参考代码：

<https://github.com/swuxyj/DeepHash-pytorch>

<https://github.com/Huenao/DPCHash_Baselines>

运行环境：

GPU运行，Conda虚拟环境，下载GPU版本的Pytorch

代码文件解释：

1. network.py: 我们定义了加载了 CNNF 的预训练网络，定义 CNNF 网络的

结构，并且定义了前向传播函数。同时，我们还定义了用于优化检索效果的网络

ResNet。

1. tools.py: 我们定义了处理参数、加载数据集、处理数据集、计算结果、评估

性能、验证与保存模型的功能，以供 DPSH.py 训练模型使用

1. DPSH.py(4 个): 4 个 Python 文件，对应 CIFAR 数据集和 NUSWIDE 数据

集分别在 2 个不同网络（CNN-F,ResNet）上的训练。只需直接运行即可开始训练

模型（如有需要，请自行调整路径）。

1. DPSH\_test\_generate.py: 用于进行测试与生成 CIFAR 的训练集、测试集、

数据库集。如果能够成功运行此代码，代表能够直接运行上述 4 个 DPSH.py 文

件进行训练，并且成功生成了用于 CIFAR 的数据集。

1. precision\_recall\_curve.py: 根据生成的 log（位于 log 文件夹中），我们可以

生成 Precision-Recall (PR) 曲线。

1. loss\_mAP\_curve.ipynb： 根据生成的LOSS.txt和mAP.txt（位于RecordTrain文件夹中），生成训练过程的损失曲线和验证的mAP曲线。
2. demo\_NUSWIDE.ipynb : 指定好路径，可以展示某个基于 NUSWIDE21 数据集

训练的模型的部分检索结果，作为演示示例。

1. demo\_CIFAR.ipynb: 指定好路径，可以展示某个基于 NUSWIDE21 数据集训

练的模型的部分检索结果，作为演示示例。（因为 CIFAR 与 NUSWIDE21 读取数

据集的方法不同，所以有两个 demo）

运行准备：

1. 在DPSH下创建三个文件夹, 分别为log、RecordTrain、Save, 分别在这三个文件夹中创建两个文件夹，分别命名为ResNet和CNNF。这将分别对应用来保存log日志文件，loss和mAP记录文件，模型及其二进制哈希码表示。
2. 在DPSH下创建一个文件夹，命名为CNNFmodel，根据链接（http://www.vlfeat.org/matconvnet/models/imagenet-vgg-f.mat）下载预训练模型，将该预训练模型放入其中。
3. 在DPSH下创建文件夹datasets，在该文件夹中继续创建两个子文件夹，分别命名为cifar和nus-wide。下载NUSWIDE数据集，解压放入nus-wide文件夹中。
4. 您需要在DPSH.py以及tools.py中修改符合您运行环境的路径。

运行：

python DPSH\_test\_generate.py

如果能够成功运行该代码，证明您可以成功运行，并且生成了一个用于CIFAR训练的train.txt以及test.txt以及database.txt。随后你可以输入

python DPSH\_CNNF\_CIFAR.py

python DPSH\_CNNF\_NUSWIDE21.py

python DPSH\_ResNet\_CIFAR.py

python DPSH\_ResNet\_NUSWIDE21.py

demo

您可以指定路径，并运行两个ipynb来生成一个demo。

