统 CUDA 版本：**
   在 win 搜索框搜索 powershell，打开并输入 nvidia-smi

```
(base) PS C:\Users\China> nvidia-smi
Fri Nov 28 21:53:02 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 560.94                 Driver Version: 560.94         CUDA Version: 12.6     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                  Driver-Model | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |                  MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 3070      WDDM  |   00000000:01:00.0  On |                  N/A |
|  0%   50C    P0             52W /  270W |    3538MiB /   8192MiB |      1%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

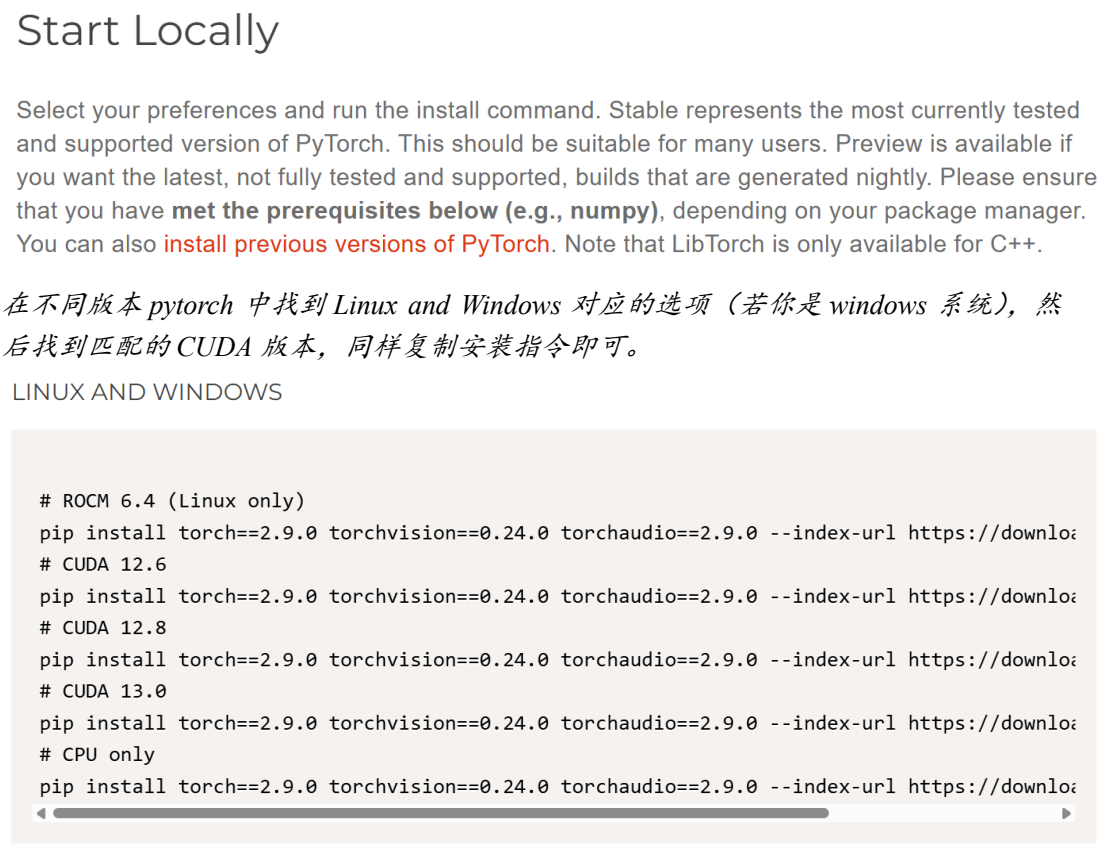
4. **获取 CUDA 版本对应的 pytorch 安装指令：**
   Pytorch 网站链接：https://pytorch.org/get-started/locally/
   注：安装需满足：pytorch 对应的 CUDA 版本≤系统的 CUDA 版本
   例如我的系统 CUDA 版本为 12.6，那么我可以安装 CUDA 12.6 及以下的 pytorch 版本。

NOTE: Latest Stable PyTorch requires Python 3.10 or later.



选择完对应的 compute platform 之后，复制安装指令，即"Run this command"框中的指令。

若以上 pytorch 对应的 CUDA 版本均不匹配你系统的 CUDA 版本，点击"install previous versions of pytorch"查看更多 pytorch 版本。



在不同版本 pytorch 中找到 Linux and Windows 对应的选项（若你是 windows 系统），然后找到匹配的 CUDA 版本，同样复制安装指令即可。

5. **安装 CPU 版本的 pytorch（电脑有独立显卡的跳过这部分）：**

在 compute platform 勾选 CPU，复制安装指令

PyTorch Build	Stable (2.9.1)			Preview (Nightly)	
Your OS	Linux		Mac	Windows	
Package	Pip		LibTorch	Source	
Language	Python			C++ / Java	
Compute Platform	CUDA 12.6	CUDA 12.8	CUDA 13.0	~~ROCm 6.4~~	CPU
Run this Command:	pip3 install torch torchvision				

6. **安装 Pytorch**

在 Anaconda Prompt 中激活虚拟环境，输入：

conda activate pytorch_env

输入此前复制的 pytorch 安装指令。例如，若是 CUDA 版本，输入：

pip3 install torch torchvision --index-url https://download.pytorch.org/whl/cu126

等待安装完成。安装完成后，测试是否安装成功：

在终端输入 python，输入如下指令：

```
(base) PS C:\Users\China> conda activate pytorch_gpu
(pytorch_gpu) PS C:\Users\China> python
Python 3.11.8 | packaged by Anaconda, Inc. | (main, Feb 26 2024, 21:34:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.cuda.is_available())
True
```

若你安装的是 CUDA 版本的 pytorch，安装成功后，指令会输出 True。

7. **安装其他依赖：**

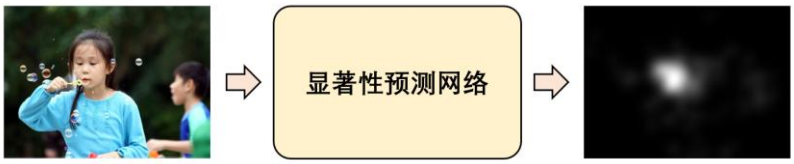
在项目目录下打开终端，输入：

pip install -r requirements.txt

等待其他工具库安装完成。

第二步：网络训练

任务整体流程即设计并训练一个显著性预测网络，给定输入的彩色图像，输出显著性预测的结果。



输入图像　　显著性预测结果

首先下载项目使用的数据集 3-Saliency-TestSet 和 3-Saliency-TrainSet

以下所有途径下载后文件相同，任选其一即可

- 百度云盘下载：

https://pan.baidu.com/s/1mOCFxATcCkHGbK8Vdtv5yQ

- DropBox 下载：

https://www.dropbox.com/sh/i79cbllw6763zxg/AAA3-jPaRlYHMvsMyRbtRRmaa?dl=0

- 北航网盘：

https://bhpan.buaa.edu.cn/link/AA84F755C78F1F4062BB81EBD5B41D5F7A

将下载后的训练与测试数据放在项目文件夹下，文件结构如下：

项目文件夹
|——*3-Saliency-TestSet*
|　　|——*FIXATIONMAPS*
|　　|　　|——...
|　　|——*Stimuli*
|　　|　　|——...
|——*3-Saliency-TrainSet*
|　　|——*FIXATIONMAPS*
|　　|　　|——...
|　　|——*Stimuli*
|　　|　　|——...
|——*dataset.py*
|——*main.py*
|——...

运行项目中的 main.py 函数即可开始训练，等待训练完成，训练的模型权重会保存在文件夹下的 resnet18_saliency_best.pth 文件。

下面提供如何调整参数并优化模型的方式：

1. 优化网络结构：
   - 本次课程采用 resnet18 作为网络的 backbone，网络结构代码在 model.py 文件中，可以尝试在网络结构中修改、添加一些网络层（例如卷积层、线性层），观察网络性能的变化。
   - 该版本网络预测的显著性图存在块状伪影，可以考虑优化网络结构（例如删去池化层等）解决这个问题。
2. 优化训练参数：

```
IMG_SIZE = (256, 256)  # 图像尺寸
BATCH_SIZE = 16  # 批次大小
EPOCHS = 10  # 训练轮数
LR = 1e-3  # 学习率
criterion = nn.MSELoss() # 损失函数

model = ResNet18Saliency(pretrained=True).to(DEVICE) # 初始化模型
optimizer = optim.Adam(model.parameters(), lr=LR) # 优化器
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.5)  # 学习率衰减
```

可以修改 main.py 中上述参数，进一步优化网络性能：

- IMG_SIZE：训练时输入图像的大小。由于提供的训练与测试图像大多为 1080P，直接输入网络训练会因为图像过大而显存溢出，因此需要在训练时将图像缩放到 IMG_SIZE 的大小。可以根据自己的 CUDA 显存合理调整该参数，一般而言，IMG_SIZE 与测试图像大小越接近，性能可能越好。
- BATCH_SIZE：一次性输入给网络的数据批次，BATCH_SIZE 越大，占用的显存会越多。这个可以根据自己的 CUDA 显存合理调整，尽量保证 BATCH_SIZE 是 2 的幂次，且尽可能大。
- EPOCHS：训练轮数。如果在训练中观察到 loss 还在持续下降，说明网络并没有完全拟合，可以通过增加训练轮数让网络拟合程度更好。
- LR：学习率，控制每次训练 iteration 网络参数的变化程度。可以尝试修改此参数优

化网络的性能。

- Criterion：损失函数，用于衡量训练中网络输出与真值的偏差程度，可以尝试替换，观察性能变化。例如替换为 nn.BCELoss ()
- Optimizer：优化器，用于根据 loss 优化网络参数。可以尝试替换，观察性能变化。
- Scheduler：学习率衰减，用于控制训练过程中的学习率变化。一般网络训练开始时学习率会较大，保证网络快速拟合，后期逐渐减小，防止在最优解附近震荡。可以尝试替换，或者修改 step_size 和 gamma 来观察性能变化。

注：若电脑没有独立显卡，网络训练可能过慢，可以直接采用助教训练好的模型 resnet18_saliency_best.pth 进行测试。

第三步：网络测试

运行项目中 test,py，会自动加载训练模型权重，进行测试，并输出测试集上预测结果的平均 CC 系数，以及平均 KL 散度。预测的显著性图会保存在项目中的 saliency_results 目录

改进方向：

1. 目前测试的逻辑是：为了与训练过程保持一致，将测试图像缩放到 IMG_SIZE 尺寸，输入网络得到输出后再放大回原始尺寸。这一过程会引入误差，可以对这一逻辑进行改进。
2. 提供的测试文件只会输出全部测试图像的平均指标，可以改进代码，针对不同图像类别输出相应的预测指标。