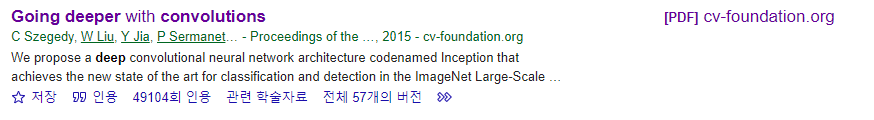
**2021270678 컴퓨터융합소프트웨어학과 나강민**

**Going deeper with convolutions**

SZEGEDY, Christian, et al. Going deeper with convolutions. In:Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. p. 1-9.

**1. Introducion**

2014년 기준으로, 최근 3년 동안 딥러닝과 CNN의 발전으로 객체 분류 및 감지 능력이 향상되었다. 이는 단순히 하드웨어의 발전, 더 많은 데이터셋과 모델로 이루어진 것이 아니라 소프트웨어 개선으로 인해 많은 개선이 이루어졌기 때문이다.

복잡한 이미지 인식 문제를 해결하려면 대규모 컴퓨팅 리소스가 필요하다. 따라서 모바일 기기나 임베디드 시스템과 같은 저전력 디바이스에서는 활용이 어렵다. 그러나 GoogleNet은 이러한 문제를 고려하여 15억 개의 곱셈-덧셈 연산을 계산할 수 있는 예산을 유지하기 위해 설계되었다.

이 논문에서는 'Inception'이라는 Deep Neural Network Architecture에 초점을 맞추고 있다. Inception이라는 이름은 'Network in network' 논문에서 나온 유명한 대사(인터스텔라) 'we need to go deeper' 인터넷 밈에서 영감을 받았다. 여기서 'deep'은 'Inception module'과 'increased network depth' 두 가지 의미를 가지고 있다.

**2. Related Work**

LeNet-5을 시작으로 합성곱 신경망(CNN)은 일반적으로 표준 구조를 갖는다. 합성곱 레이어가 쌓이고 1개 이상의 FC 레이어가 이어지는 구조다. 또한 ImageNet과 같은 대규모 데이터에서의 최신 트렌드는 레이어 수와 사이즈를 늘리고, 오버피팅을 해결하기 위해 dropout을 적용한다. GoogleNet도 이와 같은 구조를 취하고 있다.

'network in network paper'에서 소개된 1x1 컨볼루션 레이어는 두 가지 목적으로 사용된다. 이를 활용하기 위해 GoogleNet도 이 레이어를 사용한다. 첫째는 병목 현상을 제거하기 위한 차원 축소이고, 둘째는 네트워크 크기를 제한하기 위해 사용된다. 이를 통해 네트워크의 깊이와 너비를 증가시킬 수 있으며, 유의미한 성능 저하 없이 수행할 수 있다.

**3. Motivation and High Level Considerations**

딥 뉴럴 네트워크는 현재 다양한 분야에서 매우 높은 성능을 보이고 있다. 그러나, 모델의 성능을 높이기 위해서는 레이어 수와 각 레이어의 유닛 수를 증가시켜야 한다. 이 방법은 일반적으로 더 많은 특징을 추출하고, 더 복잡한 패턴을 인식할 수 있도록 한다.

하지만, 이러한 방법에는 두 가지 단점이 있습니다. 첫째로, 네트워크 크기가 커질수록 파라미터 수가 증가하기 때문에, 학습 데이터의 수가 적은 경우에는 오버피팅이 발생하기 쉽다. 두번째로, 네트워크 크기가 커질수록 컴퓨터의 자원 사용량이 증가하는 문제가 있다. 이는 엄청난 연산량을 요구하며, 대부분의 기울기가 0이면 낭비되는 연산이 많아진다.

이러한 문제를 해결하기 위해, Google의 연구진은 Inception 구조를 생각했다. Inception 구조는 다양한 크기의 필터를 함께 사용하여 다양한 크기의 정보를 추출하는 모듈을 사용한다. 이를 통해 네트워크는 더 깊고 넓은 구조를 가질 수 있으며, 높은 정확도를 유지할 수 있다.

**4. GoogLeNet**

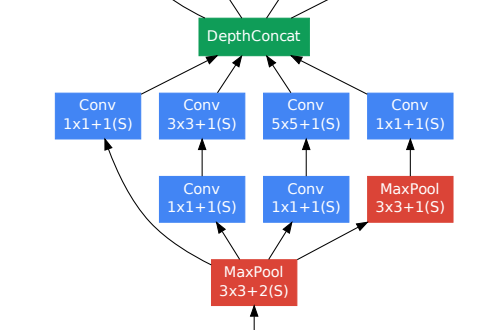
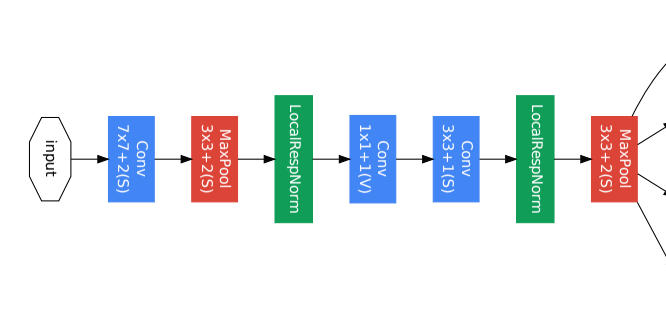
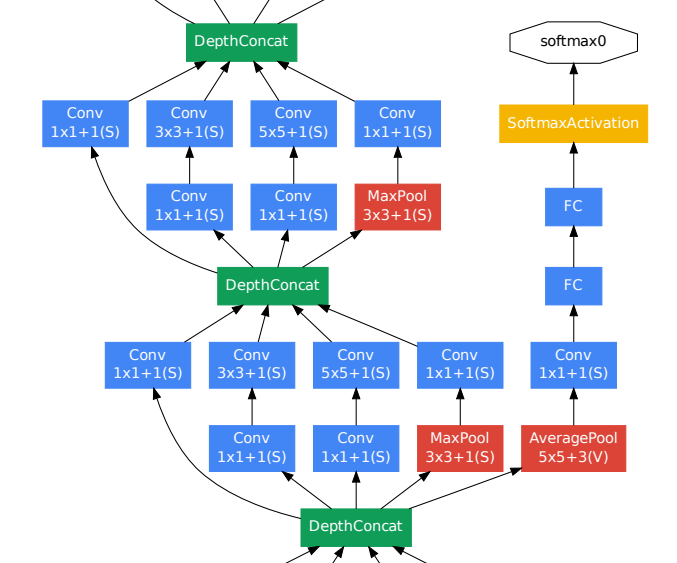
****

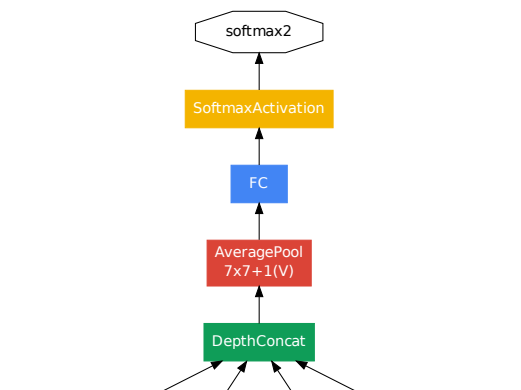
그림 2

그림 1

1. Input과 가장 가까운 낮은 레이어에 위치한 것으로, 기본적인 CNN 모델을 적용하고 있다.

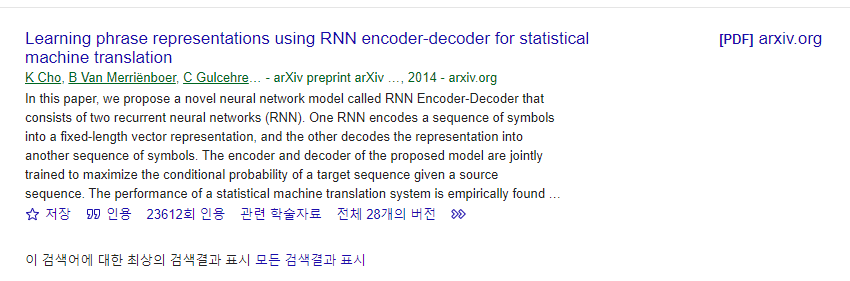
2. Inception 모듈은 1x1, 3x3, 5x5 컨볼루션 연산과 최대 풀링(max-pooling) 연산을 함께 사용한다. 1x1 컨볼루션 연산은 입력 데이터를 압축하고 파라미터 수를 줄인다. 이를 통해 입력 데이터의 크기를 줄이고, 모델의 복잡도를 낮출 수 있다. 3x3 컨볼루션 연산은 입력 데이터로부터 특징 맵을 생성한다. 이를 통해 모델은 입력 데이터의 다양한 특징을 추출할 수 있다. 5x5 컨볼루션 연산도 3x3 컨볼루션 연산과 마찬가지로 특징 맵을 생성합니다. 최대 풀링 연산은 입력 데이터에서 중요한 정보를 추출한다. 이를 통해 모델은 입력 데이터에서 가장 중요한 정보를 추출하여 다음 레이어로 전달할 수 있다.이러한 Inception 모듈은 여러 번 반복하여 사용된다. 각 Inception 모듈에서 추출한 특징 맵은 다음 레이어에서 다양한 특징을 추출하는 데 활용된다. 이를 통해 모델은 입력 데이터의 다양한 특징을 추출하고, 높은 정확도를 유지할 수 있다.

3. gradient vanishing 문제를 해결하기 위해서 중간에 intermediate result를 출력하여 추가적인 역전파를 유발하여 gradient 전달이 가능하게끔 했다. 이를 위해 auxiliary classifier를 적용했다.

4. 이는 마지막 예측 결과가 나오는 부분으로, average polling layer를 사용하고 있다. 이는 이전 layer에서 추출된 feature map을 각각 평균 내어 1차원 벡터로 만들어준 것으로, GAP(Global Average Pooling)이 적용된 결과다. 이것은 최종적으로 이미지 분류를 위한 softmax layer와 연결하기 위한 것이다.

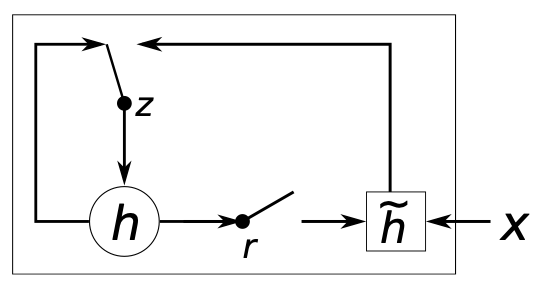
googLeNet은 이미지 분류와 관련된 다양한 응용 분야에서 높은 성능을 보이지만 Inception 모듈로 인해 복잡한 네트워크 구조를 가지고 있어 이해하고 어려운 부분이 있어 이보다 구조가 간단한 VGG모델을 많이 사용하기도 했습니다.

**Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation**



Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation.arXiv preprint arXiv:1406.1078.

이 논문은 Encoder, Decoder라고 불리는 두개의 RNN 구조를 사용한 네트워크와 새로운 형태의 hidden unit인 GRU를 제안한다.

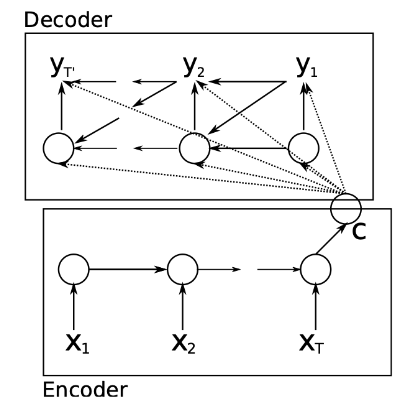
기존의 RNN 구조는 간단하게 설명하면 Hidden state를 통해 Input이 Output이 나오는 구조이다. 이번 논문에서는 새로운 형태의 Hidden State를 제안하고 있으며 위 그림은 새로운 Hidden State의 구조이다.

그림에서는 두 형태의 h와 h~이 나오는데, h를 Hidden state, h~를 현재의 Hidden state을 의미한다. new gate인 r은 Reset Gate로 이전의 Hidden state인 h를 얼마나 유지하고 반영할 것인지를 정하며 z는 Update Gate로 현재의 Hiddedn state인 h~을 얼마나 반영할 것인지를 정한다. r(Reset Gate)가 열려있을 수록, 즉 0에 가까울 수록 hidden state는 이전의 hidden state의 정보를 무시하고 새로운 input으로 업데이트 한다.

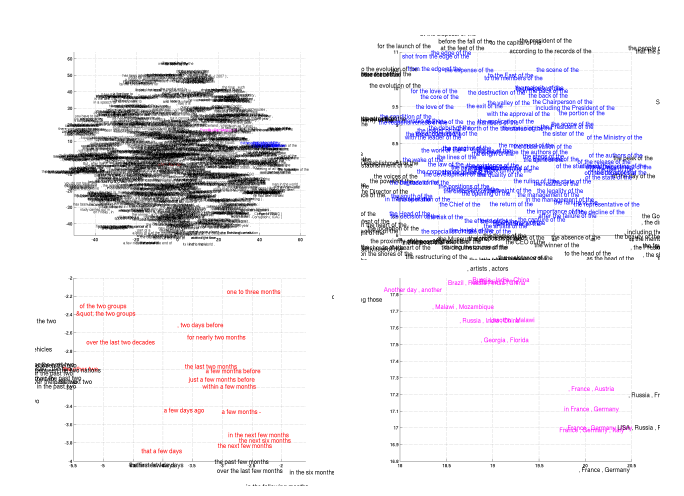


r(reset gate), z(update)에 대한 식은 위 그림과 같다. 간단히 설명하면 ResetGate는 Input(x)에 weight(Wr)를 곱한 값과 이전 hidden state(h<t-1>에 weight(Ur)를 곱한 값을 더해 활성화 함수 sigmoid function에 넣어 구한다.

위 식은 왼쪽 차례대로 update gate, new hidden state, h~를 구하는 식이며 위와같은 방식으로 구하게 된다.

이 그림은 이 논문의 핵심인 Encoder-Decoder 구조를 도식화한 그림이다. 위에 설명한 새로운 Hidden state GRU(Gated Recurrent Unit)을 빗대어 설명하려 한다. 먼저 Encoder에서는 input sequence x의 임베딩 벡터를 넣어 Hidden state를 update한다. “i hate exam”이라는 예시의 문장을 나누어 x1 = i x2 = hate, xT = exam이 들어가고 hidden state를 업데이트 하며 최종적 합쳐진 c라는 벡터가 만들어진다. Decoder은 Context Vector(c)를 추가로 입력받는 GRU 아키텍처를 활용하여 Target 문장을 생성하는 역할을 한다. Context Vector를 활용하여 생성하기 때문에, 이전 시점의 출력 결과와 현재 시점의 Context Vector를 입력으로 사용한다.

GRU와 Encoder와 Decoder를 통해 Sequence-to-Sequence 아키텍처를 제시했다. 가변적인 길이의 Source 문장의 문법적, 의미적 특징을 고정된 크기의 벡터로 압축할 수 있는 아키텍처이다. 위 설명한 Encoder와 Decoder로 구성되어 있으며 Encoder는 문장을 압축, Decoder는 문장을 생성하는 역할이다. Decoder의 문장 생성방법은 Hidden Vector에 Linear Layer를 추가하여 Vocabulary개수만큼 Vector를 생성하여 최종 Vector에 Softmax 함수를 취하면 특정 단어가 나올 확률을 의미하는 Vector를 생성할 수 있다. 앞선 말한 Encoder의 방법과 현재 설명한 Decoder의 방식으로 Sequence to Sequence 아키텍처는 Target문장을 Label로 활용하여 Decoder로부터 나온 확률을 기반으로 학습한다. 이 방식은 Gradient Decent 방식으로 학습한다. 이 방식은 Decoder와 Encoder 이용한다. Encoder의 소스 문장을 넣고, 순차적으로 Decoder를 이용하여 타켓 문장의 단어들이 생성 될 확률을 구하여 번역한다.

Encoder를 통해 압축된 Vector를 이용하여 의미적 문법적으로 유사한 구문들이 뭉쳐져 있는 것을 확인하는 시각화 자료이다.

즉 이 논문에서는 GRU아키텍처를 제공하여 Input, Output, Hidden state인 LSTM보다 간단하게 Update gate, Reset Gate 두 노드로 계산량을 줄였다는 점이 핵심이다. 또한 Seq-to-Seq 아키텍처의 Encoder, Decoder로 구성하여 가변적인 길이를 압축 생성할 수 있다는 점에서 기본에 번역기계와는 더 정확하고 큰 데이터셋에 적응할 수 있다는 점이 핵심이었다.

이 논문은 번역기 또는 다양한 자연어 처리에서 널리 사용되는 기계 번역의 핵심기술의 초기라고 볼 수 있다. 그래서 굉장히 다양한 부분에서 더욱 발전하여 사용하고 있다. 자연어 질의 답변 생성, 대화시스템, 문서요약 등의 분야에서 이 모델을 활용하여 높은 성능을 보이고 있다.