Image Classification Using ConvNet

Hardik Khichi 2016CS50404
Yash Malviya 2016CS50403

COL341: Machine Learning - Assignment 3 : Submission : 17th October

1 Introduction

Major winning Convolutional Neural Networks (CNNs), such as AlexNet, VGGNet, ResNet, GoogleNet, include tens to hundreds of millions of parameters, which impose considerable computation and memory overhead. This limits their practical use for training, optimization and memory efficiency. On the contrary, light-weight architectures, being proposed to address this issue, mainly suffer from low accuracy.

We have implemented a elementary model architecture, called **SimpleNet**, which follows a set of designing principles, with which they empirically show, a well-crafted yet simple and reasonably deep architecture can perform on par with deeper and more complex architectures. SimpleNet provides a good tradeoff between the computation/memory efficiency and the accuracy. This simple 13-layer architecture outperforms most of the deeper and complex architectures to date such as VGGNet, ResNet, and GoogleNet on several well-known benchmarks while having 2 to 25 times fewer number of parameters and operations.

2 The rise of Deep Learning

Since the resurgence of neural networks, deep learning methods have been gaining huge success in diverse fields of applications, including semantic segmentation, classification, object detection, image annotation and natural language processing. What has made this enormous success possible is the ability of deep architectures to do feature learning automatically, eliminating the need for a feature engineering stage.

CNNs, have been one of the most popular deep learning methods and also a major winner in many computer vision and natural language processing related tasks lately. Since CNNs take into account the locality of the input, they can find different levels of correlation through a hierarchy of consecutive application of convolution filters. This way they are able to find and exploit different levels of abstractions in the input data and using this perform very well on both coarse and fine level details. Therefore the depth of a CNN plays an important role in the discriminability power the network offers. The deeper the better.

3 SimpleNet, What and Why?

Designing more effective networks were desirable and attempted from the advent of neural networks. The computational and memory usage overhead caused by such practices, limits the expansion and applications of deep learning methods. The architecture used for SimpleNet exhibits the best characteristics of works done to achieve a good accuracy on a simplified architecture.

It proposes a 13 layer convolutional network that achieves state of the art result on CIFAR100. The network has fewer parameters (2 to 25 times less) compared to all previous deep architectures, and performs either superior to them or on par despite the huge difference in number of parameters and depth. For architectures such as SqueezeNet/FitNet where the number of parameters is less than SimpleNet but also are deeper, it's network accuracy is far superior to what can be achieved with such networks. It's architecture is also the smallest (depth wise) architecture that both has a small number of parameters compared to all leading deep architectures, and also unlike previous architectures such as SqueezeNet or FitNet, gives higher or very competitive performance against all deep architectures. Our model then can be compressed using deep compression techniques and be further enhanced, resulting in a very good candidate for many scenarios.

4 Design of Architecture

We propose a simple convolutional network with 13 layers. The network employs a homogeneous design utilizing 3 X 3 kernels for convolutional layer and 2 X 2 kernels for pooling operations.

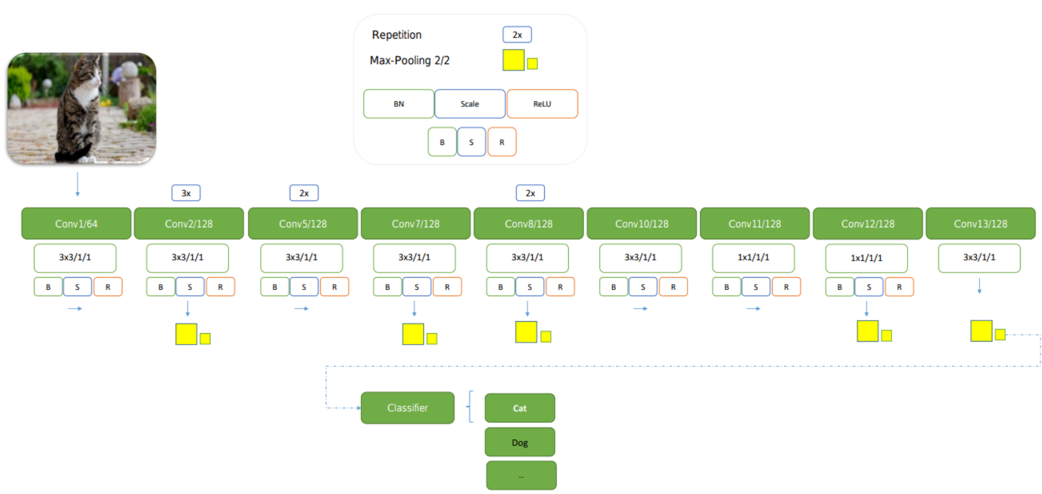

Figure 1: Architecture Of Neural Network

The only layers which do not use 3 X 3 kernels are 11th and 12th layers, these layers, utilize 1 X 1 convolutional kernels. Feature-map down-sampling is carried out using nonoverlaping 2 X 2 maxpooling. In order to cope with the problem of vanishing gradient and also over-fitting, we used batch-normalization with moving average fraction of 0.95 before any ReLU non-linearity.

5 Reference

simplenet v1: https://arxiv.org/pdf/1608.06037v7.pdf