Image Classification CIFAR100

Hardik Khichi 2016CS50404

Yash Malviya 2016CS50403

COL341: Machine Learning - Assignment 3

1 Submission : 17th October

1.1 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.

1.2 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.

1.3 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.

1.4 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.

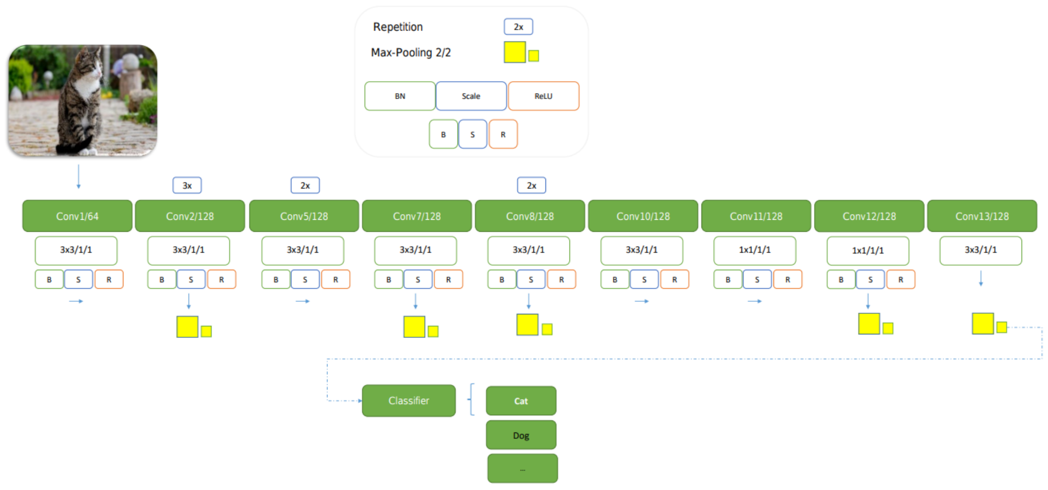

Figure 1: Architecture Of Neural Network

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.

1.5 Reference

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

2 Submission : 24th October

2.1 Res2Net

The module proposed in the paper involves residual-like connections within a single residual block, hence it was named Res2Net. This models tries to improve on traditional CNNs. In pattern recognition tasks, patterns can occur at multiple scales.

- Objects of different size
- Contextual info may be present in a much larger area than the actual image

Convolutional neural networks (CNNs) naturally learn coarse-to-fine multi-scale features through a stack of convolutional operators. This paper describe a building block to replace the commonly used bottleneck Block in backbone network of other neural networks such as ResNet, ResNext, DLA.

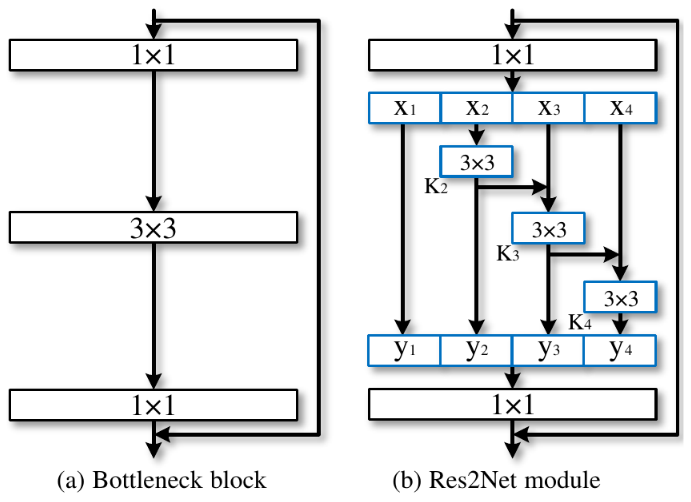

Figure 2: Architecture Of Neural Network

Just like the paper we replace Bottleneck Block in ResneXt-29 with Res2net module.

2.2 Accuracy for SimpleNet on test data

Completed training and testing of SimpleNet After training for 10 epochs: 0.4834
After training for 50 epochs: 0.6356

We weren't able to complete our implementation of Res2Net, so submitting the SimpleNets tested code.

2.3 Reference

Res2Net: https://arxiv.org/pdf/1904.01169v2.pdf

3 Submission : 31st October

3.1 Introduction to ResNext

Designing architectures becomes increasingly difficult with the growing number of hyper-parameters (width, filter sizes, strides, etc.), especially when there are many layers. VGG-nets exhibit a simple yet effective strategy of constructing very deep networks: stacking building blocks of the same shape. This strategy is inherited by ResNets which stack modules of the same topology. This simple rule reduces the free choices of hyperparameters, and depth is exposed as an essential dimension in neural networks.

3.2 ResNext Module Structure

These blocks are subject to two simple rules inspired by VGG/ResNets:

- If producing spatial maps of the same size, the blocks share the same hyper-parameters (width and filter sizes).
- Each time when the spatial map is downsampled by a factor of 2, the width of the blocks is multiplied by a factor of 2. The second rule ensures that the computational complexity, in terms of FLOPs (floating-point operations)

3.3 Cardinality

This strategy exposes a new dimension, which we call "**cardinality**" (the size of the set of transformations), as an essential factor in addition to the dimensions of depth and width. This model empirically show that even under the restricted condition of maintaining complexity, increasing cardinality is able to improve classification accuracy. Moreover, increasing cardinality is more effective than going deeper or wider when we increase the capacity.

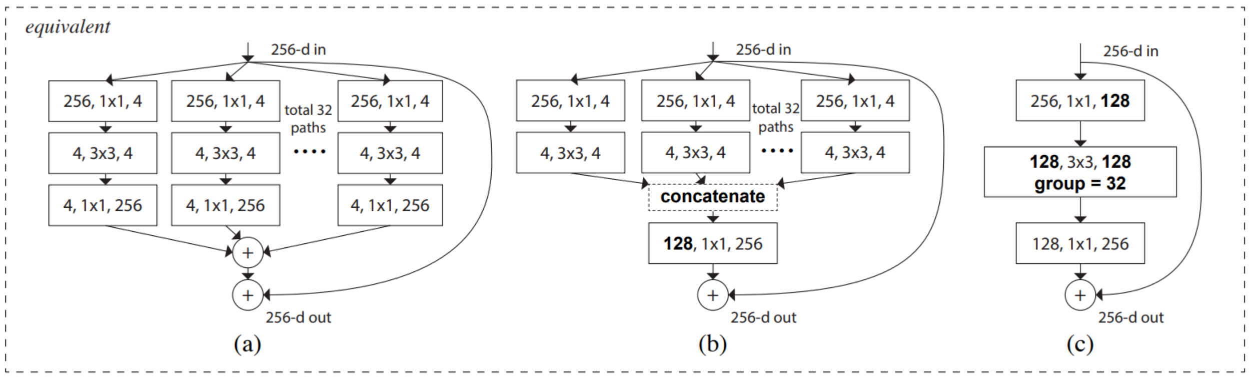

Figure 3. Equivalent building blocks of ResNeXt. **(a)**: Aggregated residual transformations, the same as Fig. 1 right. **(b)**: A block equivalent to (a), implemented as early concatenation. **(c)**: A block equivalent to (a,b), implemented as grouped convolutions [24]. Notations in **bold** text highlight the reformulation changes. A layer is denoted as (# input channels, filter size, # output channels).

Figure 3: Design Of ResNext Module

3.4 Architecture

First Layer
Conv2D: 64 filters, 3x3 kernel, stride 1x1, padding none
Batch Normalization
Relu

Residual Stages:
Each Stage has 3 residual blocks(explained in Section 3.3)
Followed by Conv2D layer halving number of channels

Final Layer
Dense Layer i.e. single fully connected layer with 100 neurons
Relu

Conv2D and Dense initializers: he normal
Optimizer: Adam
Data Augmentation Used: None

3.5 Accuracy for ResNext on test data

Completed training and testing of Resnext After training for 10 epochs: 56.25
This is the best Validation Accuracy so far. Training is currently running therefore we have not written premature results.

3.6 Reference

ResNeXt: https://arxiv.org/pdf/1611.05431.pdf

4 14th November

4.1 Introduction to Deep Resnet

Deep residual networks were shown to be able to scale up to thousands of layers and still have improving performance. However, each fraction of a percent of improved accuracy costs nearly doubling the number of layers, and so training very deep residual networks has a problem of diminishing feature reuse, which makes these networks very slow to train. To tackle these problems, we propose a architecture of ResNet blocks, where we decrease depth and increase width of residual networks.

4.2 Width vs depth in residual networks

The problem of shallow vs deep networks has been in discussion for a long time in machine learning. Previous literature have shown that shallow circuits can require exponentially more components than deeper circuits. The authors of residual networks tried to make them as thin as possible in favor of increasing their depth and having less parameters. They introduced a **bottleneck** block which makes ResNet blocks even thinner.

Widening of ResNet blocks (if done properly) provides a much more effective way of improving performance of residual networks compared to increasing their depth. In particular, it's observed wider deep residual networks significantly improve over a Resnet, having 50 times less layers and also being more than 2 times faster.

4.3 Use of dropout in ResNet blocks

Dropout was first introduced in 2014 as a simple way to prevent neural networks from overfitting and then was adopted by many successful architectures. It was mostly applied on top layers that had a large number of parameters to prevent feature co-adaptation and overfitting. It was then substituted by batch normalization which was introduced as a technique to reduce internal co-variate shift in neural network activation by normalizing them to have specific distribution.

4.4 Architecture

4.5 Reference

Wide Residual Networks by Sergey Zagoruyko, Nikos Komodakis:
https://arxiv.org/abs/1605.07146v4

5 Results

5.1 Summary

We have worked on implementation of 4 models for this assignment namely SimpleNet, Res2net, ResNext and DeepResnet

5.2 Performance of Different Models

The best performance obtained by different models within the allowed training time of 6 hours was calculated. For every model, number of epochs that we would be able to execute within the permitted time were estimated. These results are obtained after training each model for the above condition. Performance depicted here is accuracy percentage.

- SimpleNet: 63.56 %
- ResNext:
  Our implementation of ResNeXt has 1 initial layer (init) and 2 or 3 (0, 1, 2) ResNeXt Blocks after each block dimension of data is halved using a Conv2D Layer Finally a Dense Layer with number of classes neurons(100)

Model	**Configuration**	**Accuracy**
ResNext1	init: 32 ; 1: 32, 16 ; 2: 16, 8	28.57 %
ResNext2	init: 64 ; 1: 64, 32 ; 2: 32, 16 ; 3: 16, 8	30.5 %
ResNext3	init: 16 ; 1: 16, 8 ; 2: 8, 4 ; 3: 4, 4	20.21 %

- WideResnet:
  N is number of residual conv blocks in Resnet and k is width of of

Model	**Configuration**	**Accuracy**
WideResnet1	N=2 k=2	44.26 %
WideResnet2	N=2 k=4	45.09 %
WideResnet3	N=2 k=8	43.54 %
WideResnet4	N=2 k=16	43.72 %
WideResnet5	N=4 k=2	42.14 %
WideResnet6	N=4 k=4	46.21 %

5.3 Explanation

- For different configurations of WideResnet, we can explain the relation between their performance and their architecture. We observe that when we fix the depth of our WideResnet models, N=2. on increasing the depth of models K = 2 to 16 the accuracy of models also increases. This correlation is observed till a particular width after which accuracy starts to decrease. Note: We had implemented an early-stopper due which results are slightly altered. If we had fixed the number of epochs for all models results would have been more closer to the hypothesis.

- On increasing depth for N = 2 to 4 we observe that accuracy of our model has increased as a deeper model will obviously train better than a shallow model with same width of neurons.