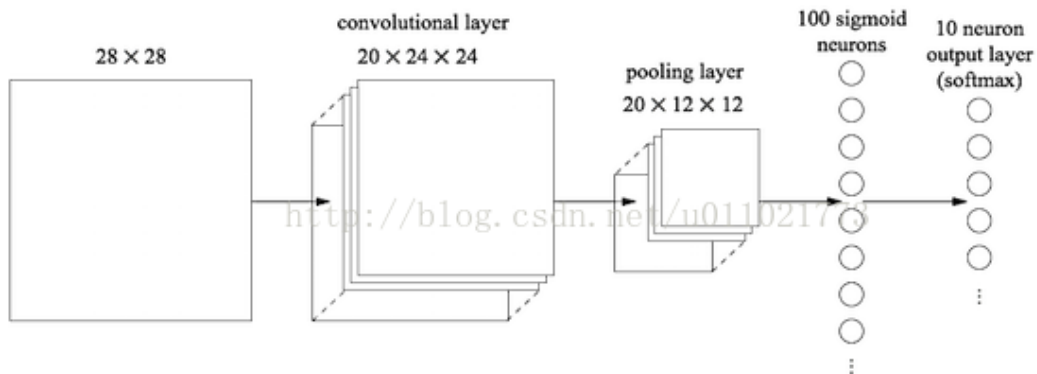


前期准备

1. 什么是全连接层

- 我们把所有神经元都会和下一层的所有神经元相连的这种连接，叫做 **全连接(Fully-connected layer)**
- 全连接一般会把卷积输出的二维特征图转化成一维的一个向量,把一张图高度浓缩成一个数了
- 全连接的目的是什么呢？因为传统的网络我们的输出都是分类，也就是几个类别的概率甚至就是一个数--类别号，那么全连接层就是高度提纯的特征了，方便交给最后的分类器或者回归



- [解释一下全连接层](#)

2. 一些英文

1. **Stride**->步长
2. **Filter**->卷积核
3. **Padding**->填充
4. **Kernel size**->卷积核的大小
5. **Normalization layer**->归一层
6. **Convolutional layer**->卷积层
7. **Fully-Connected layer**->全连接层

3. 一些解释

1. 为什么要用卷积运算

卷积运算的目的是提取输入的不同特征，第一层卷积层可能只能提取一些低级的特征如边缘、线条和角等层级，更多层的网络能从低级特征中迭代提取更复杂的特征。

2. 为什么要用激活函数

3. 为什么要有池化层

通常在卷积层之后会得到维度很大的特征，将特征切成几个区域，取其最大值或平均值，得到新的、维度较小的特征。

4. 为什么要用全连接层

把所有局部特征结合变成全局特征，用来计算最后每一类的得分。

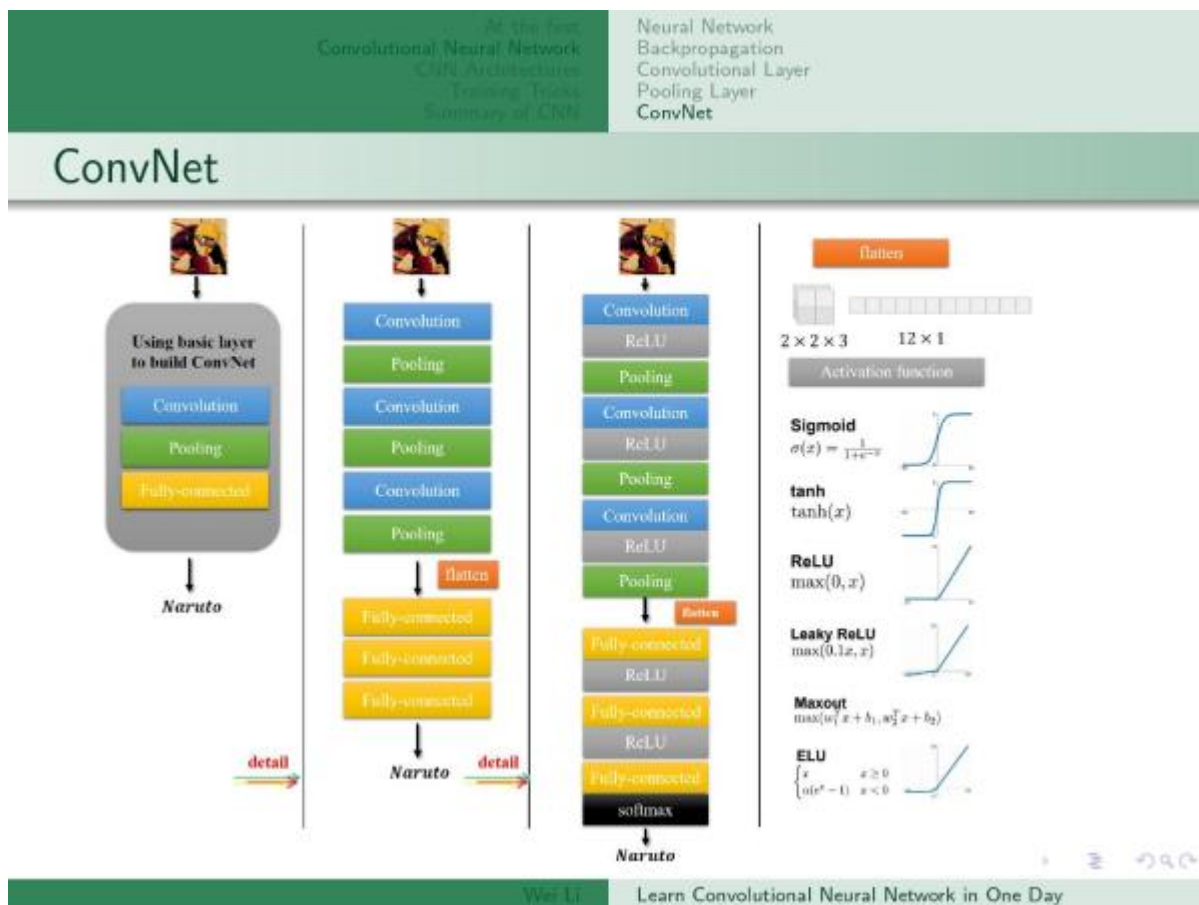
5. 数据预处理

1. [资料1](#)

6. 训练速率

如果设置的学习速率太小，你的模型可能需要几年才能收敛;如果学习速率太大，在开始训练几个样本之后，你的损失值(loss)可能会迅速增加。一般来说，0.01 的学习速率是安全的

4. 搭建一个神经网络一般的步骤



5. 典型神经网络训练过程

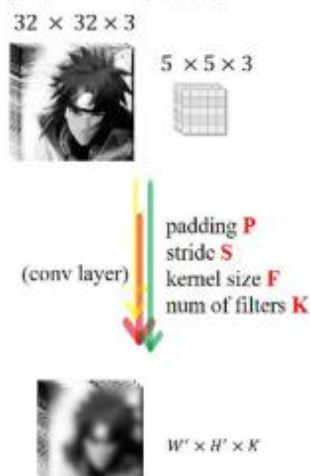
- 定义一个有着可学习的参数（或者权重）的神经网络
- 对着一个输入的数据集进行迭代: `for step, (inputs, labels) in enumerate(train_loader)`
- 用神经网络对输入进行处理 `output = cnn(inputs)`
- 计算**Loss** (对输出值的修正到底有多少) `loss_func = nn.CrossEntropyLoss()`
- 梯度归零 `optimizer.zero_grad()`
- 将梯度传播回神经网络的参数中 `loss.backward()`
- 更新网络中的权重 `optimizer.step()`

通常使用简单的更新规则(SGD): `weight = weight + learning_rate * gradient`

6. 如何计算卷积后图片大小(尺寸)

Convolutional Layer(cont.)

Hyperparameters(P,S,F,K)



$$W = 32, H = 32, D = 3, F = 5$$

Padding: $P = 0$
Stride: $S = 1$

$$W' = \frac{32 - 5}{1} + 1 = 28$$

28 × 28

Padding: $P = 0$
Stride: $S = 2$

$$W' = \frac{32 - 5}{2} + 1 = 14$$

14 × 14

Padding: $P = 2$
Stride: $S = 1$

$$W' = \frac{32 - 5 + 2 \times 2}{1} + 1 = 32$$

32 × 32

$$W' = (W - F + 2P) / S + 1$$

$$H' = (H - F + 2P) / S + 1$$

$$D' = K$$

$W \times H \times D$ → $W' \times H' \times D'$

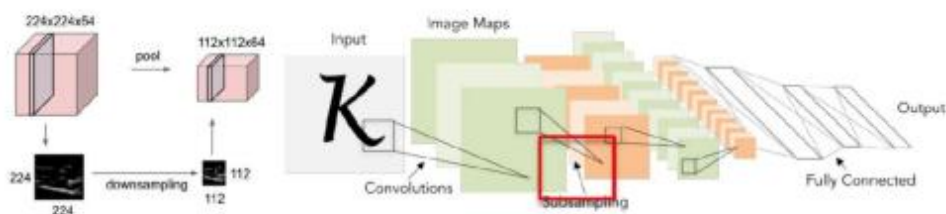
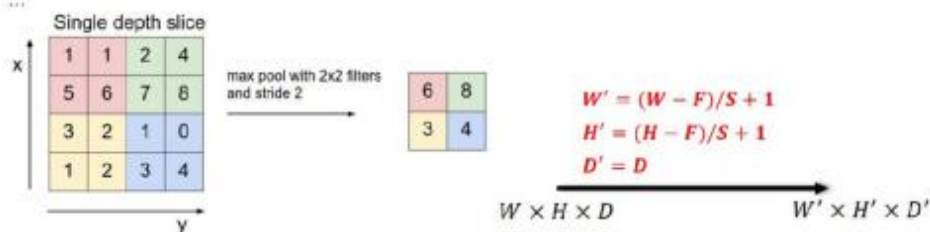
2. 图片的尺寸= (原来的尺寸-卷积核的尺寸+填充*2)/步长+1

3. 深度 = 卷积核的数量

7. 计算池化后图片的大小(尺寸)

Pooling Layer

- Max pooling
- Avg pooling
- L2-norm pooling



2. 图片的尺寸 = (原来的尺寸-卷积核的尺寸) / 步长+1
3. 深度不变
4. 为什么要把特征图摊平?

Alexnet

1. 图解

At the first
Convolutional Neural Network
CNN Architectures
Training Tricks
Summary of CNN

Basic Networks
VGG and GoogLeNet
Residual Network and Variants
DenseNet
Dual Path Networks

AlexNet - Detail

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] **CONV1**: 96 11x11 filters at stride 4, pad 0

[27x27x96] **MAX POOL1**: 3x3 filters at stride 2

[27x27x96] **NORM1**: Normalization layer

[27x27x256] **CONV2**: 256 5x5 filters at stride 1, pad 2

[13x13x256] **MAX POOL2**: 3x3 filters at stride 2

[13x13x256] **NORM2**: Normalization layer

[13x13x384] **CONV3**: 384 3x3 filters at stride 1, pad 1

[13x13x384] **CONV4**: 384 3x3 filters at stride 1, pad 1

[13x13x256] **CONV5**: 256 3x3 filters at stride 1, pad 1

[6x6x256] **MAX POOL3**: 3x3 filters at stride 2

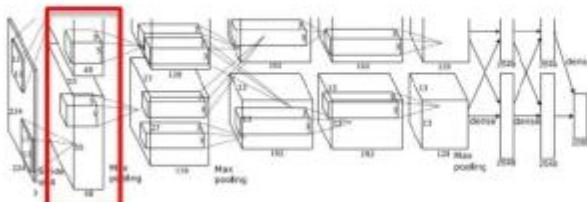
[4096] **FC6**: 4096 neurons

[4096] **FC7**: 4096 neurons

[1000] **FC8**: 1000 neurons (class scores)

Details/Retrospectives:

- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%



Historical note: Trained on GTX 580 GPU with only 3 GB of memory. Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.

$[55 \times 55 \times 48] \times 2$

We U

Learn Convolutional Neural Network in One Day

2. 各种图片尺寸的解读

1. CONV1 $55 = (227 - 11 + 2 \cdot 0) / 4 + 1 = 55$
2. MAXPOOL1 $27 = (55 - 3) / 2 + 1 = 27$
3. CONV2 $27 = (27 - 5 + 2 \cdot 2) / 1 + 1 = 27$
4. MAXPOOL2 $13 = (27 - 3) / 2 + 1 = 13$
5. MAXPOOL3 $6 = (13 - 3) / 2 + 1 = 6$

3. Alexnet为啥取得比较好的效果

1. 使用了Relu激活函数
2. Dropout

Dropout也是经常说的一个概念，能够比较有效地防止神经网络的过拟合。相对于一般如线性模型使用正则的方法来防止模型过拟合，而在神经网络中Dropout通过修改神经网络本身结构来实现。对于某一层神经元，通过定义的概率来**随机删除一些神经元**，同时保持输入层与输出层神经元的个人不变，然后按照神经网络的学习方法进行参数更新，下一次迭代中，重新随机删除一些神经元，直至训练结束。

Pytorch

1. 为什么要用torch.nn.Sequential

1. `torch.nn.Sequential` 是一个 `Sequential` 容器，模块将按照构造函数中传递的顺序添加到模块中
2. [点击这里查看更多](#)

2. 一些函数的参数介绍

1.

```
nn.Conv2d(                # input shape (1, 28, 28)
    in_channels=1,          # input height
    out_channels=16,        # n_filters
    kernel_size=5,          # filter size
    stride=1,               # filter movement/step
    padding=2 ),            # if want same width and length of this image
after con2d,                padding=(kernel_size-1)/2 if stride=1
                             # output shape (16, 28, 28)
```

2. [torch.max\(\)](#)

待解决问题

1. 如何使用已经保存了的神经网络
2. `res = conv5_out.view(conv5_out.size(0), -1) out = self.dense(res)` 这两行代码的意思
3. 为什么要用激活函数?

参考资料

1. [卷积神经网络（CNN）的理解和实现](#)
2. [一日搞懂卷积神经网络](#)
3. [Pytorch view\(\)、squeeze\(\)、unsqueeze\(\)、torch.max\(\)](#)