VGG-16 on Cifar-100 dataset

Aren Beglaryan
Applied Statistics for Data Science, Yerevan State University

ABSTRACT

In this work we will use VGG-16 to classify Cifar-100 dataset. Our metric will be accuracy because we have balanced data and will try to increase accuracy using VGG, Resnet connections, data augmentation. We will have very deep convolutional neural network (16 layers) and 3 resnet connections.

1. INTRODUCTION

Convolutional networks (ConvNets) have recently enjoyed a great success in large-scale image recognition. With ConvNets becoming more of a commodity in the computer vision field, a number of attempts have been made to improve the original architecture of VGG. As a result, we come up with significantly more accurate VGG architectures.

2. ABOUT DATASET

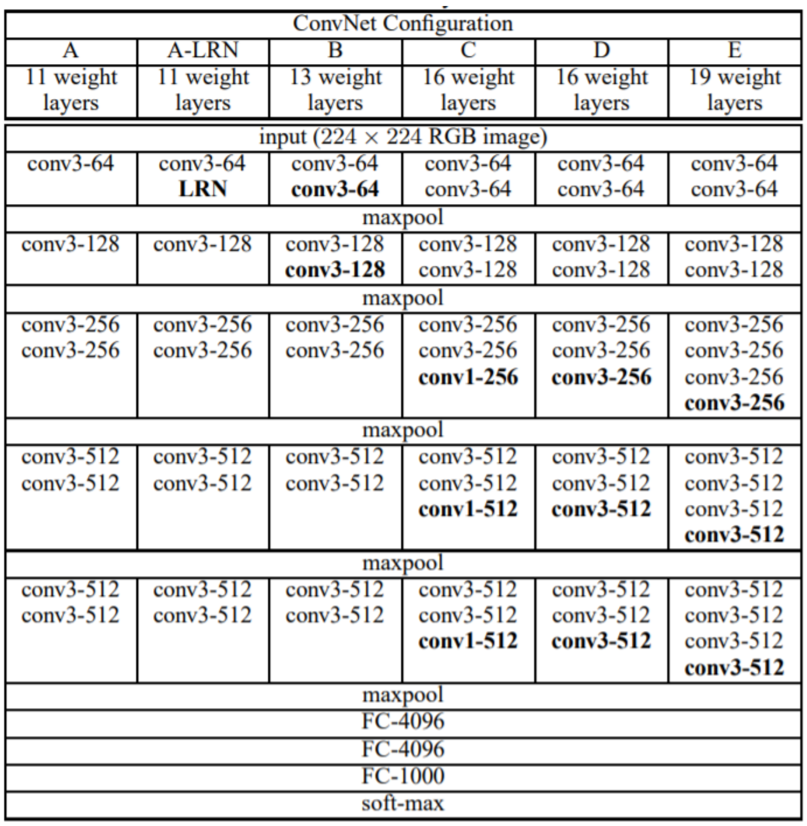
Cifar-100 is dataset which has 100 classes containing 600 images each. There are 500 training and 100 testing images per class.

(https://www.cs.toronto.edu/~kriz/cifar.html)

3. ABOUT VGG ARCHITECHTURE

During training, the input to our ConvNets is a fixed-size 32×32 RGB image. The only preprocessing we do is subtracting the mean RGB value, computed on the training set,

from each pixel. The image is passed through a stack of convolutional (conv.) layers, where we use filters with a very small receptive field: 3 × 3 (which is the smallest size to capture the notion of left/right, up/down, center. The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3 × 3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2 × 2 pixel window, with stride 2. A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 100 - way Cifar-100 classification and thus contains 100 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. All hidden layers are equipped with the rectification (ReLU) non-linearity. We note that our network contain Batch Normalisation (BN) after each Conv layer.
We will use D section in the table below. Only dofference is the last FC layer 100 in spite of 1000 and starting from 32 x 32 image in spite of 224 x 224. You can see VGG original paper here. (https://arxiv.org/pdf/1409.1556.pdf)



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

4. TRAINING

We use PyTorch library to build our network. First we implement a small network (3 Conv layers + 2 FC layers) and saw that was overfitted and the the best accuracy on the test dataset is 32%. Then started to add more layers and and increase neurons of each layer and saw it was overfitting. After that I added some residual connections (3 connections) , did data augmentation (Random crop and Horizontal flip). And after that did hyperparamether tuning and finally get 50 % accuracy on this data. (Note that this is top-1 accuracy but you can find 72% accuracy, which is the best on top-3).

Namely, the training is carried out by optimising the multinomial logistic regression objective using Adam optimizer, but in VGG paper they used SGD (I decided to do this because there was no Adam when they built VGG model). The batch size was set to 256. The training was regularised by weight decay (the L2 penalty multiplier set to $5 \cdot 10e{-4}$) and dropout regularisation for the 2, 4, 7, 10, 13 layers and first two fully-connected layers (dropout ratio set to 0.5). Also we used residual connections from 2 to 5 layers, 6 to 9 layers and 10 to 12 layers. The learning rate was initially set to $10e{-3}$, and then decreased by a factor of 10 at the milestones 40, 70, 90, 110. This means it will be decreased by a factor of 10 four times. And we run this using Google Colab.

REFERENCES

VGG paper (https://arxiv.org/pdf/1409.1556.pdf)

Resnet paper (https://arxiv.org/pdf/1512.03385.pdf)

Cifar-100 (https://www.cs.toronto.edu/~kriz/cifar.html)

VGG Neural Networks: The next step after AlexNet(https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c)

An overview of Resnet and it's variants (https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035)

PyTorch documentation (https://pytorch.org/docs/stable/index.html)

Short introduction to data augmentation (https://markku.ai/post/data-augmentation/)

Batch normalization in Neural Networks (https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c)

Batch normalization paper (https://arxiv.org/pdf/1502.03167.pdf)