


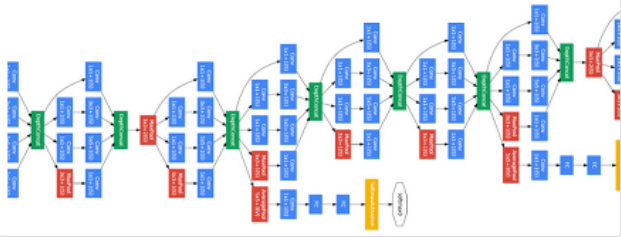
DNN_02_8 GoogleNet


🕒 생성일	@2022년 6월 23일 오전 12:37
🏷️ 유형	머신러닝/딥러닝
👤 작성자	 동훈 오

GoogLeNet (Going deeper with convolutions) 논문 리뷰

이번에는 ILSVRC 2014에서 VGGNet을 제치고 1등을 차지한 GoogLeNet을 다뤄보려 한다. 연구팀 대부분이 Google 직원이어서 아마 이름을 GoogLeNet으로 하지 않았나 싶다. 그럼 대체 왜 Google 팀에서는 이 구조의 이름을 인셉션이라 지었는지, VGGNet과는 어떤 점에서 다른지 알아보기 위해 출발해보겠다! 먼저 초록에서는 GoogLeNet의 특징에 대해

🔗 <https://phil-baek.tistory.com/entry/3-GoogLeNet-Going-deeper-with-convolutions-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0>



 <https://arxiv.org/pdf/1409.4842.pdf>

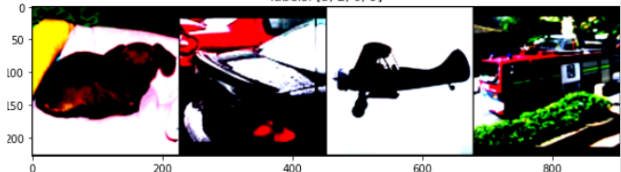
[논문 구현] PyTorch로 GoogLeNet(2014) 구현하고 학습하기

이번 포스팅에서는 GoogLeNet(Inception-v1)을 파이토치로 구현하고 학습까지 해보겠습니다. 논문 리뷰는 아래 포스팅에서 확인하실 수 있습니다. 전체 코드는 여기 에서 확인하실 수 있습니다! 스타도 부탁드립니다! 데이터셋은 torchvision 패키지에서 제공하는 STL10 dataset을 이용합니다. STL10 dataset은 10개의 label로 이루어져 있으며,

🔗 <https://deep-learning-study.tistory.com/523>

image indices: [2732 2607 1653 3264]
orch.Size([3, 228, 906])

labels: [5, 2, 0, 9]



Abstract

We propose a deep convolutional neural network architecture codenamed Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular incarnation used in our submission for ILSVRC14 is called GoogLeNet, a 22 layers deep network, the quality of which is assessed in the context of classification and detection.

1. Introduction

In the last three years, mainly due to the advances of deep learning, more concretely convolutional networks [10], the quality of image recognition and object detection has been progressing at a dramatic pace. One encouraging news is that most of this progress is not just the result of more powerful hardware, larger datasets and bigger models, but mainly a consequence of new ideas, algorithms and improved network architectures. No new data sources were used, for example, by the top entries in the ILSVRC 2014 competition besides the classification dataset of the same competition for detection purposes. Our GoogLeNet submission to ILSVRC 2014 actually uses $12\times$ fewer parameters than the winning architecture of Krizhevsky et al [9] from two years ago, while being significantly more accurate. The biggest gains in object-detection have not come from the utilization of deep networks alone or bigger models, but from the synergy of deep architectures and classical computer vision, like the R-CNN algorithm by Girshick et al [6].

GoogLeNet 은 AlexNet 보다 12 배가 더 적은 파라미터를 사용하고도 더 좋은 성능을 기록했다.

In this paper, we will focus on an efficient deep neural network architecture for computer vision, codenamed Inception, which derives its name from the Network in network paper by Lin et al [12] in conjunction with the famous “we need to go deeper” internet meme [1]. In our case, the word “deep” is used in two different meanings: first of all, in the sense that we introduce a new level of organization in the form of the “Inception module” and also in the more direct sense of increased network depth. In general, one can view the Inception model as a logical culmination of [12] while taking inspiration and guidance from the theoretical work by Arora et al [2]. The benefits of the architecture are experimentally verified on the ILSVRC 2014 classification and detection challenges, on which it significantly outperforms the current state of the art.

페이퍼에서 사용한 인셉션의 구조는 ‘Network in network’ 논문에서 비롯되었다고 한다. GoogLeNet 을 만든 저자들에게 deep 이란 의미는 두 가지 의미를 가진다고 한다. 첫 번째는 Inception module 이라는 형태의 새로운 차원을 도입했다는 것이고, 두번째 의미는 직관적으로 네트워크의 깊이를 증가시켰다는 것이다.

2. Related Work

Starting with LeNet-5 [10], convolutional neural networks (CNN) have typically had a standard structure – stacked convolutional layers (optionally followed by contrast normalization and max-pooling) are followed by one or more fully-connected layers. Variants of this basic design are prevalent in the image classification literature and have yielded the best results to-date on MNIST, CIFAR and most notably on the ImageNet classification challenge [9, 21]. For larger datasets such as Imagenet, the recent trend has been to increase the number of layers [12] and layer size [21, 14], while using dropout [7] to address the problem of overfitting.

LeNet 이후 CNN 에서는 Conv layer 를 쌓고 fully-connected가 뒤따라오는 구조가 전형적으로 되었다. GoogLeNet 에서도 같은 전형적인 구조를 사용했다. 그리고 large datasets 를 감당하기 위해 레이어의 수를 증가 시켰고 dropout 을 적용해서 overfitting 문제를 해결하려고 했다.

Despite concerns that max-pooling layers result in loss of accurate spatial information, the same convolutional network architecture as [9] has also been successfully employed for localization [9, 14], object detection [6, 14, 18, 5] and human pose estimation [19]. Inspired by a neuroscience model of the primate visual cortex, Serre et al. [15] use a series of fixed Gabor filters of different sizes in order to handle multiple scales, similarly to the Inception model. However, contrary to the fixed 2-layer deep model of [15], all filters in the Inception model are learned. Furthermore, Inception layers are repeated many times, leading to a 22-layer deep model in the case of the GoogLeNet model.

GoogLeNet 에서 inception module 은 여러 번 반복되고 22개의 layer 를 이룬다.

Network-in-Network is an approach proposed by Lin et al. [12] in order to increase the representational power of neural networks. When applied to convolutional layers, the method could be viewed as additional 1×1 convolutional layers followed typically by the rectified linear activation [9]. This enables it to be easily integrated in the current CNN pipelines. We use this approach heavily in our architecture. However, in our setting, 1×1 convolutions have dual purpose: most critically, they are used mainly as dimension reduction modules to remove computational bottlenecks, that would otherwise limit the size of our networks. This allows for not just increasing the depth, but also the width of our networks without significant performance penalty.

Network in Network 는 신경망의 표현력을 높이기 위해 제안된 접근법이다. 이 방법은 1×1 conv layer 가 추가되며, ReLU 활성화함수가 뒤 따른다. GoogleNet 에서는 1×1 conv layer 를 사용한 두 가지 목적이 있다.

1. 병목현상을 제거하기 위한 차원 축소
2. 네트워크 크기 제한

3. Motivation and High level considerations

The most straightforward way of improving the performance of deep neural networks is by increasing their size. This includes both increasing the depth – the number of levels – of the network and its width: the number of units at each level. This is as an easy and safe way of training higher quality models, especially given the availability of a large amount of labeled training data. However this simple solution comes with two major drawbacks.

네트워크의 depth, width 를 증가시키는 방법은 신경망의 성능을 높이는 확실한 방법이다. 하지만 여기에는 2가지 치명적인 약점이 있다.

Bigger size typically means a larger number of parameters, which makes the enlarged network more prone to overfitting, especially if the number of labeled examples in the training set is limited. This can become a major bottleneck, since the creation of high quality training sets can be tricky

and expensive, especially if expert human raters are necessary to distinguish between fine-grained visual categories like those in ImageNet (even in the 1000-class ILSVRC subset) as demonstrated by Figure 1.

첫 번째로, 크기가 커진다는 것은 파라미터의 수가 늘어난다는 것인데, 이는 특히 학습 데이터의 수가 제한된 경우 오버피팅에 더 노출되기 쉽다. 주요한 병목현상을 일으킬 수도 있는데 ImageNet 처럼 세밀한 카테고리를 구별해야 하는 경우, 고품질의 트레이닝 셋을 생성하는 것은 매우 까다롭고 비용이 너무 높기 때문이다.

Another drawback of uniformly increased network size is the dramatically increased use of computational resources. For example, in a deep vision network, if two convolutional layers are chained, any uniform increase in the number of their filters results in a quadratic increase of computation. If the added capacity is used inefficiently (for example, if most weights end up to be close to zero), then a lot of computation is wasted. Since in practice the computational budget is always finite, an efficient distribution of computing resources is preferred to an indiscriminate increase of size, even when the main objective is to increase the quality of results.

두 번째 약점으로, 네트워크의 크기가 커질 수록 컴퓨팅 자원의 사용도 증가하게 된다. 예를 들어, 필터 개수가 늘어난다면, 연산량은 quadric 하게 증가한다. 실전에서 사용할 수 있는 컴퓨팅 자원은 제한되어 있기에, 네트워크의 크기를 증가시키는 것보다 컴퓨팅 자원을 효율적으로 분배하는 것이 더 중요하게 고려된다.

The fundamental way of solving both issues would be by ultimately moving from fully connected to sparsely connected architectures, even inside the convolutions. Besides mimicking biological

두 가지 약점을 해결하는 방법은 dense 한 fully-connected 구조에서 sparsely connected 구조로 바꾸는 것이다.

만약 dataet 의 분배 확률을 sparse 하면서도 더 큰 심층 신경망으로 표현 가능하다면, 입력 layer 에서 출력 layer 로 향하는 layer 간의 관계를 통계적으로 분석한 후, 연관 관계가 높은 것들만 연결하여 최적의 sparse 한 네트워크를 만들 수 있다.

On the downside, today's computing infrastructures are very inefficient when it comes to numerical calculation on non-uniform sparse data structures. Even if the number of arithmetic operations is reduced by $100\times$, the overhead of lookups and cache misses is so dominant that switching to sparse matrices would not pay off. The gap is widened even further by the use of steadily improving, highly tuned, numerical libraries that allow for extremely fast dense matrix multiplication, exploiting the minute details of the underlying CPU or GPU hardware [16, 9]. Also, non-uniform sparse models require more sophisticated engineering and computing infrastructure. Most current vision oriented machine learning systems utilize sparsity in the spatial domain just by the virtue of employing convolutions. However, convolutions are implemented as collections of dense connections to the patches in the earlier layer. ConvNets have traditionally used random and sparse connection tables in the feature dimensions since [11] in order to break the symmetry and improve learning, the trend changed back to full connections with [9] in order to better optimize parallel computing. The uniformity of the structure and a large number of filters and greater batch size allow for utilizing efficient dense computation.

-내용 추가중-