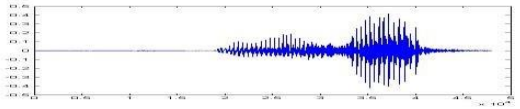
5-1. PyTorch入门

- 机器学习与深度学习
- PyTorch深度学习训练

机器学习

- 机器学习≈ 寻找一个函数
  - 语音识别

$f($  $)=$“How are you”

  - 图片识别

$f($  $)=$“Cat”

  - 下围棋

$f($  $)=$“5-5”（下一步棋）

  - 对话系统

$f($ “Hi”（你说的内容） $)=$ “Hello”（系统的回答）

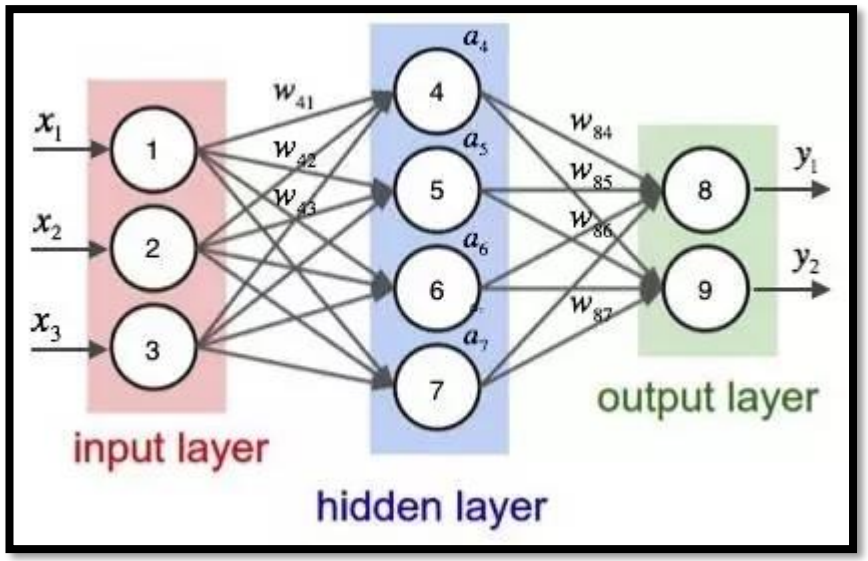
机器学习的核心是寻找f!

这里的f 可以是简单的闭式表达：

f(x) = x^2 + 2*x + 1

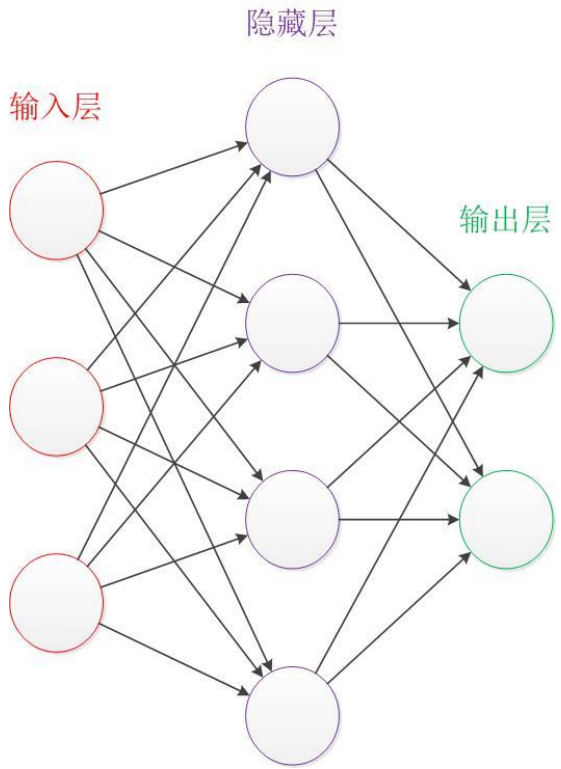
也可以是一个复杂的神经网络：

f(x1,x2,x3) =



神经网络

- 让我们来看一个经典的神经网络。这是一个包含三个层次的神经网络。红色的是**输入层**，绿色的是**输出层**，紫色的是**中间层**（也叫**隐藏层**）。输入层有3个输入单元，隐藏层有4个单元，输出层有2个单元。后文中，我们统一使用这种颜色来表达神经网络的结构。



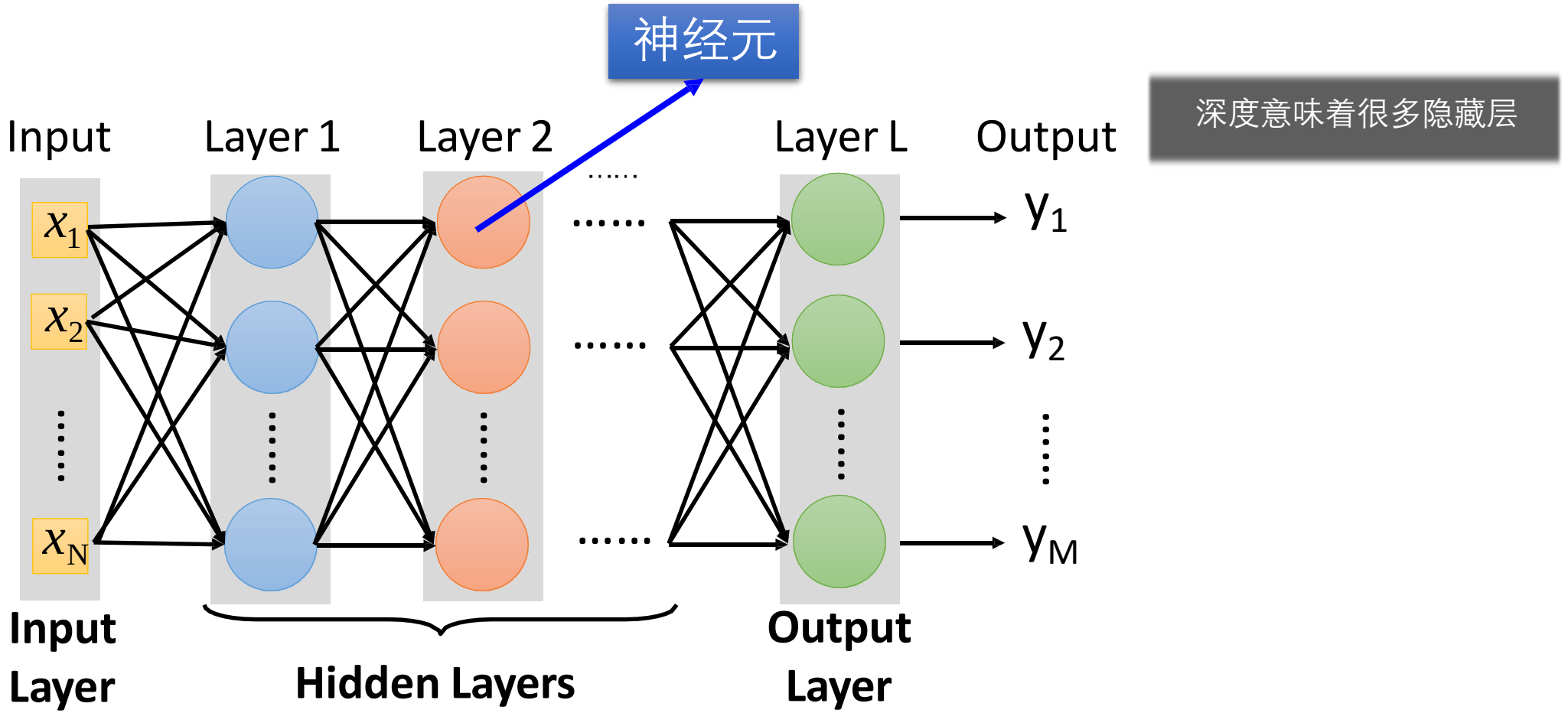
1. 设计一个神经网络时，输入层与输出层的节点数往往是固定的，中间层则可以自由指定；

2. 神经网络结构图中的拓扑与箭头代表着**预测**过程时数据的流向。

3. 结构图里的关键不是圆圈（代表“神经元”），而是连接线（代表“神经元”之间的连接）。每个连接线对应一个不同的**权重**（其值称为**权值**），这是需要训练得到的。

深度学习

- 现代神经网络的层数非常多（可能有50层甚至几百层）
- 对这种多层神经网络的训练，可以认为就是深度学习

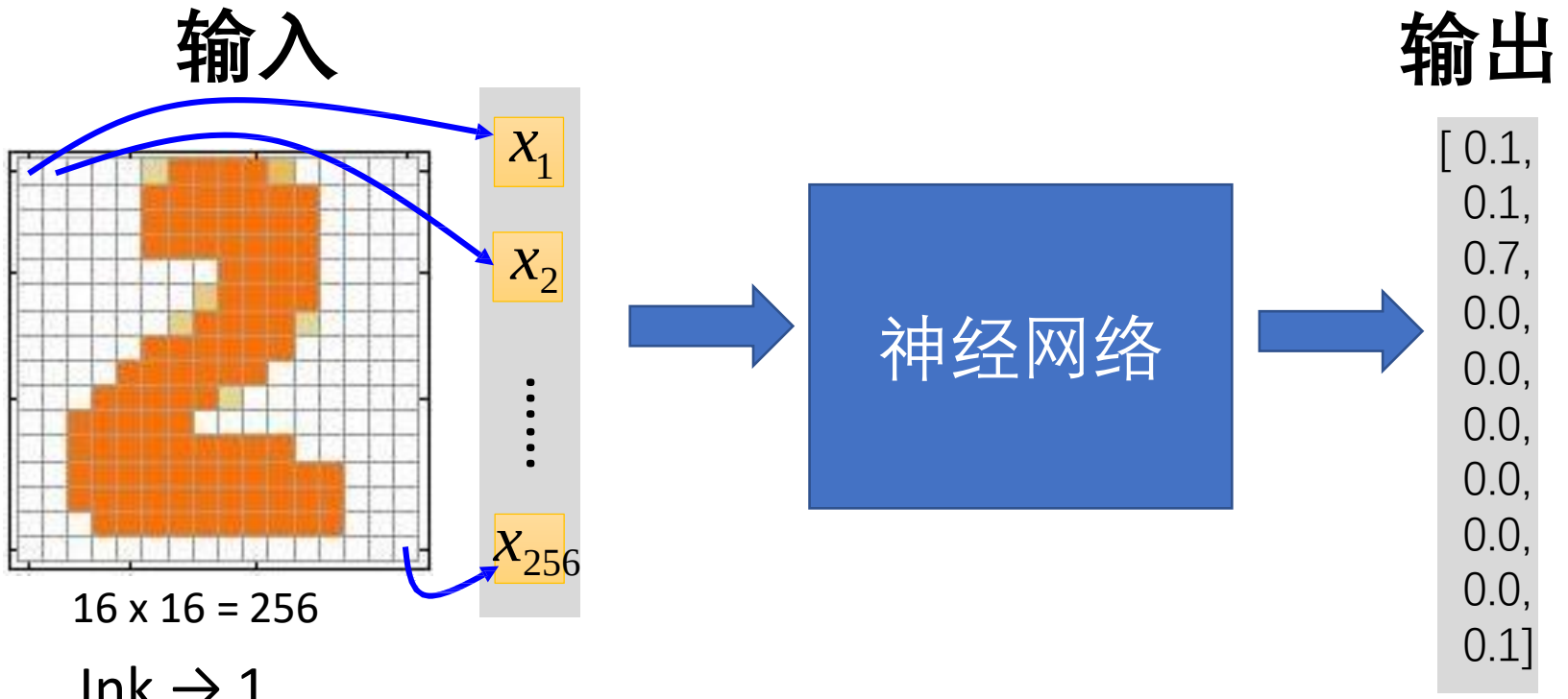


深度意味着很多隐藏层

深度学习

可以看出：深度学习是机器学习的一个子领域。

以手写数字的图像分类为例：输入图片可以看作一个256维的向量



Ink → 1

No ink → 0

输出是一个10维的向量，分别表示属于第i类的概率。
本例中，神经网络认为10%的可能性为“0”，10%的可能性为“1”，
70%的可能性为“2”，10%的可能性为“9”。即，模型认为输入图片最有可能表示“2”

模型训练

- **损失（loss）：神经网络输出和目标之间的距离**

以上文数字”2”为例子，

其输出是 output = [0.1,0.1,0.7,0,0,0,0,0,0.1]

目标是 target = [0,0,1,0,0,0,0,0,0,0] # 以100%的概率认为是”2”

计算这两者之间的L2损失函数：

Loss = $\sum_{i=0}^{9}$ (output_i – target_i)^2

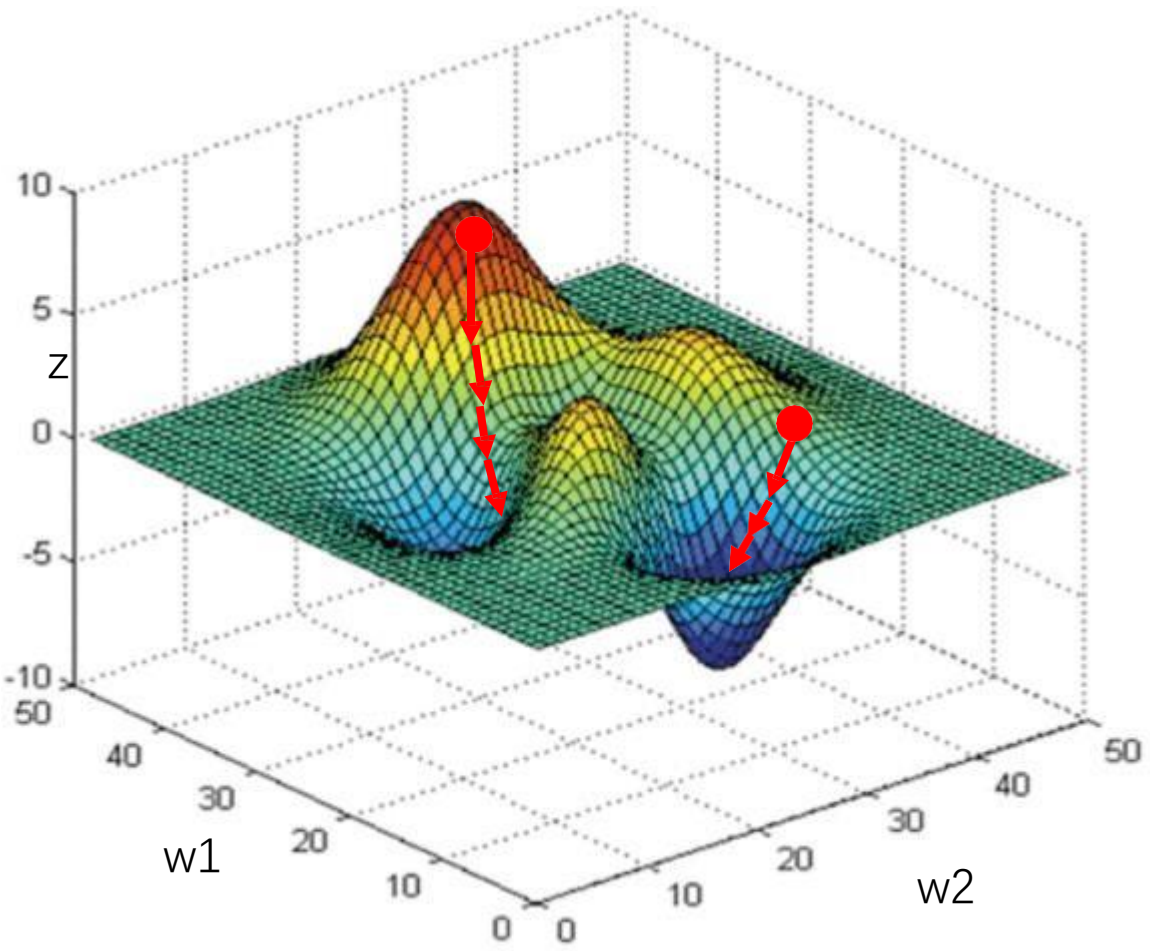
显然，loss=0最佳

- 此外，还有CrossEntropyLoss、FocalLoss等，计算方式有所不同，在此不做介绍。

模型训练

- 损失的计算，可以指导模型参数的修正方向（红色箭头）：

右图表示了loss与模型参数w1、w2取值之间的关系。
z轴表示损失函数的值，w1、w2轴表示模型的参数。
我们期望损失函数最小，则需要告知模型参数是该增大还是减小（图中红色箭头），使得loss尽可能降低



- 学习率(learning rate，LR)：规定了参数修正的幅度。若损失函数要求w1、w2变小，则LR可调节w1、w2是应该减小0.1、1.0、还是10，即“迈的步子该多大”。
- LR过小会导致loss下降不显著，LR过大会导致“走过头”，无法到达谷底

深度学习框架

- 有很多工具可以帮助你快速实现深度学习

- 本实验采用目前最火热的PyTorch。PyTorch已经成为科研人员的首选深度学习框架之一，在人工智能、机器学习、计算机视觉、自然语言处理相关领域学术论文中的使用率不断增长。

实验准备

- 到这里，我们初步了解了神经网络和深度学习。更多参考资料：

https://www.cnblogs.com/subconscious/p/5058741.html

https://zhuanlan.zhihu.com/p/88399471

https://www.zhihu.com/question/26006703/answer/536169538

- 接下来我们将尝试使用PyTorch进行深度学习。

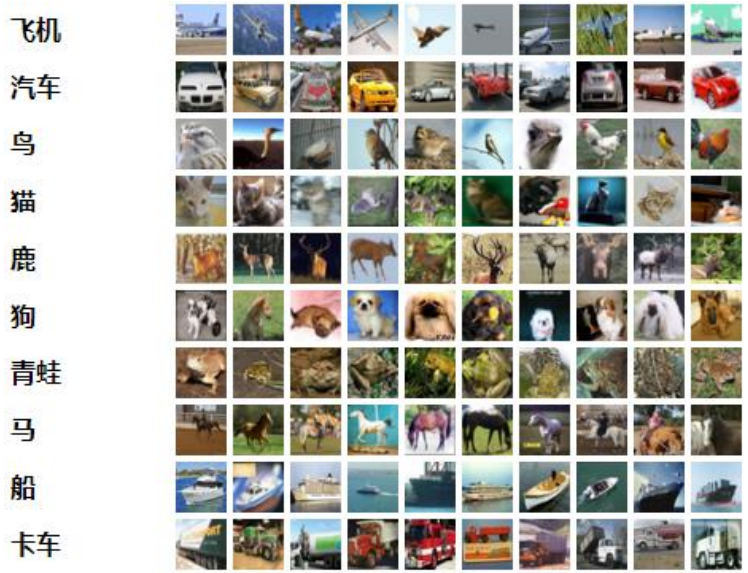
- 别担心，如果你对上述内容还有疑问，应该不会对接下来的实验有太多影响。只需记住：深度学习并不神秘，神经网络也只是大量节点的加权求和。

实验准备: 名词解释

- Training set： 训练集，所有的训练数据
- Epoch：训练轮次。深度学习需要对training set遍历多遍，每一遍叫做一个训练轮次。
- Batch：批。在**每个训练轮次**中，由于训练集可能很大，无法一次性放入神经网络计算得到prediction，因此需要分批输入。在每个epoch中，若每次将128张图片输入网络，则认为 batch size =128。每128张图片构成一个batch。

实验5.1 CIFAR-10图片分类

- 利用深度学习对CIFAR-10（ http://www.cs.toronto.edu/~kriz/cifar.html ）图片进行分类。（见exp2.py）
- CIFAR-10数据集由10个类的60000个32x32彩色图像组成，每个类有6000个图像。有50000个训练图像和10000个测试图像。
- 注意：训练时，我们只能使用50000个训练图像。这样，在测试时模型没见过这10000张测试图片，其准确率评估才准确



实验5.1 CIFAR-10图片分类

- 本实验中，我们采用简单的ResNet20 （一个20层的神经网络）作为模型，在50000个训练图片上对其参数进行更新。经过10轮训练后，在10000张测试图片上的准确率可以达到70%以上。
- 事实上，现在最好的模型（不是使用ResNet20）可以达到95%以上的准确率，这一数字远远好于随机猜测的准确率——10%. 当然，这些模型训练的轮数远多于10轮，模型也要比resnet20复杂。
- 一个多年前的排行榜：https://www.kaggle.com/c/cifar-10/leaderboard

实验5.1 CIFAR-10图片分类

练习：

- 补充exp2.py的代码，使得程序可以完整运行。
- 使用resnet20模型，训练一个cifar-10的分类器。（推荐训练策略：以0.1的学习率(learning rate, lr)训练5个epoch，再以0.01的lr训练5个epoch。）

提示：使用model.load_state_dict(torch.load(restore_model_path)['net'])可以加载已保存的模型，继续训练。训练可能会需要很久的时间，请耐心等待。若笔记本性能有限，可以加载我们已经预训练了5轮的pretrain_model.pth，这样可以跳过lr=0.1的训练阶段，直接进行lr=0.01的5轮训练。

（可选）思考：Train acc 和 Test acc有什么关联和不同? 在lr从0.1变到0.01后，acc发生了什么变化？为什么?

提交要求：

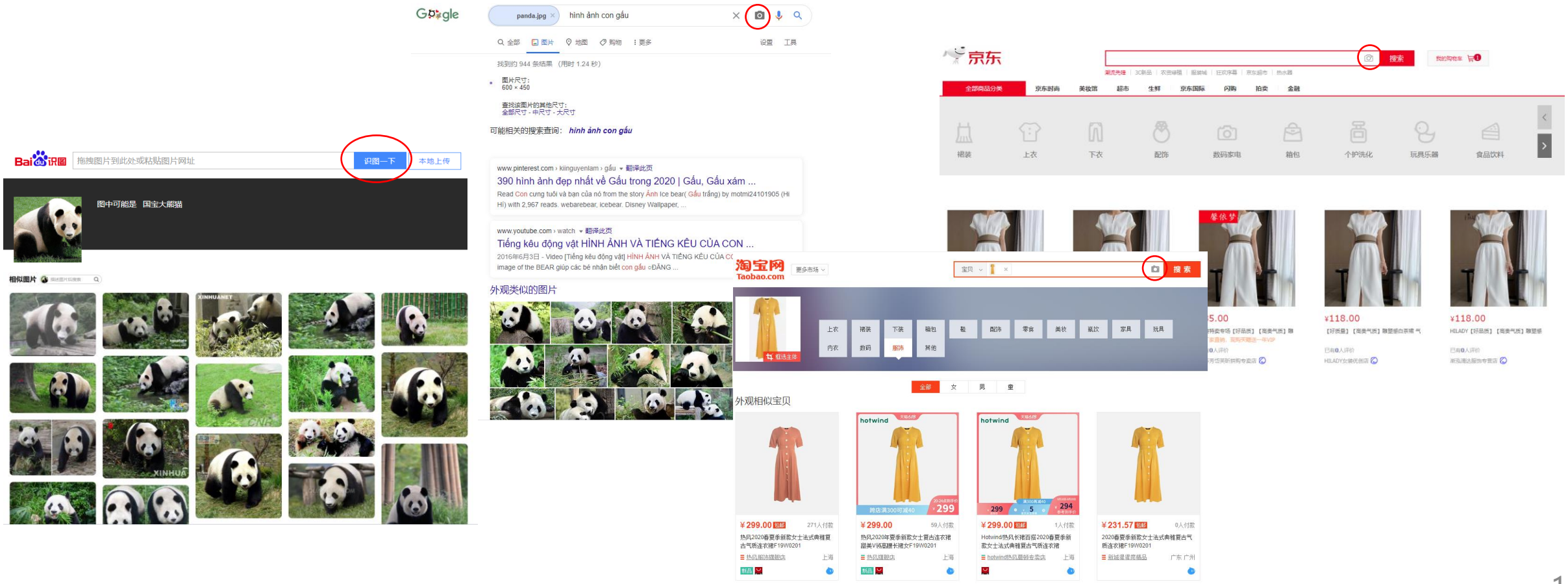
- 补充完整的exp2.py（使得算法可以按照给定的推荐训练策略,从第0个epoch开始自动完成训练），以及最终模型 final.pth（评分时会读入此模型查看测试准确率）。
- 在报告中展示test acc变化趋势，以各种可能的方式使最终的测试准确率尽可能高。
- 提交时请务必将cifar-10数据集删掉！

5-2. Search by CNN features

图像检索
卷积神经网络
使用Pytorch提取图像特征

图像检索

- 早期的图像检索技术研究主要是基于文本的图像检索(TBIR)，通过对图片的文本描述来查找图像库中具有相应标签的图像。随着技术的发展，基于内容的图像检索(CBIR)，即以图搜图也成为了各大网站的标配功能。



图像检索

- 以图搜图是如何实现的呢？以图搜图的核心其实就是在被检索的图像库(Library)中找到和参与检索的图片(Query)最相似的图片集合。那么问题就变成了如何定义图片之间的相似程度。

Query

a

b

c

d

e

f

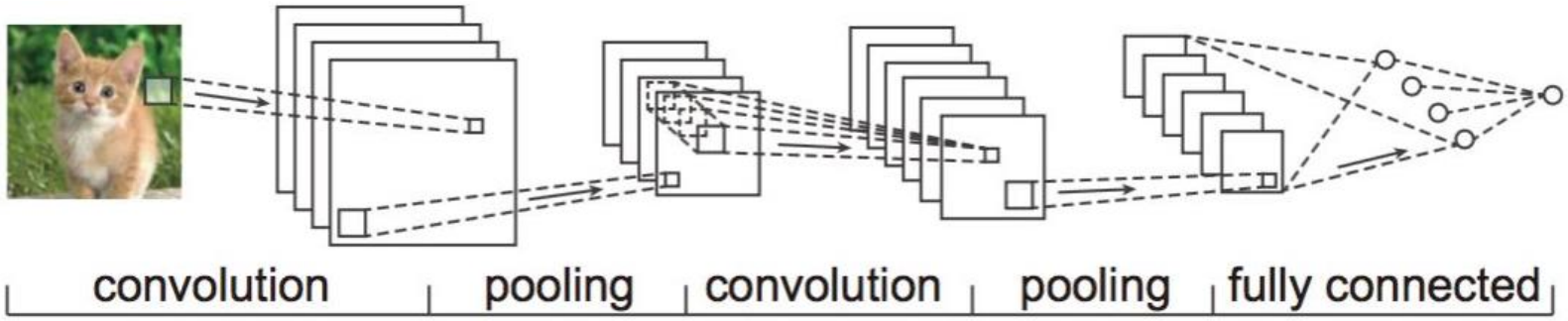
Library

图像检索

- 对于上页ppt中的例子，与待检索图像“Query”最相近的应该是图像“f”；但是图像库中的其他图片与“Query”的相似程度应该怎么排序呢？
- 如果从考虑图像颜色分布的情况来看，图像“b”和“e”也会有较高的相似程度；如果考虑交通工具这个范畴，图像“c”则会有较高的排序。
- 所以如果我们能从图像的内容中提取出相应的语义信息，如颜色，纹理，种类等等，那么这个以图搜图的检索任务就自然得到了解决。
- 深度学习模型通过一种端到端学习的范式，使得模型能够学习到图像的这种特征。所以我们可以使用模型来将图像转换成一个能够代表图像的向量，从而用向量之间的相似程度代表图像之间的相似程度，从而解决以图搜图的任务。

图像特征的提取

- 传统的图像特征提取有很多的方法，如SIFT，HOG等，我们在前面课程已经做过相关实验。
- 通过上一节实验课的学习，大家应该了解了一些深度学习的入门知识和 pytorch 的使用方法。本节课我们将使用深度模型提取图像的特征，从而来帮助我们构建以图搜图的搜索引擎。



卷积神经网络

- 上节实验课我们简单介绍了什么是神经网络，并且使用了一种卷积神经网络ResNet20在Cifar10数据集上训练了一个图像分类的模型。现在我们来介绍一下什么是卷积神经网络。
- 卷积神经网络是一类包含卷积计算且具有深度结构的前馈神经网络（Feedforward Neural Networks），是深度学习（deep learning）的代表算法之一。其在图像处理的领域有着非常广泛的应用。
- 更详细的介绍请阅读下面的链接（希望大家都能认真学习一下，便于对后续实验内容的理解）。
- https://cs231n.github.io/convolutional-networks/

使用Pytorch提取图像特征

```
import torch
print(torch.hub.list('pytorch/vision'))
```

导入torch模块，看看pytorch官方给我们提供了哪些现成的深度模型。

```
Using cache found in /root/.cache/torch/hub/pytorch_vision_master
['alexnet', 'deeplabv3_resnet101', 'deeplabv3_resnet50', 'densenet121', 'densenet161', 'densenet169', 'densenet201', 'fcn_resnet101', 'fcn_resnet50', 'googlenet', 'inception_v3', 'mnasnet0_5', 'mnasnet0_75', 'mnasnet1_0', 'mnasnet1_3', 'mobilenet_v2', 'resnet101', 'resnet152', 'resnet18', 'resnet34', 'resnet50', 'resnext101_32x8d', 'resnext50_32x4d', 'shufflenet_v2_x0_5', 'shufflenet_v2_x1_0', 'squeezenet1_0', 'squeezenet1_1', 'vgg11', 'vgg11_bn', 'vgg13', 'vgg13_bn', 'vgg16', 'vgg16_bn', 'vgg19', 'vgg19_bn', 'wide_resnet101_2', 'wide_resnet50_2']
```

```
alexnet = torch.hub.load('pytorch/vision', 'alexnet', pretrained=True)
print(alexnet)
```

打印出模型的信息，通过之前对于卷积神经网络的学习，看看alexnet模型是怎么搭建起来的。pretrained参数为True则下载官方在ImageNet数据集上预训练好的模型参数。

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

同学们可以用这种方法查看其他的模型都是如何搭建起来的，比如我们之后用到的ResNet50模型。

alexnet网络参考论文
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

使用Pytorch提取图像特征

请大家参考给出的示例代码extract_features.py，学习如何用pytorch提取示例图像panda.jpg的特征。直接运行程序得到如下结果。

```
root@cad54f9508a7:/workspaces/hello_world/data# python extract_feature.py
Load model: ResNet50
Using cache found in /root/.cache/torch/hub/pytorch_vision_master
Prepare image data!
Extract features!
Time for extracting features: 0.27
Save features!
```

Ps：第一次运行程序，会自动下载预训练好的模型，请保持网络的通畅。在cpu上，用ResNet50模型提取一张图片大概要0.3s，不同的电脑配置可能有所差异.
如果预训练模型不能正常加载，可以不用预训练。

代码解读

```
print('Load model: ResNet50')
model = torch.hub.load('pytorch/vision', 'resnet50', pretrained=True)
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])
trans = transforms.Compose([ transforms.Resize(256),
                             transforms.CenterCrop(224),
                             transforms.ToTensor(),
                             normalize])
```

导入ResNet50模型，可以加载预训练参数，定义处理图片的归一化操作和预处理方式。

```
print('Prepare image data!')
test_image = default_loader('panda.jpg')
input_image = trans(test_image)
input_image = torch.unsqueeze(input_image, 0)
```

读入图片，并使用之前定义好的处理方式(trans)处理图片，为送入模型提特征做准备。

使用Pytorch提取图像特征

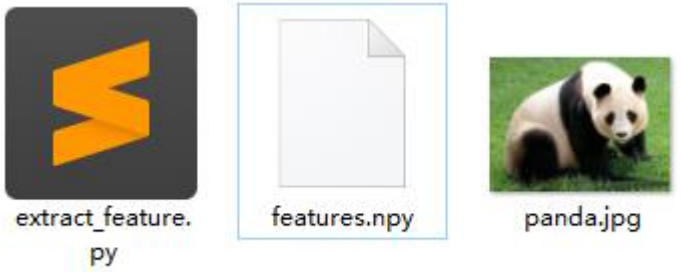
```
def features(x):
    x = model.conv1(x)
    x = model.b1(x)
    x = model.relu(x)
    x = model.maxpool(x)
    x = model.layer1(x)
    x = model.layer2(x)
    x = model.layer3(x)
    x = model.layer4(x)
    x = model.avgpool(x)
    return x
```

定义函数features()用来提取图像的特征。正常调用模型 `model(input_image)` ，则会返回ResNet50模型在ImageNet数据集上的最终分类结果。通常来讲，我们可以把模型最终分类的前一层当作模型学习到的图像特征，并用这种特征来完成我们的图像检索任务。
所以我们需要重新定义一个函数features()来提取模型倒数第二层的输出结果。请同学们结合最开始print的模型信息思考函数features()为什么要这么写。如果换成其他的模型又该怎么完成features()这个函数。

```
print('Extract features!')
start = time.time()
image_feature = features(input_image)
image_feature = image_feature.detach().numpy()
print('Time for extracting features: {:.2f}'.format(time.time()-start))

print('Save features!')
np.save('features.npy', image_feature)
```

调用函数features()完成示例图像panda.png特征的提取，并将特征保存下来。



图像检索

向量A　　向量B　　向量C

保存得到的特征为一个向量，如何计算向量之间的相似程度呢？

方法一：首先将向量归一化，计算向量之间的欧氏距离，通过距离的远近来反映向量之间的相似程度。

方法二：通过计算向量之间的夹角，夹角越大则越不相似，反之越相似。

练习

- 本次实验要求构建一个不小于50张的图像库，用不仅限于Resnet50的模型提取图像的特征，并使用上一页PPT提到的方法，计算图片之间的相似度。
- 请使用一些不在图像库中的图片进行测试，完成以图搜图的检索任务。
- 报告中要求给出检索的结果，与被检索图片Top5相似的图片和排序，说明排序的方法，给出排序的得分情况。