華中科技大學

电子信息与通信学院

《机器学习》报告

Quick, Draw!
Doodle Recognition Challenge

姓　　名　陈列可 赵欣然 邹琪珺

班　　级　种子 1601 班

时　　间　2019 年 5 月 19 日

目录

1 简介

1.1 选题介绍

Quick,Draw! 是一个游戏，规则是：让玩家画出给出的物体，然后让机器来猜测是哪种物体。本游戏有 340 种物体用来判别，数据集包括 50M 张玩家画的图。玩家画的图也会被添加在数据集中去。这是一种非常生动有趣的普及 AI 方式。

1.2 选题意义

本游戏不仅有趣，而且还有许多重要的实际意义。

如，手写识别，光学字符辨别（OCR），语音识别（ASR）以及自然语言处理（NLP）。.

1.3 数据集

数据集分为 raw 数据集与 simplified 数据集，simplified 数据集对许多信息进行了简化，比如对于直线，若这个直线由 8 个点构成，简化后数据集只有 2 个点了。

要注意的是：有些名词是由多个单词构成，如”The Great Wall”在我们的结论中，要将空格替换为下划线。

1.4 测评方法

判断出可能性最大的三种物体，让后使用 MAPK 对整体准确度进行评估。

图 1. 发送系统框图

2 数据预处理

2.1 读取 csv

通过同一后缀，符合所有条件的文件名称。

```
path = './doodle_dataset/train_simplified'# 数据集存储位置
filenames = glob.glob(os.path.join(path, '*.csv'))
filenames = sorted(filenames)# 获得所有类似后缀并排序
```

然后把检测到文件名截取尾部，只留下名字。

新建一个 dictionary 通过编号来索引名称。

再新建一个 dictionary，通过名称来索引编号。

读取表格时，读取指定行‘drawing’，读取 nrows 行，并跳过 skiprows 行，如下操作。

```
self.doodle = pd.read_csv(file, usecols=['drawing'], nrows=nrows,
skiprows=skiprows)
```

skiprows 是跳过的行数

注意 skiprows 不能把第一行跳过去，因为若以表头索引，跳过去就没有了

Skiprows 可以用 lamda 表达式来实现：下列式子就是跳过 2：3999 行

skiprows=lambda x: (x < 4000) & (x > 1)

2.2 构建数据集

使用 ConcatDataset 来构建一个数据集，这个数据集把之前的所有的都打乱了，

ConcatDataset 中有许多 DoodlesDataset 类，然后使用 DataLoader 来将这个数据集转化为加载器。(？？为什么要这样做？在哪一步将数据打乱的？)

2.3 取出数据以及图像显示

如何取出数据呢?

可以构建迭代器，也可以使用 for 来构成。

```
dataiter = iter(loader)
images, label = dataiter.next()
```

(这个是不是调用了__getitem__方法？iter(loader)得到的东西是不是随机的，这又是不是偶然？如果我想指定得到数据集中某一个类该如何操作？)

显示图像

```
def imshow(img):
    npimg = img.numpy() # 转化为 ndarry
    plt.imshow(np.transpose(npimg, (1, 2, 0)))

plt.figure(figsize=(16,24)) # 每次显示 16*24 的图（这样一行可以显示 8 个）
imshow(torchvision.utils.make_grid(images[:24])) # 显示前 24 张图
```

3 模型构建

可以选择自己构造模型，也可以使用现有的模型。

3.1 自行构建模型

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
```

```
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

在初始化中构造网络的各个层。

然后创建 forward 方法对输入网络的数据进行计算。

3.2 使用已有网络结构

```
model = torchvision.models.vgg11(pretrained = True)
```

如果仅使用结构，则不需要加后边的参数。

为了匹配输入的参数与输出的参数，我们应该修改第一层与最后一层的参数

首先使用

```
print(model.parameters)
```

来查看模型的结构。

然后得到了网络的结构：

(features): Sequential(
(0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(1): ReLU(inplace)
(2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(4): ReLU(inplace)
(5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(6): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(7): ReLU(inplace)
(8): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(9): ReLU(inplace)
(10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(11): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(12): ReLU(inplace)
(13): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(14): ReLU(inplace)
(15): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(16): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(17): ReLU(inplace)
(18): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(19): ReLU(inplace)
(20): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)

)
(avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
(classifier): Sequential(
(0): Linear(in_features=25088, out_features=4096, bias=True)
(1): ReLU(inplace)
(2): Dropout(p=0.5)

```
        (3): Linear(in_features=4096, out_features=4096, bias=True)
        (4): ReLU(inplace)
        (5): Dropout(p=0.5)
        (6): Linear(in_features=4096, out_features=1000, bias=True)
      )
    )>
```

可以看到，最后一层输出 out features 是 1000，我们把最后一层的 Linear 替换掉

```
num_classes = 340
model.classifier[6] = nn.Linear(in_features=4096, out_features=340, bias=True)
```

然后，我们的输入的图片的 type 为 tensor(batchsize, 1, 225,225)

而第一层网络为

(0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))

可见，卷积网络输入为 3 层，输出为 64 层，和我们输入的 1 层不符。

则，我们调整第一层，讲输入的三层捏成一层。这个 apply 是一个值得学习的新操作。

```
def squeeze_weights(m):
    m.weight.data = m.weight.data.sum(dim=1)[:,None]
    m.in_channels = 1
model.features[0].apply(squeeze_weights)
```

3.3 存储与读取模型

若训练好了模型，可以使用以下方式将模型保存：

这种方式没有保存多余的数据，仅仅巧妙的保存了网络的参数。

```
filename_pth='checkpoint_vgg11_SVD.pth'
torch.save(model.state_dict(), filename_pth)
```

如果已经训练好了同样结构的模型，在创建模型之后可以使用读取参数的方式加载，

```
loadModel = torch.load(os.path.join('./', 'checkpoint_vgg11_SVD.pth'))
model.load_state_dict(loadModel)
```

4 模型的训练

4.1 模型的测试

首先使用一个与数据集一样结构的（除了 batchsize 不同）跑一下模型看一看结果的维度是否正确。

```
model(torch.randn(12,1,224,224)).size()
```

得到

Tensor.size[12,340]

输入为 12 组数据，得到 340 个输出，这符合我们的预期。

4.2 将模型加载到显卡中

```
device = 'cuda'
model.to(device);
```

可以通过 Torch.cuda.is_available()来判断是否有可用 cuda。

注：如果显卡的算力小于等于 3.0，在使用显卡中的数据计算的时候会报错。我的 GTX760 算力正好 3.0，刚刚好不能用，然后找朋友借了一个 1070ti

如果要使用 GPU 来计算，要将是所有的变量都放在 GPU 中进行计算。X.to(device)操作可以转到 GPU。

如果想将 GPU 中的数据转出来，则需要使用 y.cpu()。但如果转出来时显存占用没有减少，则需要及时删除变量。

如果删除了变量，显存还是爆满，则是因为缓存没有清除，应当使用下边的命令清除缓存。

```
torch.cuda.empty_cache()
```

如果清除了显存，删除了变量，但还是满，则将数据集 batchsize 改小，如从 128 改成 64，可以解决显存问题。 （可以直接在任务管理器中查看显存）

4.3 网络的结构

卷积层--》Relu 层--》池化层--》卷积层--》Relu 层--》池化层……--》线性分类器

卷积层：对图像来说，卷积层是特别适用的，因为卷积关联了某一点的周围部分，做为一张图，这是一个整体。

激活函数层：使用非线性激活函数增加系统的复杂性。可以指定调节敏感性。如果没有激活函数层，整个系统就会变为一个完全线性的机器，失去了意义。

池化层：池化层降低了输入的特征值，我认为是对特征的一种提取

4.4 优化器

优化器是用来根据 loss 来求其对各层的梯度，进而更新网络中各个层的参数。因此，先要确定一个计算 loss 的方法。这个计算 loss 的方法是和数据的类型有关的。

实验中选择一个优化器，优化器用来根据梯度来更新参数的。具体一点，优化器是根据当前的状态与本次迭代得到的梯度，来决定下一步各参数的增减。

在我看来，优化器可以理解成一个低通滤波器，他平滑了当前的速度，来避免震荡，这好比控制理论中 PID 的 I，而学习率相当于 PID 中的 P。

如 SGD 这种“滤波器”，是对梯度进行随机增加的

SGD 每次更新时对每个样本进行梯度更新，对于很大的数据集来说，可能会有相似的样本，这样 BGD 在计算梯度时会出现冗余，而 SGD 一次只进行一次更新，就没有冗余，而且比较快，并且可以新增样本。

```
optimizer = torch.optim.SGD(model.parameters(), lr = 0.0002,
momentum=0.9);
```

ss function 有 nn.L1Loss（也叫 MAELoss）

$$\mathrm{loss}(x,y)=\frac{1}{n}\Sigma|x_i-y_i|$$

还有 nn.MSELoss 也就是求 L2 距离

$$\text{loss}(x,y) = \frac{1}{n}\Sigma(x_i - y_i)^2$$

```
criterion = nn.CrossEntropyLoss()
```

CrossEntropyLoss 交叉熵，是针对单目标分类问题产生的，这个损失函数是由 LogSoftmax 与 NLLLoss 结合计算的。Log_Softmax 最后一层的输出，然后:变正号，将所有正确项的值相加除以总测试数。然后后边的话为 NLLLoss。

4.5 损失函数

4.5.1 Loss 函数的本质

经鄙人研究发现,loss 函数的本质就是“归一化”并“计算离谱程度”

归一化

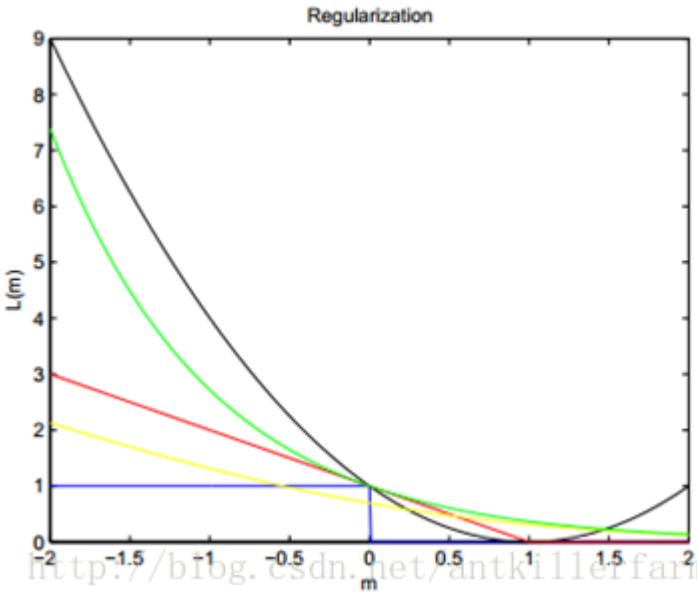
输入与输出是正相关,且输出最好不要变号,并限定在一定的范围内,由此可见 softmax 中

$$y_i = \frac{e^{x_i}}{\Sigma e^{x_i}}$$

很好的符合了这几个条件

计算”离谱程度”

首先我们先去计算，猜测结果完全符合我们要求，y_i的情况，然后我们想出一个式子，使得在最靠谱的条件下(预测与结果完全符合)，损失值最小，而且，当越来越离谱时，loss 会增加。



虽然不同 Loss 函数对“离谱程度”的评价不同，但此图向我们介绍了置信度与不同 Loss 的关系。横轴为置信度，纵轴为不同 Loss 的值。因为 Loss 都是和 batchsize 无关的，所以可以用来衡量网络的置信度。

其中蓝色的阶跃函数又被称为 Gold Standard，黄金标准，因为这是最准确无误的分类器 loss function 了。分对了 loss 为 0，分错了 loss 为 1。

其中红色线条就是 SVM 了，由于它在 m=1 处有个不可导的转折点，右边都是 0，所以分类正确的置信度超过一定的数之后，对分界面的确定就没有一点贡献了。

黄色线条是 Logistic Regression 的损失函数，与 SVM 不同的是，它非常平滑，但本质跟 SVM 差别不大。

绿色线条是 boost 算法使用的损失函数。

黑色线条是 ELM（Extreme learning machine）算法的损失函数。

4.5.2 交叉熵损失函数

交叉熵损失函数就很好的做到了这两点（CrossEntropyLoss）

交叉熵首先使用 Softmax，然后取对数，最后采用 NLLLoss 算法计算出最后的 Loss

第123行分别是第123张图片的结果，假设第123列分别是猫、狗和猪的分类得分。

可以看出模型认为第123张都更可能是猫。

然后对每一行使用Softmax，这样可以得到每张图片的概率分布。

```
[132]: sm=nn.Softmax(dim=1)

[133]: sm(input)

[133]: tensor([[0.6600, 0.0570, 0.2830],
               [0.5570, 0.1083, 0.3347],
               [0.4542, 0.2169, 0.3290]])
```

这里dim的意思是计算Softmax的维度，这里设置dim=1，可以看到每一行的加和为1。比如第一行0.6600+0.0570+0.2830=1。

```
[135]: sm=nn.Softmax(dim=0)

[136]: sm(input)

[136]: tensor([[0.2212, 0.0627, 0.1527],
               [0.3050, 0.1946, 0.2950],
               [0.4738, 0.7427, 0.5524]])
```

如果设置dim=0，就是一列的和为0。比如第一列0.2212+0.3050+0.4738=1。

我们这里一张图片是一行，所以dim应该设置为1。

然后对Softmax的结果取自然对数：

```
[139]: torch.log(sm(input))

[139]: tensor([[-0.4155, -2.8648, -1.2623],
               [-0.5852, -2.2232, -1.0945],
               [-0.7893, -1.5285, -1.1117]])
```

Softmax后的数值都在0~1之间，所以ln之后值域是负无穷到0。

NLLLoss的结果就是把上面的输出与Label对应的那个值拿出来，再去掉负号，再求均值。

Softmax 将全实数域的数全都正相关映射在 (0, 1) 中, 不过因为生成的类别较多，所有数字都为较小的正数。

Log 的作用是将利用 log 在零点附近斜率极大的特点，将上一层输出的正数差距放大，变为看起来差距较明显的负数。

NLL Loss 是选择将所有值取负，变为正数，然后选取正确项的预估值，求和。即得到损失函数。

根据上述，我们也制造了自己的 Loss 函数进行测试。我们第一步对输出的处理仍然选择 Softmax,然后，根据 Log 的意义在于放大零附近的梯度，我们采用了 1/x 来实现，然后对所有正确项的倒数进行相加，

4.5.3 Loss 不下降怎么办？

有下降的趋势，但非常缓慢：说明学习率太小了。

有较明显的抖动，正确率也迟迟不提升。可能是学习率太大了，或者数据集 batchsize 太大了，每次更新参数对整体的影响太大导致。

Loss 虽然下降，但准确率也下降了。这是因为，你编的 loss 函数一定有问题！

4.6 学习率调整器

学习率调节器是用来根据不同的迭代次数改变学习率用的。

因为最开始可以优化的范围较广，所以开始的变化率可以大一些，后来慢慢收敛，变化率应当越来越少。下边这个函数对学习变化率的控制如下式子所示：

```
scheduler = torch.optim.lr_scheduler.MultiStepLR(optimizer,
milestones=[5000,12000,18000], gamma=0.5)
```

$$lr = \begin{cases} lr & (iter < 5000) \\ gamma * lr & (5000 \le iter < 12000) \\ gamma^2 * lr & (12000 \le iter < 18000) \\ gamma^3 * lr & (iter \ge 18000) \end{cases}$$

4.7 进行训练

对于一次训练，有一下几个步骤

```
for x, y in loader:
    x, y = x.to(device), y.to(device)
    optimizer.zero_grad() # 梯度清零
    output = model(x) # 正向计算
    loss = criterion(output, y) # 计算损失函数
    loss.backward() # 反向求梯度
    optimizer.step() # 根据梯度自动更新各层网络参数
    scheduler.step() # 根据迭代次数更新学习频率
```

5 测试与分析

5.1 随机抽取数据集中进行测试。

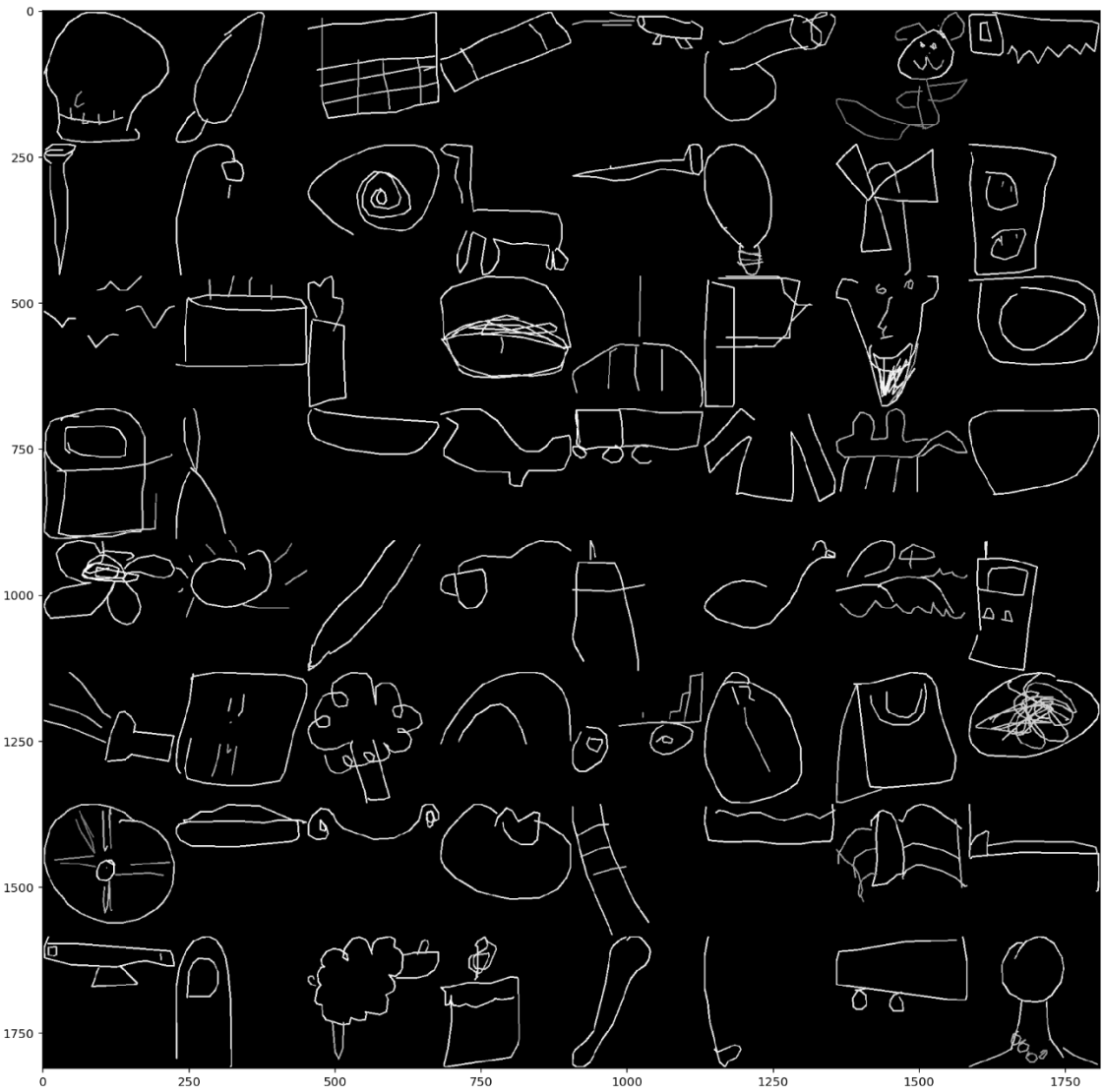
判定图片，显示预测结果与期望结果

以 resnet34 网络，损失函数为 CrossEntropyLoss，优化器为 Adam，为例。

以 64 张图片进行测试，前三个为猜测的标签名，若三个都没有猜中目标，则提示 Ben Dan, Guess Failed！最后统计猜中个数（三个之内猜中）以及准确率。

```
Guess: [              skull] [          light_bulb] [     hot_air_balloon] ||||||Expected [              skull]
Guess: [       baseball_bat] [             feather] [               leaf] ||||||Expected [            popsicle]
Ben dan ,Guess failed!
Guess: [           calendar] [             laptop] [        spreadsheet] ||||||Expected [           calendar]
Guess: [             eraser] [             crayon] [             marker] ||||||Expected [             eraser]
Guess: [             cannon] [          crocodile] [         sea_turtle] ||||||Expected [             cannon]
Guess: [             cannon] [              drill] [               bear] ||||||Expected [             cannon]
Guess: [             monkey] [                dog] [               yoga] ||||||Expected [             monkey]
Guess: [               saw] [                key] [            see_saw] ||||||Expected [               saw]
Guess: [              nail] [        streetlight] [             pencil] ||||||Expected [              nail]
Guess: [        streetlight] [         floor_lamp] [      traffic_light] ||||||Expected [        streetlight]
Guess: [               eye] [          hurricane] [              onion] ||||||Expected [          hurricane]
Guess: [           giraffe] [              horse] [              camel] ||||||Expected [           giraffe]
Guess: [           clarinet] [           trombone] [            trumpet] ||||||Expected [           clarinet]
Guess: [         light_bulb] [     hot_air_balloon] [           necklace] ||||||Expected [         light_bulb]
Guess: [           windmill] [               axe] [             pliers] ||||||Expected [           windmill]
Guess: [       power_outlet] [            stereo] [      traffic_light] ||||||Expected [       power_outlet]
Guess: [    animal_migration] [              bird] [              ocean] ||||||Expected [    animal_migration]
Guess: [               cake] [       birthday_cake] [            hot_tub] ||||||Expected [               cake]
Guess: [             candle] [           lipstick] [            matches] ||||||Expected [             candle]
Guess: [          hamburger] [            helmet] [           sandwich] ||||||Expected [             helmet]
Guess: [         chandelier] [              rake] [              broom] ||||||Expected [         chandelier]
Guess: [               axe] [           mailbox] [             hammer] ||||||Expected [               axe]
Guess: [             goatee] [             beard] [         strawberry] ||||||Expected [             goatee]
Guess: [              pool] [        hockey_puck] [             camera] ||||||Expected [             camera]
Guess: [          paint_can] [         calculator] [             cooler] ||||||Expected [          paint_can]
Guess: [             pliers] [    The_Eiffel_Tower] [           scissors] ||||||Expected [             pliers]
Guess: [            bathtub] [             canoe] [         watermelon] ||||||Expected [             canoe]
Guess: [              whale] [            dolphin] [              shark] ||||||Expected [              duck]
Ben dan ,Guess failed!
Guess: [              truck] [       pickup_truck] [              train] ||||||Expected [              truck]
Guess: [             jacket] [            sweater] [            t-shirt] ||||||Expected [            sweater]
Guess: [              camel] [             castle] [           mosquito] ||||||Expected [              camel]
Guess: [             pillow] [          underwear] [         watermelon] ||||||Expected [             basket]
Ben dan ,Guess failed!
Guess: [        ceiling_fan] [         camouflage] [             flower] ||||||Expected [        ceiling_fan]
Guess: [               sun] [             spider] [         flashlight] ||||||Expected [               sun]
```

```
Guess: [              pencil] [             crayon] [             marker] ||||||Expected [              pencil]
Guess: [              teapot] [              mouse] [          motorbike] ||||||Expected [              teapot]
Guess: [          skyscraper] [       fire_hydrant] [         lighthouse] ||||||Expected [          skyscraper]
Guess: [               swan] [        garden_hose] [               pond] ||||||Expected [               swan]
Guess: [              ocean] [              beach] [   animal_migration] ||||||Expected [              beach]
Guess: [          cell_phone] [          telephone] [         calculator] ||||||Expected [          telephone]
Guess: [         flashlight] [              sword] [          megaphone] ||||||Expected [         flashlight]
Guess: [        power_outlet] [                map] [               jail] ||||||Expected [        power_outlet]
Guess: [           broccoli] [               tree] [          palm_tree] ||||||Expected [               tree]
Guess: [            rainbow] [               tent] [             bridge] ||||||Expected [            rainbow]
Guess: [          motorbike] [            tractor] [            snorkel] ||||||Expected [            tractor]
Guess: [              clock] [            compass] [        alarm_clock] ||||||Expected [              clock]
Guess: [          paint_can] [           backpack] [                cup] ||||||Expected [         dishwasher]
Ben dan ,Guess failed!
Guess: [              brain] [        soccer_ball] [          blueberry] ||||||Expected [              brain]
Guess: [              wheel] [                fan] [            compass] ||||||Expected [              wheel]
Guess: [            hot_dog] [              mouth] [          hamburger] ||||||Expected [            hot_dog]
Guess: [         garden_hose] [              snake] [          moustache] ||||||Expected [          moustache]
Guess: [          blueberry] [              cloud] [               pond] ||||||Expected [          blueberry]
Guess: [             ladder] [           stitches] [             stairs] ||||||Expected [             ladder]
Guess: [               pool] [               shoe] [            bathtub] ||||||Expected [               pool]
Guess: [           backpack] [             spider] [               book] ||||||Expected [           trombone]
Ben dan ,Guess failed!
Guess: [                bed] [       diving_board] [              bench] ||||||Expected [                bed]
Guess: [            see_saw] [           airplane] [              anvil] ||||||Expected [            see_saw]
Guess: [               toe] [               nail] [             finger] ||||||Expected [             finger]
Guess: [              sheep] [           broccoli] [               bush] ||||||Expected [              sheep]
Guess: [             candle] [        house_plant] [               cake] ||||||Expected [             candle]
Guess: [              spoon] [       baseball_bat] [       hockey_stick] ||||||Expected [       baseball_bat]
Guess: [          golf_club] [       hockey_stick] [                leg] ||||||Expected [          golf_club]
Guess: [              truck] [          firetruck] [                bus] ||||||Expected [              truck]
Guess: [          necklace] [               yoga] [               nose] ||||||Expected [          necklace]
 Totally 64 test images, right 59 images, accuracy 92 %
```



整体测试准确度

随后在每个标签数据集（未参加训练）中抽取 100 中图片参与测试（共 34000 张）进行正确性测试。得到三张之内猜中概率为 89%。

```
In [20]: calcu()
100%|██████████| 160/160 [00:33<00:00,  4.75it/s]
Accuracy of the network on the 10200 test images: 89 %
```

5.2 修改参数对比规律

5.2.1准确率和各个参数的测试

网络结构	损失函数	优化器	学习率规划器	准确度
Resnet18	CrossEntropyLoss	Adam	MultiStepLR	89%
Resnet18	CrossEntropyLoss	Adam	StepLR	89%
Resnet18	Softmax+$\frac{1}{x}$	SGD	MultiStepLR	63%
Resnet34	CrossEntropyLoss	Adam	MultiStepLR	89%
Resnet34	CrossEntropyLoss	SGD	MultiStepLR	89%

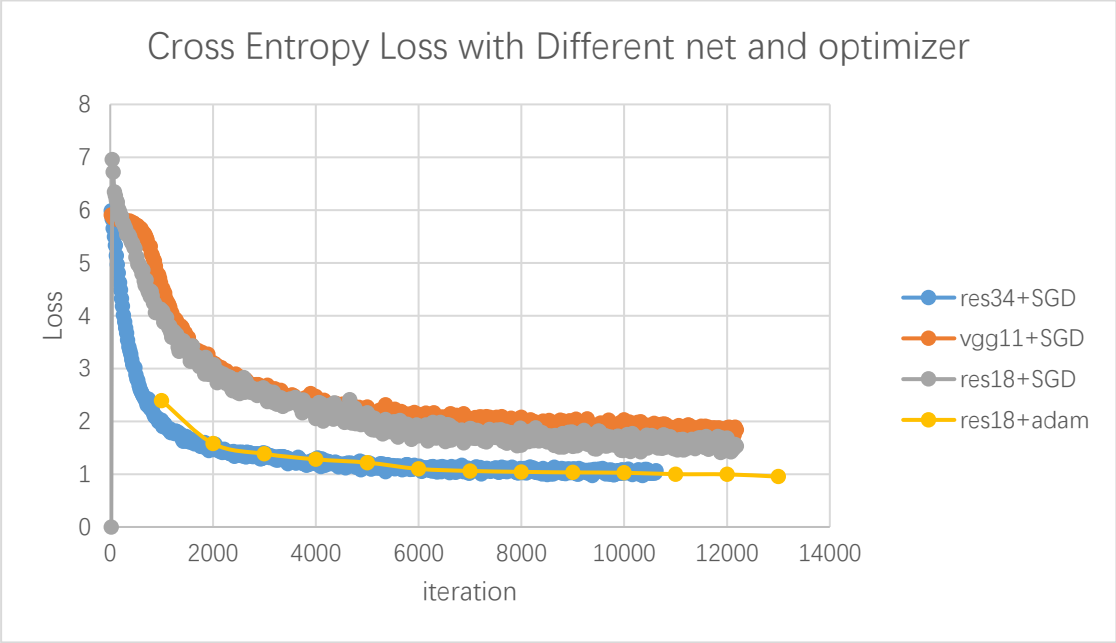
Densenet121	CrossEntropyLoss	Adam	MultiStepLR	80%
VGG11	CrossEntropyLoss	SGD	MultiStepLR	81%

从表格来看，如果数据集一定，准确度主要与网络与损失函数的选取有关。而与优化器，学习率规划器关系较小。

这也符合我们的预期，因为优化器的作用主要是协调参数的变化，可以理解成给予适当的阻力，平滑了突变的参数，有助于训练更迅速的走向最优解的捷径；而学习率规划器的作用是在调节学习率，在多次训练后能够主动降低学习率，加大细化程度。这两种都不能改变一种网络训练的本质。

但如果改变了网络与损失函数，就会对结果产生直接的影响。改变网络结构就不用说了，改变损失函数，就会改变每一层的梯度，使得每层的敏感度发生了变化，也可以说，对每种特征的敏感度不同了，显而易见，结果一定会发生变化。

5.2.2 网络与优化器对 Loss 的影响

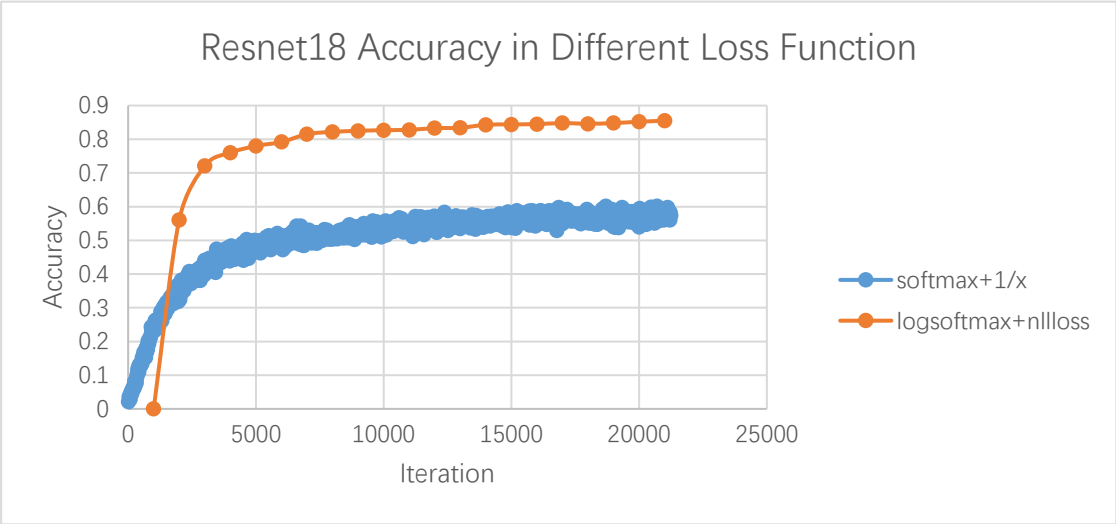


因为系统的输出张量的 shape 是[batchsize,340]，故 Loss 函数对于衡量不同的网络以及不同的优化器，学习率规划器都有同样有效的作用。由图可见 SGD 优化器可以让系统在初始训练时较为迅速的降低系统的 Loss，而 adam 就会在开始较为平稳的降低 Loss。他们可能适用于不同的样本，不过我还没有搞清楚。

对于不同的网络，在使用相同的优化器 SGD 时，Loss 下降的趋势比较相像，但值高低不定。从这里看来，res18 与 res34 的发挥较好一些。

本次训练只用了每种 4000 个样本，为了节约时间。在图中看来，网络性能还有继续完善的空间。

5.2.3 使用不同的 Loss Function 对系统准确度的影响



因为我们的 Loss 算法不同，只能将准确率做为衡量不同损失函数的标准。

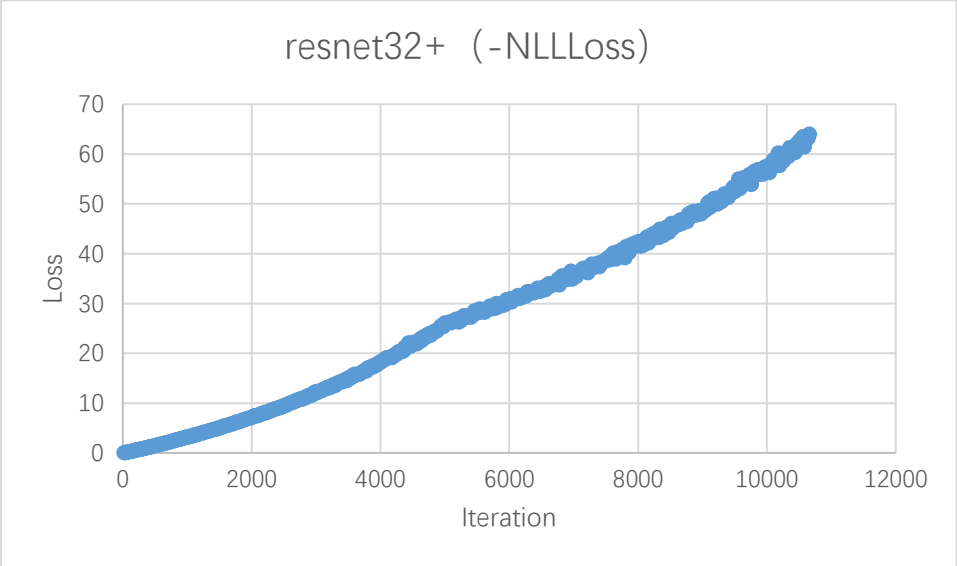
Softmax+1/x 是我自己尝试的一种计算损失函数的方法，第一步是在归一化，第二部是将置信程度改变为离谱程度。1/x 即将“置信程度“转化为“离谱程度”，1/x 放大了归一化得到的置信程度的

错误的训练

在计算的时候，不能如果采用了错误的方法，就会造成 Loss 的值不降反升。

如果不采用 lgSoftmax 直接采用 NLLLoss,就会使 Loss 从 0 附近一直降到负值。

一开始我觉得不对，没见过负的 Loss，就直接把 Loss 反号。然后使其梯度下降。这样做使不对的，因为一开始的置信度多数为正的，加了负号就变成了负的，数据会越训练越混乱。由此可见。NLLLoss 只适合在归一化后为负数时使用最合适。



6 小结

我们组在上此课程之前也没怎么接触过机器学习相关的知识吧，只是道听途说也没有自己试过。机器学习对我们来说是一个有价值却一直无从下手的课程。在种子班能有这样一个机会学习面上班学不到的机器学习是非常幸运的。我虽然暂时也没有在机器学习领域深造的鉴定信心，但我认为机器学习是一种非常实用的工具。机器学习与其他领域可以结合的很好，比如与嵌入式和机械结合起来就是无人驾驶，和医疗结合起来就是智慧医疗，与网络结合起来等等。本次虽然算是对神经网络一次浅探，但我觉得在这么短短的一段时间里知道了这么多确实已经很满足了，我知道仔细学习起来，机器学习的路还有很长很长。虽然初步探究的过程中遇到了很多麻烦，比如不知道显卡为什么莫名其妙的内存满了，Loss 为什么迟迟没有下降，为什么越训练结果越差了等等，但在仔细思考之后，请教了擅长 AI 领域的同学孙昊海大哥之后，很多问题都是有了一个似乎正确的答案，我知道这门学科不太容易对一个问题找到一个非常明确的答案，但在原理上还是可以大概说得通的。此外另要感谢组员陈列可同学对此项目付出较长时间的探索与实践，希望他一定会在 AI 之路找到更多自己想要得到的答案。