AlexNet

概述

AlexNet是2012年**ImageNet**竞赛冠军获得者Hinton和他的学生Alex Krizhevsky设计的。也是在那年之后，更多的更深的神经网络被提出，比如优秀的vgg,GoogleNet。

AlexNet将LeNet的思想发扬光大，把CNN的基本原理应用到了**很深很宽**的网络中。

与LeNet-5的区别

1. AlexNet有**50M**个可训练参数，而LeNet-5只有小小的**60K**个可训练参数。
2. AlexNet除了输出层，使用**Relu**作为激活函数，而Lenet-5使用**sigmoid**和**tanh**。
3. AlexNet**输出层**使用**softmax**，而LeNet-5使用**欧氏距离**作为输出。
4. AlexNet使用**重叠池化(Overlapping Pooling)**并且是**最大池化**，而Lenet-5并不是重叠池化，也不是最大池化。
5. AlexNet使用了两个**GPU**。
6. AlexNet采取了各种手段防止**过拟合**：
    - **数据增强**
    - **dropout**
    - **局部响应归一化（LRN）**

LRN

AlexNet对卷积层的输出进行LRN（局部响应归一化），公式如下：

$$b_{x,y}^{i}=\frac{a_{x,y}^{i}}{(k+\alpha\sum_{j=max(0,i-\frac{n}{2})}^{min(N-1,i+\frac{n}{2})}(a_{x,y}^{i})^{2})^{\beta}}$$

其中，$a_{x,y}^{i}$是第i个卷积核在（x,y）位置上的输出，n是相邻的n个卷积核，N是该层卷积核的总数，k,n,α,β都是超参数。

论文中采用：k=2，n=5,$\alpha=10^{-4}$,$\beta=0.75$。

整体架构

AlexNet由5个卷积层，3个全连接层所构成。

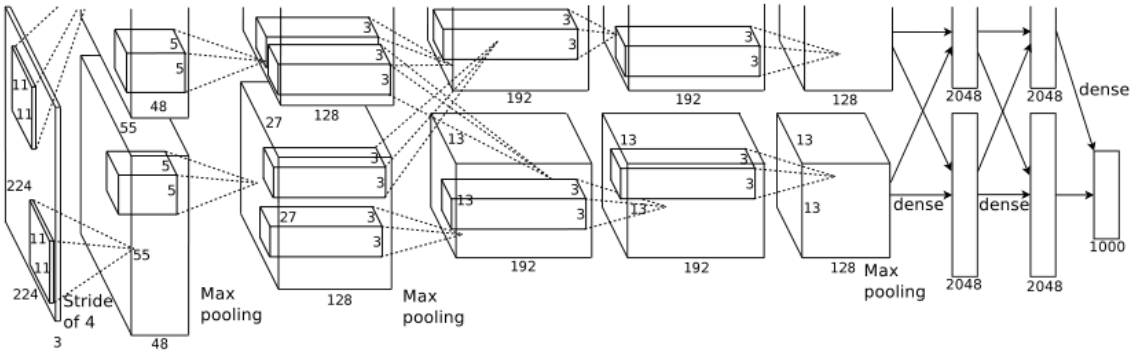
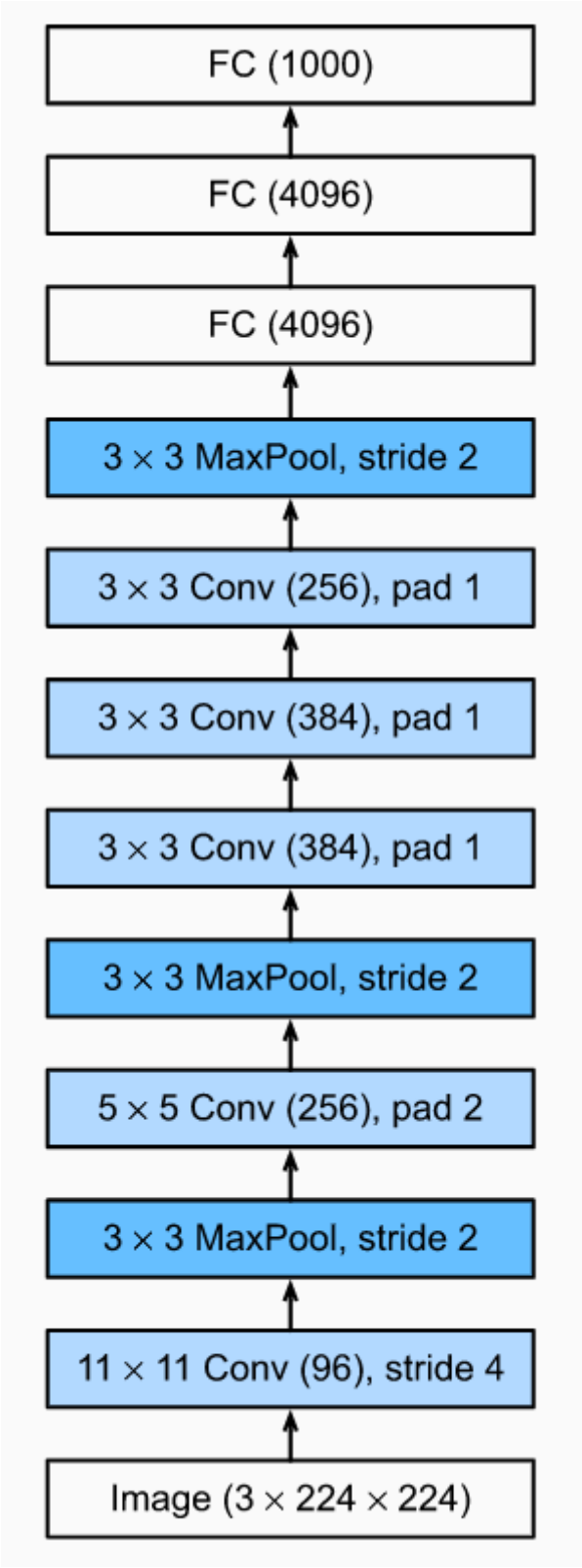


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

emm,这图看起来很复杂的样子，那是因为论文使用了两个**GPU**来加速训练，所以整个架构被分成了两部分，下面我只讨论不使用GPU的架构。

整理后如下图：



输入

之所以输入224*224的图片，是因为做了**数据增强**的操作，论文将256*256的图片提取4个**corner patches**和1个**center patches**（个人理解是图片的四角和中心的图片）以及它们的**水平反射**(horizontal reflections)，一共10张图片。论文还改变了图片RGB通道的强度，在整个ImageNet上做了**PCA**，目的就是为了**预防过拟合。**

隐藏层

第一个卷积层它的卷积核大小为11*11，共96个，**步长**为4，**无填充**，激活函数使用**Relu**，之后并不是直接进行最大池化，而是进行**LRN**（局部响应归一化），然后在进行**重叠的最大池化。**

第二个卷积层除了卷积核大小不一样以及**填充方式为same**以外，和第一个卷积层执行的是一样的操作。

第三个到第五个卷积层是**直接相连**的，中间没有任何其他操作。

第一个和第二个全连接层都有4096个神经单元，并且都使用了**Dropout**方法（**预防过拟合**）。

第三个全连接层即输出层有1000个神经单元，因为有1000个类别，激活函数使用**softmax。**

代码实现

```
import tensorflow as tf
import numpy as np

model = tf.keras.Sequential([tf.keras.layers.Conv2D(96,(11,11),
(4,4),padding='valid',activation='relu',input_shape=(224,224,1)),
                             tf.keras.layers.MaxPool2D((3,3),(2,2)),
                             tf.keras.layers.Conv2D(256,(5,5),padding='same',activation='relu'),
                             tf.keras.layers.MaxPool2D((3,3),(2,2)),
                             tf.keras.layers.Conv2D(384,(3,3),padding='same',activation='relu'),
                             tf.keras.layers.Conv2D(384,(3,3),padding='same',activation='relu'),
                             tf.keras.layers.Conv2D(256,(3,3),padding='same',activation='relu'),
                             tf.keras.layers.MaxPool2D((3,3),(2,2)),
                             tf.keras.layers.Flatten(),
                             tf.keras.layers.Dense(4096,activation='relu'),
                             tf.keras.layers.Dropout(0.5),
                             tf.keras.layers.Dense(4096,activation='relu'),
                             tf.keras.layers.Dropout(0.5),
                             tf.keras.layers.Dense(1000,activation='softmax')])

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

#我没有下载数据集，因为224*224的数据集实在太大，我的笔记本顶不住。
model.fit(x_train,y_train,epochs=10)
```

模型参数

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 54, 54, 96)        11712
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 26, 26, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 26, 26, 256)       614656
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 384)       885120
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 384)       1327488
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 256)       884992
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 6400)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              26218496
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              4097000
=================================================================
Total params: 50,820,776
```

备注

代码实现中我只实现了网络的搭建，并**没有涉及到训练**，因为涉及到的数据集过于庞大，电脑吃不消。