25.07.29

GoogleNet  
참조 논문 : Going Deeper with Convolutions(14)

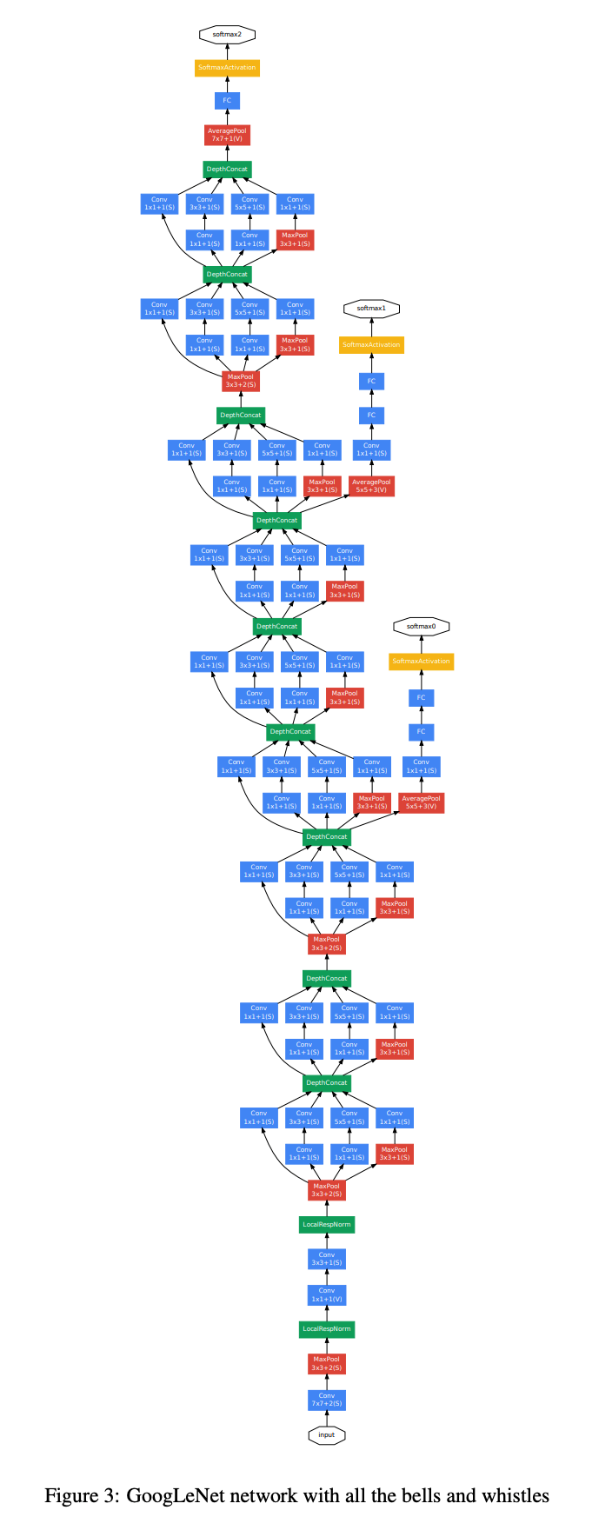
**GoogleNet**  
계산량을 유지하면서 신경망의 깊이와 폭을 늘릴 수 있도록 설계된 네트워크.   
설계 과정에서 Hebbian principle와 multiscale processing의 원리를 적용.

**Hebbian principle(헤비안 원리)란**  
헤비안 학습((Hebbian Learning)이라고도 한다. 헤비안 학습은 연결되어 있는 뉴런들은 같이 발화하며 학습이 이루어진다는 내용으로, 학습 과정에서 일어나는 시냅스 가소성을 설명하고자 했다.   
뉴런 활성과 값 간의 상관 관계를 이용해 연결 가중치를 업데이트한다.  
  
구글 넷에서는 Inception 모듈을 설계하는 데에 영향을 주었다.

**multiscale processing(멀티스케일 처리)**다양한 해상도, 크기의 특징을 함께 사용하여 모델의 성능을 향상시키는 기술이다.  
구글넷의 Inception 모듈에서 다양한 크기의 conv 레이어를 사용하는 것이 그 예이다.

**googleNet 특징**

GoogleNet은 깊이를 단순히 늘리는 대신 병렬 구조를 활용해 네크워크를 넓게 구성한다. 다양한 스케일의 특성을 추출하고 이를 통합해 표현력을 높이는 방식의 네트워크이다.



위는 GoogleNet의 네트워크 구조 이미지이다.

GoogleNet은 22개의 층으로 이루어져 있으며 세부분으로 나눌 수 있다. 단순히 Conv, pool layer를 쌓는 초반 6개의 층을 Stem이라고 한다. 이후 9층을 Inception 모듈을 쌓고 마지막 output classifier가 존재한다. 이때 output classifier에서는 Fc레이어를 사용하지 않고, 학습 파라미터 양을 줄이기 위해 Global Average Pooing을 사용한다.  
또한 Inception 모듈 중간에 보조 분류기가 존재한다. 보조 분류기는 신경망이 깊기 때문에 기울기 손실을 완화하기 위해 사용하며, 학습 과정에서만 사용한다.

Global Average Pooing(평균 풀링)에 대해서 보다 자세하게 얘기하자면, Fully-connected layer를 사용하면 마지막 피처맵을 일렬로 펼친 뒤 소프트 맥스를 사용해 분류하는 것과 다르게, 피처맵을 벡터화 하여 소프트맥스 처리를 한다. 이 경우 학습 매개변수의 개수가 줄어든다.

GoogleNet에서는 Inception 모듈을 활용함.  
텍스트, 도표, 평면도, 라인이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

(a)이미지가 초기 Inception 모듈의 이미지이며, (b)가 (a)의 문제를 해결한 구조의 이미지이다.

Inception 모듈은 Input에 병렬로 Conv(1X1, 3X3, 5X5), Pool 레이어를 통과, 여러 특징을 추출한 뒤, 추출한 특징맵을 채널 방향으로 합친다. Inception 모듈 덕분에 다양한 특징을 결합해 광범위한 특징을 추출할 수 있다.  
(a) 이미지가 초기 Inception의 구조인데, Naive Inception Module을 사용했을 때는 연산량이 매우 증가한다. 이를 해결하기 위해 1x1 conv 레이어(bottleneck layer)를 추가한 것이 (b)의 구조이다. 1x1 conv 레이어를 통해 차원 축소가 일어나 연산량이 줄어든다.

**Inception 버전 별 차이**

Inception 모듈은 버전에 따라서 분류가 된다. 앞서 소개한 것이 Inception 모듈의 v1 이었다.

**Inception-v2**기존의 GoogleNet에서 연산량을 더 줄여보기 위해 Conv 필터를 분해하는 것이 핵심 아이디어다. 5x5 컨볼루션을 3x3 컨볼루션 2개로 분해하면 동일한 receptive field를 유지하면서 파라미터 수와 계산량을 줄일 수 있다.도표, 스케치, 평면도, 라인이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

도표, 텍스트, 라인, 기술 도면이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

nxn 컨볼루션 또한 1xn과 nx1으로 나누어 주었다. 이는 연산량을 줄이는 동시에 네트워크에 비선형을 추가하는 효과가 있었다.

배치 정규화를 적용해 학습 속도를 높이고 모델의 안정성을 높였다.

**Inception-v3**Inception-v2의 개선 사항을 포함하여, 라벨 스무딩(Label Smoothing)을 적용했다. Label Smoothing은 학습시 정답 라벨을 1이 아닌 1에 가까운 값으로, 오답 라벨을 0이 아닌 0에 가까운 값으로 변환해 과적합을 방지하고 일반화 성능을 향상시키는 기법이다.  
  
또한 보조 분류기에 배치 정규화를 적용시켰다.

* GoogleNet 모델을 Pytorch를 통해 구현한 결과

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Epoch** | **Train Acc (%)** | **Train Loss** | **Test Acc (%)** | **Test Loss** |
| 1 | 53.46 | 1.27 | 61.27 | 1.08 |
| 2 | 71.39 | 0.82 | 71.67 | 0.82 |
| 3 | 79.08 | 0.60 | 76.02 | 0.69 |
| 4 | 84.09 | 0.46 | 78.39 | 0.63 |
| 5 | 87.82 | 0.35 | 82.07 | 0.54 |
| 6 | 90.69 | 0.27 | 80.91 | 0.61 |
| 7 | 93.09 | 0.21 | 81.11 | 0.59 |
| 8 | 94.68 | 0.16 | 80.05 | 0.70 |
| 9 | 95.54 | 0.13 | 79.73 | 0.71 |
| 10 | 96.65 | 0.10 | 82.63 | 0.61 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epoch | Train Accuracy | Train Loss | Test Accuracy | Test Loss |
| 100 | 99.64 | 0.01 | 88.71 | 0.6 |

에폭을 100까지 늘렸을 때, 느리지만 천천히 Train Accuracy가 상승하는 모습을 확인할 수 있었다.

* Pytorch 내의 GoogleNet 모델을 불러와 돌린 결과

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Epoch** | **Train Acc (%)** | **Train Loss** | **Test Acc (%)** | **Test Loss** |
| 1 | 54.57% | 2.04 | 64.89% | 0.99 |
| 2 | 72.60% | 1.3 | 74.77% | 0.73 |
| 3 | 80.63% | 0.96 | 77.02% | 0.66 |
| 4 | 85.26% | 0.76 | 78.05% | 0.67 |
| 5 | 88.91% | 0.6 | 77.57% | 0.69 |
| 6 | 91.71% | 0.47 | 82.15% | 0.55 |
| 7 | 93.84% | 0.38 | 82.26% | 0.57 |
| 8 | 95.66% | 0.29 | 79.47% | 0.73 |
| 9 | 96.61% | 0.25 | 84.82% | 0.52 |
| 10 | 97.51% | 0.19 | 81.59% | 0.72 |

결과 비교

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Train Acc (%)** | **Train Loss** | **Test Acc (%)** | **Test Loss** |
| 구현한 모델 | 96.65 | 0.10 | 82.63 | 0.61 |
| Pytorch 모델 | 97.51% | 0.19 | 81.59% | 0.72 |