结题报告

> 本报告为清华大学 2019-2020 春季学期电子工程系媒体与认知课程大作业图像实验结题报告，分别介绍各任务开展与完成情况。

> **报告说明**：由于网络学堂提交文件大小的限制，我们的代码和模型、测试结果同时放在了天津服务器的tsinghuaee221用户下，源码在~/src，权重在~/weights，测试结果在~/results，同时我们也在清华网盘上保存了这些内容的备份，助教也可以直接从此处下载，链接如下：https://cloud.tsinghua.edu.cn/d/f753341b982e4758b588/
> **！！如果助教想要测试需要去下载权重！！**

任务一 传统机器学习 & 分类任务

文献调研与方法原理

因为对于传统的机器学习方法与分类任务比较熟悉，而且传统的机器学习方法也相对比较固定和成熟，所以我们在简单的调研和尝试后就主要锁定了HOG和SVM两个方法。

HOG（Histogram of Oriented Gradient，方向梯度直方图）特征[1]，通过计算区块内像素点数据的横纵梯度并进行按角度分类，得到区块的梯度直方图，用于刻画图像的边缘信息。HOG特征的优点是可以对几何和光学的形变保持很好的不变形，换句话说，对环境的变化的鲁棒性较强。在实际操作中，将图像分为小的细胞单元(cell)，每个细胞单元计算一个梯度方向(或边缘方向)直方图。

SVM（Support Vector Machine, 支持向量机）[2]的基本模型是定义在特征空间上的间隔最大的线性分类器——通过构建高维超平面对数据进行划分，从而实现回归。可以使用多个SVM结合实现数据多类的分类。它的优点在于解是全局最优，理论基础较为完善；而缺点则在于二次规划问题的求解计算量偏大，需通过多个SVM组合实现多分类问题。

方案设计

该任务使用传统方法，我们首先使用独立的SVM，把图片直接flatten之后当做特征输入SVM进行训练，以此作为baseline；

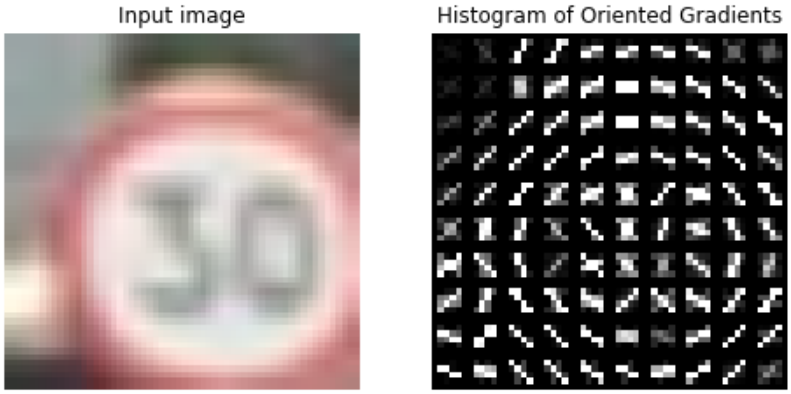
基于baseline的情况，我们认为需要对数据进行一定的预处理，因而采用了HOG来进行预处理。在将输入图片resize至同一大小之后，使用HOG方法将其处理为特征向量，然后直接交给SVM进行分类处理。

实验结果

Baseline方法为单纯使用SVM（默认参数）对图片进行分类，得到了89%的validation准确率（validation set为随机将training set中的30%数据取出组成），可见对于该数据集的分

类，传统机器学习方法的效果还是比较良好的，可以获得比较高的准确率，预计使用深度神经网络会获得更加优秀的效果

在Baseline方法直接将图片作为SVM的输入的基础上，使用scikit-image库进行图片的预处理并生成HOG，然后基于HOG特征向量使用基于scikit-learn的SVM进行训练，得到了95.9%的validation的准确率，效果较为良好。调整合适参数后HOG的效果展示如下：



代码运行方式

任务一的代码分成了训练和测试两部分，测试和训练的代码位于`/home/tsinghuaee221/src/exp1`我们也准备了训练好的权重，放在天津服务器的`/home/tsinghuaee221/weights/exp1.model`路径下（清华网盘中也位于对应文件夹中），进行训练和测试的命令分别为：

```
python exp1_train.py --path <path to image_exp/Classification/Data/Train >
```

```
python exp1_test.py --path <path to image_exp/Classification/Data/Teat> --weight <path to exp1.model>
```

问题与反思

总体来说任务一是比较简单的，使用的传统方法，方法比较固定也比较成熟，基于已有知识，我们较快地锁定了效果较好的方法，因此我们并没有在该任务上做过多调研，而是直接对传统方法进行了实现，效果还是十分不错的。

参考文献

[1] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE, 2005.
[2] Suykens, Johan AK, and Joos Vandewalle. "Least squares support vector machine classifiers." *Neural processing letters* 9.3 (1999): 293-300.

任务二 深度学习 & 分类任务

文献调研

深度学习在图像分类任务上总体是从多层感知机网络向卷积神经网络发展。Hinton于2006年发表的论文[1]中提出了基于感知机和反向传播的“深度学习”方法，通过多层感知机

的结合构成的网络对图片矩阵做多次矩阵运算得到分类结果，并通过监督的反向传播算法遏制梯度消失的问题。这种方法成为了深度学习的范式，运用在几乎所有领域的研究中。

Sermanet在2012年提出了卷积神经网络[2]，借鉴动物的视觉系统引入卷积层来学习并提取图片的特征。之后Hinton与学生设计的LeNet和AlexNet在图像分类任务上有了较大的提升，用卷积层和感知机结合的方法轻松超过了其他方法。随后人们通过加深网络的方式构建了VGGNet等深度网络，并都取得了非常好的效果。同时，各项研究表明通过增加网络深度可以一定程度上遏制过拟合，并能取得更好的效果。

Google的团队在2014年提出了GoogLeNet，或者称为Inception v1[3]。这种新网络引入了Inception块的结构，通过块内同时进行不同大小卷积核对特征的捕捉，用更少的层数与显著降低的参数量实现了效果良好的特征提取功能。这一研究表明增加深度带来的效果提升和迅速增大的参数量带来的计算压力相比并无优势，引发一轮对网络结构的研究。

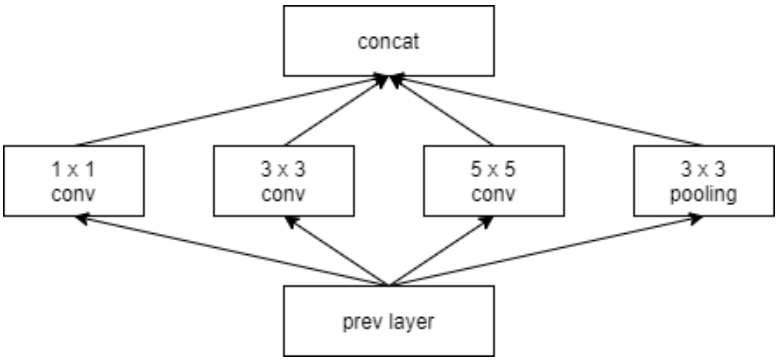
何恺明团队于2016年提出了ResNet [4]，通过引入残差连接使得网络可以学习出恒等映射 $f(x)=x$ 。引入恒等映射的目的是使网络可以自行学习适合的深度，并且加深网络不会出现反常的效果下降的现象。同时，残差连接使网络学习出恒等映射后，可以对网络进行修剪，减小参数量并提升前向速度。

受Inception和ResNet启发，Google的团队提出了适用于移动设备的MobileNet [5]。MobileNet基于Inception后续改进使用的空间可分离卷积和1x1卷积，发展出深度可分离卷积，将一次卷积分为多次卷积，再效果相差不多的情况下显著减少参数量。此外，MobilNet将ReLU改造为ReLU6，让超过6的值切顶，提升“低精度计算下的鲁棒性”。由于MobileNet是为了在移动设备上使用设计的网络，大幅减少参数量并使用了ReLU6等对精度要求不高的中间层，最终结果会有些微的下降。

对于任务二，综合考虑我们决定使用广泛使用的Inception v3作为主体，并采用迁移学习的方法取得较好的结果。具体原理将在下一节中详细阐述。

方法原理

Inception网络结构是由Google提出的知名网络结构。网络结构上与更早的关注于“深层”的网络不同，加入了多支路的设计，从而通过不同支路卷积核大小的不同来捕获不同范围的信息。Inception模块的结构基本如下图：



Inception v3 [6]在这个基础上，将3x3卷积和5x5卷积换为空间可分离卷积，即用nx1卷积和1xn卷积替换nxn卷积。这可以减少参数量、提升计算速度，但并不会影响准确率。Inception v4在此基础上添加了残差连接，但是在ImageNet数据集上并未有显著提升，且没有易获取的预训练参数，不适于迁移学习，因此我们选择使用Inception v3。在ImageNet数据集上，Inception v3至今依旧凭借较好的效果名列前茅。PyTorch中已有提供inception v3的网络模型和预训练权重，我们决定在此基础上做迁移学习。

迁移学习适用于特定任务数据集不是很大的应用场景。这一方法基于假设：在较大数据集上训练过的网络结构的一部分可以作为特征提取器，并可以泛用于各种场景。常用的迁移学习方法有两种：微调和特征提取器。特征提取器是指将预训练网络的权重全部固定，只对最后一层的分类预测进行训练，适用于数据集较小且与预训练所用数据较为相似的情形；微调则是将

相对较大的数据集直接用于进一步的训练，将预训练的权重进一步优化。迁移学习的优点在于节省时间和对数据集的要求的降低。

方案设计

采用PyTorch提供的在ImageNet数据集上预训练好的权重 `inception_v3_google-1a9a5a14.pth` ，将数据集随机划分为12000张图片的训练集和2400余张图片的验证集，使用SGD作为优化器，使用交叉熵作为损失函数，训练25个epoch并取验证集上的最佳结果对应模型进行保存。训练使用迁移学习的“微调”方式，即不固定任何权重优化整个网络。此外也尝试固定权重作为特征提取器的方法，作为前者的对照。

实验结果

我们认为Inception作为知名效果优良的CNN网络，直接作为此任务的baseline比较有代表性。使用Inception v3从头开始训练约10轮，得到的准确率约为97%，粗估超过人类进行分类，表现出了CNN的强大能力。

我们将数据集中的14464张图片随机分为了12000和2464两部分，分别用作训练和验证。基于之前的尝试，目前采用的是Inception的微调网络式迁移学习，得到了99.2286%的validation的准确率，总体效果出乎我们的意料。使用此方法训练的模型对测试集进行预测，提交验证得到的结果为98.2105%，最终模型权重为96MB。

而作为对照的使用固定网络作为特征提取器的方法，仅得到81.3642%的validation准确率。我们认为是交通标志与PyTorch提供的预训练权重使用的数据集有一定的差异，不太适用于作为特征提取器的方法。

代码运行方式

我们使用 `exp2.py` 进行模型的训练和测试集的预测

```
usage: exp2.py [-h] [--train_data TRAIN_DATA] [--test_data TEST_DATA]

               [--pretrain_weight PRETRAIN_WEIGHT] [--batch_size
BATCH_SIZE]
               [--lr LR] [--momentum MOMENTUM] [--step STEP] [--gamma
GAMMA]
               [--epoch EPOCH] [--save_model SAVE_MODEL]
               [--test_pred TEST_PRED] [--device DEVICE]
```

主要需要根据数据集和预训练权重的路径进行修改的参数为 `train_data` `test_data` `pretrain_weight` ，需要根据使用的GPU更改的参数为 `device` 。假定数据集位于 `./image_exp/Classification/Data` 下，预训练的Inception v3权重位于同一目录下，即 `./inception_v3_google-1a9a5a14.pth` ，使用命令

```
python exp2.py --device 0
```

将会使用GPU 0按照默认参数训练近1小时，将训练后的模型存储于 `./exp2_ft_model.pth` ，将预测结果存储于 `./exp2_pred.json`

注：我们直接使用 `torch.save()` 对模型进行保存，会有一些warning，但并不影响后续任务的使用。

问题与反思

目前此方法效果相对较为优秀，但是也存在一定的缺点。其最大缺点在于参数较多，保存权重所需空间较大，训练网络时间较长。此外，训练轮数可以适当减小，观测到验证集上结果在超过20轮后有微小的减小。不过由于训练函数会保存验证集上最佳结果对应模型，这对最终结果影响不大。

后续应参考一些类似mobilenet的小型网络，在尽量不影响效果的情况下减少参数量和缩短训练时间。

参考文献

[1] Hinton, G E, and R R Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks." Science (New York, N.Y.) 313.5786 (2006): 504-07. Web.
[2] Sermanet, Pierre, Soumith Chintala, and Yann LeCun. "Convolutional Neural Networks Applied to House Numbers Digit Classification." ArXiv.org (2012): ArXiv.org, Apr 18, 2012. Web.
[3] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." (2014). Web.
[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016 (2016): 770-78. Web.
[5] Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." (2018): The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, Pp. 4510-4520. Web.
[6] Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016 (2016): 2818-826. Web.

任务三 小样本分类

文献调研与方法原理

对于Few-shot learning（少样本学习）这个领域，我们是相对比较陌生的，所以我们先调研了它的相关定义。在人类快速学习能力的启发下，希望机器学习模型只需要少量样本的情况下，就可以较好地完成学习，这就是Few-shot learning。Few-shot Learning 是 Meta Learning（元学习）在监督学习领域的应用。Meta Learning，又可以认为是学习如何去学习，在 meta training 阶段将数据集分解为不同的 meta task，去学习类别变化的情况下模型的泛化能力，在 meta testing 阶段，面对全新的类别，不需要变动已有的模型，就可以完成分类。形式化来说，Few-shot 的训练集中包含了一些类别，每个类别中有数量较少的样本，不妨设为C个类别，每个类别含有K个样本，Few-shot learning就是要求模型从 C*K 个数据中学会如何区分这 C 个类别，这样的任务也被称为 C-way K-shot 问题。

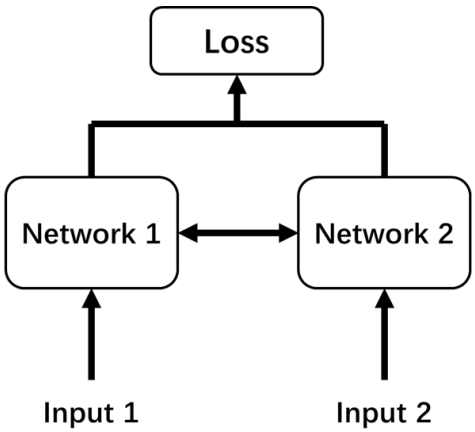
Few-shot Learning 算法研究多集中在图像领域，其模型大致可分为三类：Mode Based，Metric Based 和 Optimization Based。其中 Model Based 方法旨在通过模型结构的设计快速在少量样本上更新参数，直接建立输入 x 和预测值 P 的映射函数；Metric Based 方法通过度量 batch 集中的样本和 support 集中样本的距离，借助最近邻的思想完成分类；Optimization Based 方法认为普通的梯度下降方法难以在 few-shot 场景下拟合，因此通过调整优化方法来完成小样本分类的任务。

在本次大作业实验中，我们主要采取Metric Based模型来进行研究和尝试。

我们采取非参数化的方法作为baseline，采用的是KNN（K-近邻）方法，在元学习的框架下构造一种可以端到端训练的few-shot分类器，主要是对样本间距离分布进行建模，使得

同类样本靠近，异类样本远离。针对本次实验每类只有一个样本的特点，将KNN方法化为1-NN方法，采用欧式距离来进行衡量，将测试输入的图与已有各类的图比较欧氏距离，以最接近的作为该图的类别。根据文献[1]，1-NN方法在在omniglot数据集中的20类上进行单样本分类，可以得到大约28%的精度，已经是随机猜测（5%）的6倍精度了，所以我们认为在本实验中选用该方法作为baseline也是一个合理的选择。

孪生网络（Siamese Network）通过有监督的方式训练孪生网络来学习[1]，然后重用网络所提取的特征进行 Few-shot learning。它是一个双路的神经网络（事实上两路是共享权重的），训练时，通过组合的方式构造不同的成对样本，输入网络进行训练，通过两路CNN分别提取特征，然后在最上层通过样本对的距离判断他们是否属于同一个类，并产生对应的不相似度。在预测阶段，孪生网络处理测试样本和支撑集之间每一个样本对，最终预测结果为支撑集上不相似度最低的样本对应的类别。



孪生网络原理示意图

匹配网络（Match Network）[2]也是一种可行的方法。它为 support set 和 batch 集构建不同的编码器，最终分类器的输出是 support set 样本和 query 之间预测值的加权求和。而原型网络（Prototype Network）[3] 在此基础上，基于每个类别都存在一个原型表达，该类的原型是 support set 在 embedding 空间中的均值，将分类问题变成在 embedding 空间中的最近邻，更为简便，效果也相当。

此外，考虑到在某种程度上元学习和迁移学习的思想是一致的，所以在任务三中，我们认为之前任务所采用的方法可能也可以在任务三中呈现较好的结果，所以将其迁移到任务三中来进行尝试。

方案设计

1-NN作为非参数化方法，直接将图片都进行一定的归一化的处理，测试图片与训练集中的每类图片进行欧式距离的比较，选择最近的一类作为该测试图片的预测结果。

孪生网络主要是使用同一个CNN对不同的两幅图片进行特征提取，然后使用fc层计算其两者的不相似度，最终做分类的时候使用test set中的每一个图片与训练集中的样本进行比较，不相似度最小的一类为我们将其归入的类。在加载数据时，我们使用了取随机数的方式来保证训练数据中相同类别的组和不同类别的组各占50%。

考虑到任务3的数据与任务2的相似度非常高，在任务2中取得优异表现的迁移学习理应在任务3中也能发挥作用。使用任务2训练完后保存的模型 `exp2_ft_model.pth` 初始化Inception v3网络的特征提取部分，再使用PyTorch提供的Inception v3预训练权重初始化最后的线性层，在训练集的11张图片上训练25个epoch，再对验证集 `Test-dev` 做一次前向以表征结果，最后对整个测试集进行预测。整体流程上与任务2类似。

实验结果

对于我们来说，这个领域相对陌生，因此我们做了许多方法的尝试，目前主要有：

- KNN方法，在本实验中为1-NN，被我们当做baseline，测试的准确率为21.8%，虽然效果并不是特别理想，但是仍然优于随机猜测，具有作为baseline方法的价值。
- 孪生网络方法我们进行了一定的尝试，但是它的效果并不是特别理想。我们在实验时发现尽管对正确的来说其不相似度很小，但常有一个错误的类与之不相似度更小。推测在训练过程中出现了过拟合，并且数据预处理仍需要加强。由于其他方法的效果更好，我们后续没有再继续研究此方法。
- 目前我们取得效果最好的一种方法是迁移学习。我们使用了在任务二中取得98.6602%的Inception v3网络权重，在此基础上用任务三的数据对其进行fine tune，分别进行了微调和作为特征提取器两种训练，取得了83.1818%和70.4545%的结果。结果显著优于朴素的KNN方法，不过或许仍有提升的空间。

代码运行方式

- KNN

```
usage: exp3_knn.py [-h] [--train_data TRAIN_DATA] [--val_data VAL_DATA]
                   [--test_data TEST_DATA] [--save_model SAVE_MODEL]
                   [--test_pred TEST_PRED]
```

假定训练和测试数据集位于 `./image_exp/Classification/DataFewShot` 下，验证用数据集位于 `./Test-dev` （**注：验证集为测试集的子集，包含二十余张图片，仅用于模型运行结束后输出一次结果供使用者参考**），使用命令

```
python exp3_knn.py
```

将会按照默认参数训练，将训练后的模型存储于 `./exp3_knn_model.pth` ，将预测结果存储于 `./exp3_knn_pred.json`

对于任务3，执行此py文件应得到如下输出：

```
$ python exp3_knn.py
Val Acc: 0.13636363636363635
```

- 有关孪生网络，我们并没有对其进行最终的测试（由于他在validation中的表现并不好），但我们将运行的结果展示放在了 `src/exp3/Siamese.html` 中，可以看出其效果
- 基于任务2的Inception v3迁移学习

```
usage: exp3_transfer.py [-h] [--train_data TRAIN_DATA] [--val_data
VAL_DATA]
                        [--test_data TEST_DATA]
                        [--inception_weight INCEPTION_WEIGHT]
                        [--pretrain_exp2_model PRETRAIN_EXP2_MODEL]
                        [--batch_size BATCH_SIZE] [--lr LR]
                        [--momentum MOMENTUM] [--step STEP] [--gamma
GAMMA]
                        [--epoch EPOCH] [--save_model SAVE_MODEL]
                        [--test_pred TEST_PRED] [--device DEVICE]
```

主要需要根据数据集和预训练权重的路径进行修改的参数为 `train_data` `test_data` `val_data` `inception_weight` `pretrain_exp2_model` ，需要根据使用的GPU更改的参数为 `device` 。假定数据集位于 `./image_exp/Classification/DataFewShot` 下，验证用数据集位于 `./Test-dev` （**注：验证集为测试集的子集，包含二十余张图片，仅用于模型运行结束后输出**

一次结果供使用者参考），预训练的Inception v3权重位于同一目录下，即 `./inception_v3_google-1a9a5a14.pth` ，任务2的模型存储于 `./exp2_ft_model.pth` ，使用命令

```
python exp3_transfer.py --device 0
```

将会使用GPU 0按照默认参数训练约3分钟，将训练后的模型存储于 `./exp3_transfer_model.pth` ，将预测结果存储于 `./exp3_transfer_pred.json`
注：我们直接使用 `torch.save()` 对模型进行保存，会有一些warning，但并不影响后续使用。

问题与反思

- 从我们查阅的资料来看，孪生网络在Omniglot数据集和一些人脸对比识别的任务中都有着比较不错的表现，但这次在我们的任务中效果却不怎么理想，最高的validation准确率只有20%多，甚至不优于我们的baseline。我们认为可能的问题主要在于我们的训练数据比较少，容易出现过拟合，对数据进行增强可能会有所改进。
- 迁移学习在小样本学习有十分优异的表现，但是训练轮数过多，在约10轮后训练集上准确率便以达到100%。实际使用可以适当减小轮数，不过对整体效果影响应不大。

参考文献

[1] Siamese Neural Networks for One-shot Image Recognition
[2] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In Advances in Neural Information Processing Systems, pages 3630–3638, 2016.
[3] Snell, Jake, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning." Advances in Neural Information Processing Systems. 2017.

任务四 检测并分类

文献调研与方法原理

对于进行检测并分类的任务，目前主要有两种做法：一是一次同时进行检测与分类，二是检测和分类分成两步来进行。对于这些我们也进行了一定的调研，从包括CornerNet[1]，Random Forests[2]，yolo[3]，Faster RCNN[4]在内的多种方法中，我们着重考虑了分区域划分和分类两个步骤的Faster RCNN，以及一步完成检测和分类的著名方法yolo。

Faster RCNN是一种将检测和分类分成两部分执行的模型。其首先通过卷积将不同尺寸的图片调整为后续网络使用的“标准”尺寸，然后在卷积得到的特征图上的每个像素点周围都预测可能的方框区域，再将区域和对应的特征图一起传输给最后的分类器进行分类和回归两个任务。其中特征提取的部分可以使用任何一种图像分类任务的模型，只需约定好提取得到特征的尺寸以及可以用于区域预测的特征图。一般常见的为VGGNet和ResNet作为特征提取器的RCNN，此外也有使用MobileNet的。在此任务中，我们将首先尝试使用任务二训练好的Inception v3网络，其次我们也尝试了使用PyTorch给出的教程中基于MobileNet的网络。

yolo是一种比较成熟的目标检测模型，其全称是全称是you only look once，指只需要浏览一次就可以识别出图中的物体的类别和位置，其具体原理比较复杂，在此不再赘述。它具有快速、pipline简单、背景误检率低、通用性强等优点，但也有识别物体位置精准性差，召回率低的缺点。

方案设计

- 直接使用yolo进行训练并处理结果
- 参考PyTorch提供的教程中对Faster RCNN进行搭建
- 使用Faster RCNN并且将分类器换为我们在实验二中训练的inception网络

实验结果

- yolo：测试结果位于 `/home/tsinghuaee221/results/exp4_yolo.json`
- Faster RCNN：测试结果位于 `/home/tsinghuaee221/results/exp4_test_inception.json` 和 `exp4_test_mn.json`

代码运行方式

- yolo：由于yolo是使用了开源版本，如果助教想要进行测试需要clone官方仓库https://github.com/ultralytics/yolov3，然后将测试图片放到 `./data/samples` 文件夹或者在下面的命令中用 `--source` 参数指定测试图片，并且使用我们修改过的文件yolov3-spp.cfg、detect.py和sign.names以及训练好的权重，运行命令

```
python detect.py -cfg yolov3-spp.cfg --names sign.names --weights weights/best.pt --save-txt
```

  之后测试的结果会被保存在output文件夹，同时有图片和文本数据，将结果处理为json的脚本见exp4_test.ipynb
- Faster RCNN：需要将任务二训练好的模型 `./exp2_ft_model.pth` 放于 `exp4.py` 同一目录下，直接使用命令

```
python exp4.py
```

便可使用任务二的模型作为特征提取器训练Faster RCNN，并将模型保存于 `./model_RCNN.pth` 。之后使用命令

```
python test_exp4.py
```

便可以对测试集进行预测。以上两条命令皆预设数据集位于 `./image_exp/Detection` ，且训练需要依赖PyTorch提供的 `utils.py` `engine.py` `coco_utils.py` `coco_eval.py`

问题与反思

- Faster RCNN我们采用了任务2训练好的Inception v3和PyTorch提供的预训练MobileNet v2作为图像特征提取器。但由于构造基于Inception的网络时RoIAlign取特征是从浅层网络中提取，导致最终训练结果十分不理想，几乎没有识别对的。为了探究导致此结果的原因，我们使用PyTorch教程中的例子MobileNet，并尝试增多了RoIAlign使用的特征图来源，尤其是从更深层的特征提取器中获取特征，效果有了明显的进步。因而我们确定了是RoIAlign的定义出现错误，但由于时间原因，我们没法对修改后的基于任务2的Faster RCNN进行充分的训练。
- 对于yolo，虽然其效果比较好，但训练时它需要训练的轮次过多，比较消耗资源，并且在训练过程中常有报错中断，整体来说使用起来也并不是非常方便。

参考文献

[1] Law, Hei, and Jia Deng. "Cornernet: Detecting objects as paired keypoints." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[2] Ellahyani, Ayoub, Mohamed El Ansari, and Ilyas El Jaafari. "Traffic sign detection and recognition based on random forests." *Applied Soft Computing* 46 (2016): 805-815.

[3] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767* (2018).
[4] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *Advances in neural information processing systems*. 2015.

成员分工

- 贺鲲鹏：任务二、任务三、任务四、报告撰写
- 李煜泽：任务一、任务三、任务四、报告撰写
- 沈逸昕：任务三、任务四、报告撰写

小结

总的来说，我们在任务1、2、3上进展比较顺利，在尝试了一些网络后比较迅速地选取了主要使用的网络模型。不过由于任务4较为复杂且我们对相关网络不熟悉，加之我们尝试任务4时天津院的计算资源比较紧缺，导致走了不少弯路，最后也比较缺乏时间对网络进行改进。

此外，我们建议媒体与认知在未来的开展中引入类似kaggle的线上评测机制，方便同学对模型结果进行调试，也方便助教了解同学们作业的开展情况。

附录

最终上传的文件结构

```
│  结题报告.pdf
│
├─results
│      exp1_test.json
│      exp2_pred.json
│      exp3_knn_pred.json
│      exp3_transfer_pred.json
│      exp4_test_inception.json
│      exp4_test_mn.json
│      exp4_yolo.json
│
└─src
    ├─exp1
    │      exp1.ipynb
    │      exp1_test.py
    │      exp1_train.py
    │
    ├─exp2
    │      exp2.py
    │
    ├─exp3
```

```
    │       exp3_knn.py
    │       exp3_transfer.py
    │       Siamese.html
    │
    └─exp4
        ├─rcnn
        │       coco_eval.py
        │       coco_utils.py
        │       engine.py
        │       exp4.py
        │       test_exp4.py
        │       utils.py
        │
        └─yolo
                detect.py
                exp4_test.ipynb
                sign.names
                yolov3-spp.cfg
```

文件清单说明：
results中存放的是本次大作业各实验不同方法的结果，其中：

文件	说明
exp1_test.json	任务1 预测结果
exp2_pred.json	任务2 预测结果
exp3_knn_pred.json	任务3 knn 预测结果
exp3_transfer_pred.json	任务3 基于任务2的迁移学习
exp4_test_inception.json	任务4 Inception作特征提取
exp4_test_mn.json	任务4 MobileNet作特征提取
exp4_yolo.json	任务4 yolo

src中存放着各个实验的源码，其中：

文件	说明
exp1.ipynb	任务1 jupyter notebook
exp1_test.py	任务1 测试用py文件
exp1_train.py	任务1 训练用py文件
exp2.py	任务2 训练+测试 py文件
exp3_knn.py	任务3 knn 训练+测试 py文件
exp3_transfer.py	任务3 迁移学习 训练+测试 py文件
Siamese.html	展示孪生网络的效果
coco_eval.py	任务4 Faster RCNN依赖文件
coco_utils.py	任务4 Faster RCNN依赖文件

engine.py	任务4 Faster RCNN依赖文件
exp4.py	任务4 Faster RCNN 训练 py 文件
test_exp4.py	任务4 Faster RCNN 测试 py 文件
utils.py	任务4 Faster RCNN依赖文件
detect.py	任务4 yolo 测试 py文件(基于原文件做了修改)
exp4_test.ipynb	通过detect结果生成json
sign.names	所有类别的名字，用于detect
yolov3-spp.cfg	修改的网络结构文件