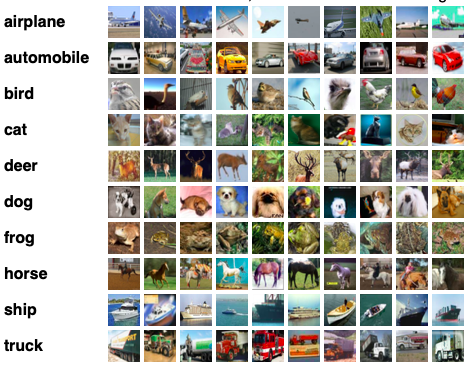
**算例：VGG16+Cifar10**

1. 数据集

经典的分类数据及cifar10，包含类别飞机、汽车、鸟、猫、鹿、狗、青蛙、马、船、卡车，共十类，其中训练集50,000张，测试集10,000张，十类等分。图片大小为32×32。部分数据集展示如下：



1. 网络框架

选取vgg16网络模型，为加速训练，添加正则层，并根据数据集十分类问题修改模型最后的全连接层。网络结构如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Layer | Kenel | Input size | Padding | Output size | Paras | 备注 |
| conv2d | 64@3×3 | 32×32×3 | same | 32×32×64 | 1792 | - |
| activation | relu | 32×32×64 | - | 32×32×64 | 0 | - |
| batch normalization |  | 32×32×64 | - | 32×32×64 | 256 | 128个非训练参数 |
| conv2d | 64@3×3 | 32×32×64 | same | 32×32×64 | 36928 | - |
| activation | relu | 32×32×64 | - | 32×32×64 | 0 | - |
| batch normalization |  | 32×32×64 | - | 32×32×64 | 256 | 128个非训练参数 |
| max pooling | 2×2 | 32×32×64 | - | 16×16×64 | 0 | - |
| conv2d | 128@3×3 | 16×16×64 | same | 16×16×128 | 73856 |  |
| activation | relu | 16×16×128 | - | 16×16×128 | 0 |  |
| batch normalization |  | 16×16×128 | - | 16×16×128 | 512 | 256个非训练参数 |
| conv2d | 128@3×3 | 16×16×128 | same | 16×16×128 | 147,584 |  |
| activation | relu | 16×16×128 | - | 16×16×128 | 0 |  |
| batch normalization |  | 16×16×128 | - | 16×16×128 | 512 | 256个非训练参数 |
| max pooling | 2×2 | 16×16×128 | - | 8×8×128 | 0 | - |
| conv2d | 256@3×3 | 8×8×128 | same | 8×8×256 | 295,168 |  |
| activation | relu | 8×8×256 | - | 8×8×256 | 0 |  |
| batch normalization |  | 8×8×256 | - | 8×8×256 | 1024 | 512个非训练参数 |
| conv2d | 256@3×3 | 8×8×256 | same | 8×8×256 | 590,080 |  |
| activation | relu | 8×8×256 | - | 8×8×256 | 0 |  |
| batch normalization |  | 8×8×256 | - | 8×8×256 | 1024 | 512个非训练参数 |
| conv2d | 256@3×3 | 8×8×256 | same | 8×8×256 | 590,080 |  |
| activation | relu | 8×8×256 | - | 8×8×256 | 0 |  |
| batch normalization |  | 8×8×256 | - | 8×8×256 | 1024 | 512个非训练参数 |
| max pooling | 2×2 | 8×8×256 | - | 4×4×256 | 0 | - |
| conv2d | 512@3×3 | 4×4×256 | same | 4×4×512 | 1,180,160 |  |
| activation | relu | 4×4×512 | - | 4×4×512 | 0 |  |
| batch normalization |  | 4×4×512 | - | 4×4×512 | 2048 | 1024个非训练参数 |
| conv2d | 512@3×3 | 4×4×512 | same | 4×4×512 | 2,359,808 |  |
| activation | relu | 4×4×512 | - | 4×4×512 | 0 |  |
| batch normalization |  | 4×4×512 | - | 4×4×512 | 2048 | 1024个非训练参数 |
| conv2d | 512@3×3 | 4×4×512 | same | 4×4×512 | 2,359,808 |  |
| activation | relu | 4×4×512 | - | 4×4×512 | 0 |  |
| batch normalization |  | 4×4×512 | - | 4×4×512 | 2048 | 1024个非训练参数 |
| max pooling | 2×2 | 4×4×512 | - | 2×2×512 | 0 | - |
| conv2d | 512@3×3 | 2×2×512 | same | 2×2×512 | 2,359,808 |  |
| activation | relu | 2×2×512 | - | 2×2×512 | 0 |  |
| batch normalization |  | 2×2×512 | - | 2×2×512 | 2048 | 1024个非训练参数 |
| conv2d | 512@3×3 | 2×2×512 | same | 2×2×512 | 2,359,808 |  |
| activation | relu | 2×2×512 | - | 2×2×512 | 0 |  |
| batch normalization |  | 2×2×512 | - | 2×2×512 | 2048 | 1024个非训练参数 |
| conv2d | 512@3×3 | 2×2×512 | same | 2×2×512 | 2,359,808 |  |
| activation | relu | 2×2×512 | - | 2×2×512 | 0 |  |
| batch normalization |  | 2×2×512 | - | 2×2×512 | 2048 | 1024个非训练参数 |
| max pooling | 2×2 | 2×2×512 | - | 1×1×512 | 0 | - |
| Flatten | - | 1×1×512 | - | 512 | - | - |
| dense1 | - | 512 | - | 512 | 262,656 | relu激活 |
| batch normalization |  | 512 |  | 512 | 2048 | 1024个非训练参数 |
| dense2 | - | 512 |  | 10 | 5,130 | softmax激活 |

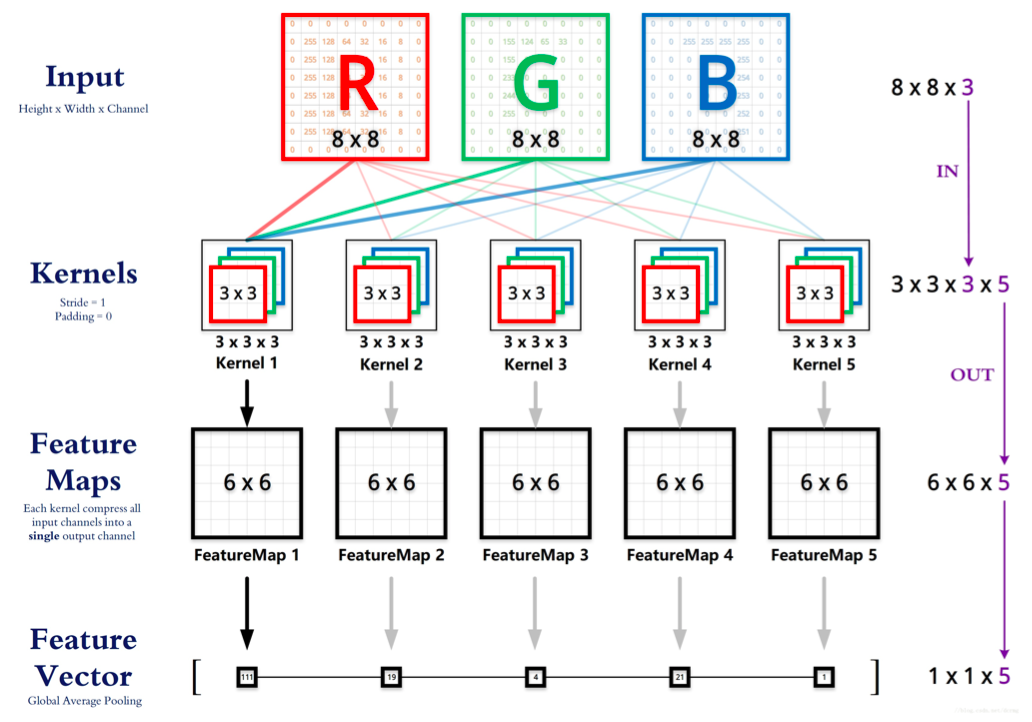
共计15,001,418个参数，其中有14,991,946个训练参数，9,472个非训练参数。

*注：vcnn论文证明的vgg16模型，拿掉全连接层，不包含bn层，估计参数为14,714,688，比本算例模型少20万参数。*

1. 细节描述
2. 卷积

卷积操作与mnist算例相比有两处不同。

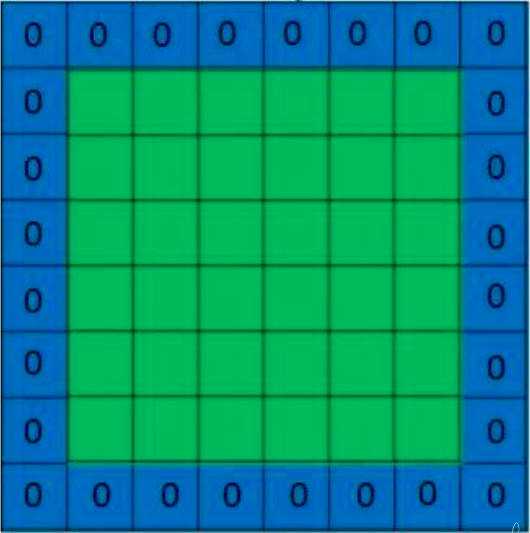
1. cifar10是RGB三通道的彩色图像，卷积涉及到通道问题，如下图所示



对于多通道图，卷积核分别对每个通道进行卷积，并将最终结果逐像素相加得到卷积结果。在本算例中，对于一张输入图像，长宽各为227，通道数为3，经过一层64个3×3的卷积核，每个卷积核要卷积3个通道，得到3张结果进行逐像素相加，最终得到64张特征图。

1. 补零

为了增加网络深度，同时保护特征图不至于因为下采样而消失，模型使用“same”作为卷积补零（padding）策略，即通过在输入图周边补零，使得卷积输出结果与输入大小相同。



若要保证输出结果与输入大小相同，padding应满足：

其中f为filter的尺寸，例如本算例中使用的均为3×3卷积核，f=3，则p＝1。

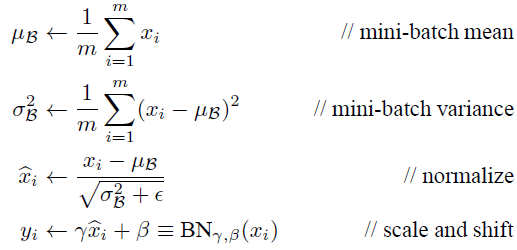
1. 激活层

激活使用的relu和softmax与mnist算例一致。

1. 正则化层

正则化 (batch\_normalization) 是在vgg网络提出后出现的技术，可以极大地加快模型收敛的速度。

bn层算法：



对每个节点，计算数据的平均值和方差，对数据去中心化后，再进行scale和shift，保持非线性性。其中红框中的和为不可训练参数，橙色框中和为训练参数。

1. 池化层

池化层选用最大池化，核为2，与mnist算例相同。

1. 全连接层

共两层，第一层dense1有512个节点，relu激活，第二层dense2有10个节点，softmax激活进行分类。计算与mnist算例相同。

备注：本算例获得的测试集准确率在86.32%，训练集准确率为99.28%，出现了过拟合的情况。经过测试，加入dropout可以有效缓解过拟合，测试机准确率可以升至97%。本文档不含dropout。