3.1 卷积神经网络(CNN)简介

多层的线性网络和单层的线性网络没有区别，而且线性模型的能够解决的问题也是有限的

3.1.2 激活函数的选择

- tanh 函数（the hyperbolic tangent function，双曲正切函数）：
  - 注 :tanh 函数存在和 sigmoid 函数一样的缺点：当 z 趋紧无穷大（或无穷小），导数的梯度（即函数的斜率）就趋紧于 0，这使得梯度算法的速度会减慢。
- ReLU 函数（the rectified linear unit，修正线性单元）
  - 当 z > 0 时，梯度始终为 1，从而提高神经网络基于梯度算法的运算速度，收敛速度远大于 sigmoid 和 tanh。然而当 z < 0 时，梯度一直为 0，但是实际的运用中，该缺陷的影响不是很大。

3.1.2.1 为什么需要非线性的激活函数

没有激活函数，那么无论神经网络有多少层，**输出都是输入的线性组合**

卷积神经网络: 图像

循环神经网络，LSTM网络：自然语言处理运用

- 目的：减少网络参数数量，达到更好效果

卷积神经网络的由来?

- **1979年，Kunihiko Fukushima（福岛邦彦），提出了Neocognitron， 卷积、池化的概念基本形成。**
- 1989年，Yann LeCun提出了一种用反向传导进行更新的卷积神经网络，称为LeNet。

3.1.2 感受野

单个感受器与许多感觉神经纤维相联系

3.1.4 边缘检测

为了能够用更少的参数，检测出更多的信息，基于上面的感受野思想。通常神经网络需要检测出物体最明显的垂直和水平边缘来区分物体

- 过滤器去过滤图片：得到水平边缘和垂直边缘一些物体形状结论

3.2 卷积神经网络(CNN)原理

3.2.1 卷积神经网络的组成

- **一个或多个卷积层、池化层以及全连接层等组成**

- **卷积层（Convolutions）**
- **池化层（Subsampling）**
- 全连接层（Full connection）

3.2.2 卷积层

- 目的
  - 卷积运算的目的是提取输入的不同特征
- 运算过程
  - 1、5，5 使用3 x3 filter过滤运算，得到3 x 3结果
  - 移动步长，观察结
  - 缺点
    - 图像变小
    - 边缘信息丢失
    - 3.2.3 padding-零填充
  - 零填充：在图片像素的最外层加上若干层0值，若一层，记做p =1
  - 因为0在权重乘积和运算中对最终结果不造成影响，也就避免了图片增加了额外的干扰信息。

- **3.2.3.1 Valid and Same卷积**
  - Valid :不填充，也就是最终大小为
    - (N - F + 1) * (N - F + 1)$(N-F+1)*(N-F+1)$
  - Same：输出大小与原图大小一致，那么 NN变成了N + 2P$N+2P$

3.2.3.2 奇数维度的过滤器大小选择问题

- 1 x 1，3 x 3， 5 x 5，7 x 7， 9x9

3.2.4 stride-步长

增加步长的影响因素之后

- (N+2P-F)/s + 1观察结果大小
- SAME:输入大小与出书大小一致
  - (N+2P-F)/s + 1 = N
  - P = ?

对于多通道图片来讲？如何去观察？

3.2.5 多通道卷积

当输入有多个通道（channel）时(例如图片可以有 RGB 三个通道)，卷积核需要拥有相同的channel数，每个卷积核 channel 与输入层的对应 channel 进行卷积，将每个 channel 的卷积结果按位相加得到最终的 Feature Map。

3.2.5.1 多卷积核（多个Filter）

- 输出feature map的个数

- 多个功能的卷积核的计算结果放在一起，能够检测到图片中不同的特征（边缘检测）

3.2.6 卷积总结

- 公式

3.2.7 池化层(Pooling)

- 减少图片的特征数量，避免全连接层参数过多
- **最大池化：Max Pooling,取窗口内的最大值作为输出**
- 平均池化：Avg Pooling,取窗口内的所有值的均值作为输出
- 通用池化层的窗口大小和步长
  - f=2∗2,s=2
- 池化层的一个作用

全连接层

- 先对所有 Feature Map 进行扁平化（flatten, 即 reshape 成 1 x N 向量）
- 再接一个或多个全连接层，进行模型学习

2.2案例：CIFAR100类别分类

- 进行模型编写
  - 两层卷积层+两个神经网络层
  - 网络设计：
- 设计步骤
  - 读取数据集
  - 编写两层+两层全连接层网络模型
    - keras.layers.Conv2D(32, kernel_size=5, strides=1, padding="same", data_format="channels_last", activation=tf.nn.relu)
      - 第一个参数：filter数量
    - keras.layers.MaxPool2D(pool_size=2, strides=2, padding="same")
      - padding="same"：没有起到相同大小作用，28 --->14
  - 编译、训练、评估

3.4 深度学习正则化

3.4.1 偏差与方差

3.4.1.2 偏差与方差的意义

- **偏差**：度量了学习算法的期望预测与真实结果的偏离程度，即刻画了**学习算法本身的拟合能力**
  - 算法的差异
- **方差**：度量了同样大小的训练集的变动所导致的学习性能的变化，即刻画了**数据扰动所造成的影响**

- 数据集的差异

训练集的错误率较小，而验证集/测试集的错误率较大，说明模型存在较大方差，可能出现了过拟合

出现过拟合？该怎么办？

3.4.2 正则化(Regularization）

- 正则化概念：
  - **简单的模型比复杂的泛化能力好**
  - **正则化，即在成本函数中加入一个正则化项(惩罚项)，惩罚模型的复杂度，防止网络过拟合。**
- L1与L2
  - **由于 L1 正则化最后得到 w 向量中将存在大量的 0，使模型变得稀疏化，因此 L2 正则化更加常用。**
  - L2结果W趋近于0
- **3.4.2.2 正则化项的理解**
  - 为什么达到效果
    - 因为加入L2正则化，W权重在更新时候，减小更快，减小的效果就是求Z=wx+b，Z就变小，求激活函数导数的时候，梯度就不会非常小
    - 模型变得更加简单，不会发生过拟合
- **3.4.2.4 tf.keras正则化API**
  - keras.layers.Conv2D(32, kernel_size=5, strides=1, padding="same", data_format="channels_last", activation=tf.nn.relu, kernel_regularizer=keras.regularizers.l2(0.01)),
  - 目的不是为了让训练更好，而是在训练很好的条件下97%，在测试集表现不好89%,有可能模型出现了过拟合，需要在原始网络当中增加正则化参数（L2正则化）
- dropout正则化
  - Droupout：随机的对神经网络每一层进行丢弃部分神经元操作。
  - 即keep_prob。假设keep_prob为0.8，保留神经元的比例
  - 对于好多层的网络，不同层丢弃比率不一样
  - **3.4.3.1 Inverted droupout**
  - **3.4.3.2 droupout为什么有效总结**
  - 调试时候使用技巧：
    - dropout 的缺点是成本函数无法被明确定义，保证损失函数是单调下降的，确定网络没有问题，再次打开droupout才会有效。
  - keras.layers.Dropout(0.2)
    - 指定丢失率
- 早停止法和数据增强
  - **3.4.4.1 早停止法（Early Stopping）**
  - **3.4.4.2 数据增强**

    - 目的：通过多种组合变换达到增加数据集大小
    - 两种形式
        - 离线增强。预先进行所有必要的变换，从根本上增加数据集的规模（例如，通过翻转所有图像，保存后数据集数量会增加2倍）。
        - 在线增强，或称为动态增强。可通过对即将输入模型的小批量数据的执行相应的变化，这样同一张图片每次训练被随机执行一些变化操作，相当于不同的数据集了。
    - 从数据集的角度

2.4 经典分类网络结构

2.4.1 LeNet-5解析

- LeNet：将近6万参数
- ALexNet:2012年，Alex Krizhevsky、Ilya Sutskever在多伦多大学Geoff Hinton的实验室设计出了一个深层的卷积神经网络AlexNet，
    - 输入：227 x 227 *3
    - 60M=6000万，5层卷积+3层全连接
    - 使用ReLU+Dropout
- AlexNet—NIN—(VGG—GoogLeNet)—ResNet
- VGG
    - 1.4亿参数量，输入224 * 224 * 3
- GoogleNet
    - 500万参数量

2.4.4 Inception 结构

- 减少参数使用量
- Network in Network中引入的1 x 1卷积结构的相关作用
- **重要作用：1x1的卷积核操作还可以实现卷积核通道数的降维和升维，实现参数的减小化**
- 减少参数数量
    - (5 * 5 * 192) * 32，　(1 * 1 * 192)　* 32

2.4.4.4 Inception层

- 建立网络中的网络，在其中增加1*1卷积结构
- 四中结构：1 * 1， 3 * 3， 5 * 5，maxpool
- 最终结果28 x 28 x 256
    - 使用更少的参数，达到跟AlexNet或者VGG同样类似的输出结果
- 改进：
    - 把inception当中的结构5* 5，拆分成1* 1和5 * 5组合，又能减少数量

layer1,layer2学习到的特征基本是颜色、边缘等低层特征

layer3学习到的特征，一些纹理特征，如网格纹理

layer4学习到的特征会稍微复杂些，比如狗的头部形状

layer5学习到的是完整一些的，比如关键性的区分特征

2.4.6 案例：使用pre_trained模型进行VGG预测

- 模型获取以及已训练好的参数加载
  - model = VGG16() print(model.summary())
- 图片读取处理
  - load_img, img_to_array
- 模型进行预测
  - image = preprocess_input(image) y_predictions = model.predict(image)

```
print(y_predictions)
# 对结果进行解码,按照排序结果
label = decode_predictions(y_predictions)
print("预测的类别是：%s, 并且预测概率为：%f" % (label[0][0][1], label[0][0][2]))
```

- 预测的条件：Google在用VGG训练ImageNet比赛当中的1000个类别才能预测
  - 特定场景的图像识别任务，必须训练自己的模型进行预测
  - 可以在VGG的基础之上做训练，节约训练时间，包括效果都会得到改善
  - 迁移学习

2.4 BN与神经网络调优

2.4.1 神经网络调优

2.4.2 批标准化（Batch Normalization）

- 根本上不是去优化模型，而是帮助我们能够更好的去 训练简单，训练过程，节省时间
- 解决目的：**内部协变量偏移**
- 对于深层网络一些层级输出，进行批标准化

2.4.2.1 批标准化公式

- 1、所以假设对于上图第二个四个神经元隐层。记做Z^{[l]}Z[/]，对于每一层的输出，进行标准化操作
- 2、在标准化之后，增加一个状态分布参数，让标准数据进行修改分布
  - garma, beta
- 作用：

    - 1、BN减少不同数据分布状态带来的影响，模型鲁棒性强，测试准确率高，防止过拟合的作用
    - 2、BN使得不同层学到不同的分布状态
    - 3、**减少了各层 W 和 b 之间的耦合性，让各层更加独立，实现自我训练学习的效果**
- 总结：
    - 比如：学习率0.1或者使用0.0001,都能学习到很好的模型
        - 没有BN可能有些模型只适用0.0001这个学习率
        - 给大学习率，学习就快