컴퓨터비전

Assignment1 - Dive into LeNet

컴퓨터 비전
1분반
소프트웨어학과
32204041
정다훈

Understand the Full Design and Reasoning behind LeNet-5 by Reading the Original Paper

Introduction

“The main message of this paper is that better pattern recognition systems can be built by relying more on automatic learning, and less on hand-designed heuristics.” - page 1

논문의 저자는 기존에 사용하던 손으로 특징 추출하는 방식은 픽셀 이미지에 직접 수행되는 자동 기계 학습으로 대체될 수 있다고 말합니다. 다시 말해, Graph Transformer Networks라고 부르는 well-principled design paradigm이 기존 모듈 통합 방식을 대체할 수 있다고 말합니다. natural data가 풍부해지고 다양해짐에 따라 정확한 인식 시스템을 전부 손으로 만들기란 불가능합니다.

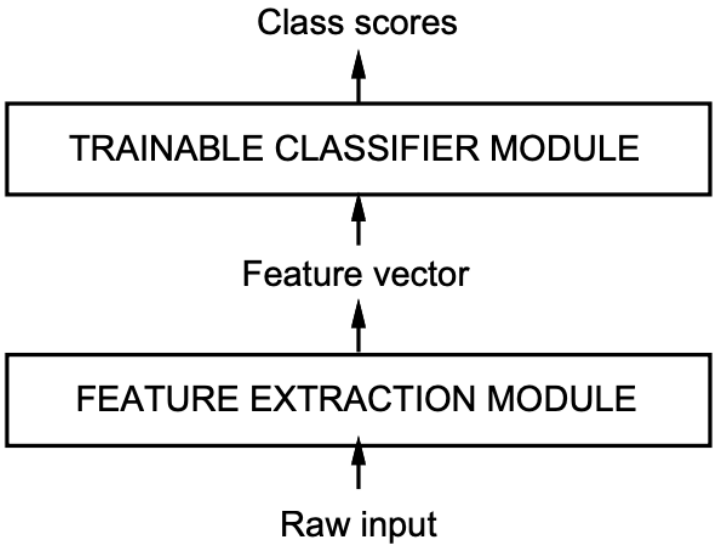


Fig. 1. Traditional pattern recognition is performed with two modules: a fixed feature extractor, and a trainable classifier.

전통적인 패턴 인식은 Figure 1에서 말하는 바와 같이 Feature Extraction Module로 저차원의 임베딩을 진행한 후 Trainable Classifier Module을 학습하여 수행했습니다. 이런 과정의 주된 이유는 Classifier를 이용한 학습 기술이 저차원 공간에 한정됐기 때문입니다. 이어서 저자는 세 가지 요인이 지난 10년간 이어져온 ‘vision’을 변화시킨다고 말합니다.

1. 저 비용의 연산 유닛을 사용한 machine은 brute-force “numerical” methods를 가능하게 만든다.
2. 방대해진 데이터는 hand-crafted feature extaction에서 벗어나 real data 자체에 의존할 수 있도록 한다.

3. 고차원의 inputs을 다룰 수 있는 강력한 머신러닝 기술이 가능해지면서 거대한 datasets만 있다면 적절한 decision functions을 생성할 수 있다.(가장 중요)

이 세가지 요인에 대한 증거로 대부분의 현대 상업용 OCR 시스템은 back-propagation을 이용하는 multi-layer Neural Network를 사용한다.

이 논문이 다루는 연구에서는 다음과 같은 내용을 다룬다.

- 손글씨 문자 인식 Task (Sections 1 and 2)
- 다른 learning techniques과 손글씨 숫자 인식에 대한 데이터에 대해 벤치마크로 성능 비교를 한다. (Section 3)
- 단일 문자 인식부터 문장 인식까지 전반적인 오류를 줄이기 위한 아이디어인, combining multiple mmodules trained를 소개한다. (Section 4)
- 또한 module이 직접 graphs를 조작한다면 다양한 길이의 손글씨 객체를 multi-module systems를 이용해 인식하는 것은 Graph Transformer Network(GTN)의 개념을 주도하는데 이것도 소개한다. (Section 4)
- 단어 또는 문자열 인식을 위해 사용하는 고전적 방법인 heuristic over-segmentation 방식을 설명한다. (Section 5)
- 수작업 세분화와 레이블링이 필요 없는 단어 수준의 인식을 위한 gradient 기반 기법을 다룬다. (Section 6)
- 입력 전체 위치를 스캔하여 segmentation heuristics를 제거할 수 있는 Space-Displacement Neural Network 접근법을 소개한다. (Section 7)
- GTN이 일반화된 변환으로 수식화될 수 있음을 보이고, GTN과 Hidden Markov Model의 연결성도 설명한다. (Section 8)
- 펜 입력 장치에서 실시간으로 글씨를 인식하는 on-line handwriting recognition 시스템을 다룬다. 이 시스템의 핵심은 CNN이며, 단어 수준에서 학습하는 장점을 보여준다. (Section 9)
- 손글씨 및 기계 인쇄 문서를 인식하는 GTN 기반 시스템을 설명한다. 핵심은 LeNet-5이며, 실제로 NCR 은행권 체크 인식 시스템을 상용 적용되어 미국 전역의 은행에서 매달 수백만 건의 체크를 처리하는 사례를 제시한다. (Section 10)

Introduction을 읽으면서 헷갈렸던 용어를 정리합니다.

1. Discriminative

Deterministic vs Discriminative

보통 Deterministic은 Stochastic과, Discriminative는 Generative와 비교됩니다.

Deterministic은 결정론적이라는 의미입니다. 같은 입력이 들어가면 항상 동일한 출력이 나오는 시스템을 언급할 때 사용합니다.
반면 Stochastic은 확률적이라는 의미입니다. 같은 입력이라도 무작위성이 개입되어 출력이 달라질 수 있음을 의미합니다.
동일 입력에 대해 결과가 고정되냐, 무작위성이 있냐에 대한 단어들입니다.

Discriminative는 판별적이라는 의미입니다. 판별 모델은 입력 데이터가 주어졌을 때, 직접적으로 클래스 또는 라벨을 구분하는 방식입니다. 이에 따라 모델링 하는 것은 조건부 확률입니다. P(y|x). 즉, “이 입력 x가 주어졌을 때, 라벨 y가 무엇일까?”를 모델링합니다.

Generative는 생성적이라는 의미입니다. 생성적 모델은 데이터와 라벨의 분포 자체를 모델링합니다. 다시 말해, 결합 확률 P(x,y) 또는 조건부 확률 P(x|y): “이 클래스 y일 때, 데이터 x가 어떻게 생겼는가?”에 대한 모델링을 합니다.

Section 6에 대한 Overview에서 언급된 ‘Discriminative’라는 단어를 톺아봤습니다. 결론적으로 ‘Discriminative’는 손글자 숫자 이미지를 입력하면 바로 숫자 클래스를 출력하는 판별적인 LeNet-5 분류기에 대한 언급을, ‘non-discriminative’는 데이터와 라벨의 분포를 모델링해서 인식하는 비판별적인 HMM 기반 접근에 대한 언급을 한 것이라고 해석할 수 있습니다.

2. Hidden Markov Model(HMM)

HMM 기반 접근은 관측 데이터가 어떤 숨겨진 상태들의 순차적 전이 과정에서 생성된다고 가정하는 확률적 모델을 바탕으로 합니다. 간단히 다시 말해서 “보이지 않는 상태가 있고, 그 상태가 일정한 확률에 따라 다음 상태로 넘어가면서 관측 가능한 데이터가 생성된다”는 가정입니다.

손글씨 인식이나 음성 인식과 같은 시퀀스 데이터 문제에서 HMM은 글자나 음소 같은 단위들을 “숨겨진 상태”로 두고, 실제 관측되는 픽셀 시퀀스나 음향 특징을 “관측값으로 봅니다. 이 모델은 다음 두 가지를 학습합니다. 첫 번째로, 각 상태가 다른 상태로 전이할 확률, 즉 전이 확률과 두 번째로, 상태에서 특정 관측값이 나타날 확률, 즉 출력 확률을 함께 학습합니다. 예를 들어, 손글씨 단어 이미지를 세로로 스캔하면서 얻은 특징 시퀀스를 입력으로 넣으면, HMM은 상태 전이와 출력 확률을 고려해 가장 가능성 높은 문자 시퀀스를 찾아냅니다. 이 접근의 핵심은 순차적 의존성을 단순한 확률 구조로 모델링해, 이전 상태에 따라 현재 상태가 정해진다는 “마르코프 성질”을 이용한다는 점입니다.

LeNet-5는 기본적으로 단일 글자 인식용 CNN입니다. 즉, 입력으로 숫자나 알파벳 한 글자 이미지가 들어오면, 그 글자를 분류하는 데 매우 잘 맞는 구조입니다. 하지만 현실 문제에서는 글자가 따로따로 주어지지 않고, 연속된 문자열(단어, 수표의 금액, 주소, 문장) 형태로 등장합니다. 이 경우에는 단순히 CNN으로는 글자 경계(segmentation)을 찾기가 어렵습니다.

여기서 HMM 기반 접근이 보완책으로 사용됩니다. 예를들어, 수표에 적힌 “TWENTY”라는 단어를 인식한다고 해봅시다.

1. 이미지 전체를 일정한 스텝으로 스캔하면서 LeNet-5가 각 위치에서 “이게 T일 확률, W일 확률, E일 확률…”을 출력합니다.
2. 이 확률값들을 HMM의 출력 확률(emission probability)로 사용합니다.
3. HMM은 상태 전이 확률(영어 단어에서 W 다음에 E가 올 가능성 등)을 고려하여, 전체 시퀀스 중 가장 자연스러운 문자 조합을 찾습니다.
4. 결과적으로 CNN은 국소적은 글자 판별을, HMM은 연속적인 단어 인식을 담당하게 됩니다.

HMM의 학습 과정에서는 주로 Baum-Welch 알고리즘 같은 Expectation-Maximization(EM) 절차가 쓰여 파라미터의 최대우도를 추정합니다. 인식 단계에서는 Viterbi 알고르짐을 사용해 주어진 관측 시퀀스에 대해 가장 가능성 높은 상태 시퀀스를 효율적으로 찾습니다. 이 방식은 사람이 직접 데이터를 세분화하지 않아도 연속적인 입력을 처리할 수 있다는

장점이 있습니다. 다만 관측 분포를 단순한 가우시안 혼합 같은 것으로 가정하기 때문에 복잡한 패턴을 충분히 표현하지 못하는 한계도 있습니다.

2-1. 최근 **HMM**은 어느 정도의 **position**에 있는가**?**

전통적으로 HMM은 시퀀스 데이터 분야에서 표준 방식이었고, 비교적 단순하고 해석 가능하다는 장점이 있었습니다. 하지만 데이터가 더 많아지고 모델이 복잡해질수록, 그리고 문맥이나 장기 의존성(long-term dependencies)이 중요해질수록 HMM은 한계를 보이기 시작했습니다. 위에서 언급했듯이 “현재 상태가 이전 상태 하나만 영향을 받는다”는 transition 가정이 있어서 복잡한 패턴을 포착하는 데 어려움이 있었습니다.

HMM이후에는 RNN / LSTM / GRU 처럼 연속 시퀀스 입력에서 상태(state)를 내부적으로 유지하면서 장기 의존성을 모델링하는 기법과 Transformer 계열과 같이 attention 메커니즘으로 시퀀스 전체의 관계를 손쉽게 모델링하는 기술을 사용합니다. 특히 attention 메커니즘을 통해 문장 길이가 길거나 복잡할 때도 잘 작동하도록 하는 것이 가능해졌습니다.

하지만 HMM이 완전히 사라진 건 아니고, 특히 데이터가 적거나, 해석 가능성이 중요하거나, 작은 자원 환경 같은 곳에서는 여전히 유용합니다.

자연어 처리와 같은 Sequence 데이터 처리 분야를 공부하면서 RNN 이후의 모델들에 대해서는 숱하게 다뤘지만 그들의 선조격인 HMM에 대해서는 처음 공부했습니다. 비록 LeNet-5의 단일 character 인식과는 무관하지만 논문을 읽으면서 흥미로운 내용을 찾아 알아가는 것은 즐거운 것 같습니다.

answer the following questions based on your reading and implementation

Introduction의 내용을 기반으로 질문 답변에 필요한 Sections를 찾아 정독하고 질문에 대한 답변을 정리합니다.

1. Why did LeNet-5 use RBF in the output layer instead of softmax?

LeNet-5 논문 Section 2에서는 출력층을 “Gaussian connections”으로 설명하면서, 각 출력 유닛이 RBF(radial basis function)처럼 동작하도록 설계했다고 밝힙니다. 이는 각 클래스가 하나의 중심점(prototype vector)으로 표현되고, 입력 특징 표현이 이 중심점에 얼마나 가까운지를 측정하는 방식입니다. 다시 말해, softmax처럼 단순히 “확률 분포”를 산출하는 대신, 거리 기반 유사도를 직접 반영하는 구조입니다. 이 접근은 각 클래스 간의 경계를 더 명확히 만들고, 학습 시 출력 공간에서 안정적인 수렴을 돕는 역할을 했습니다. 따라서 논문에서 제시된 RBF 출력층은 거리 기반 분류를 강조한 설계라 할 수 있습니다.

당시에는 오늘날처럼 softmax가 분류 문제의 사실상 표준으로 자리잡지 않았습니다. 대신 연구자들은 패턴 인식에서 자주 쓰이던 RBF 네트워크의 개념을 차용해 출력층에 적용했습니다. RBF는 입력이 특정 클래스 중심에 얼마나 가까운지를 평가하기 때문에, 잘못된 클래스와의 차이를 크게 벌려주는 효과가 있었습니다. 이는 특히 손글씨 인식처럼 클래스 간 모양이 유사한 경우, 확률적 점수보다는 근접성 평가가 더 직관적이고 안정적이라고 판단했기 때문입니다. 이런 배경에서 LeNet-5는 softmax 대신 Gaussian RBF 기반 출력을 채택했습니다.

이 선택은 결과적으로 LeNet-5가 단순한 분류기를 넘어 특징 공간(feature space)에서의 구조적 정보를 활용하도록 만들어 주었습니다. 출력층이 단순한 확률 예측이 아니라, “이 입력이 클래스 중심에 얼마나 가까운가”라는 해석을 가능하게 했던 것이죠. 이는 학습된 특징 벡터가 클래스별 클러스터와 잘 정렬되도록 유도했고, 결과적으로 더 나은 일반화를 기대할 수 있었습니다. 물론 현대 CNN에서는 softmax가 주류로 자리잡았지만, LeNet-5의 RBF 출력층 설계는 초기 딥러닝에서 “출력층을 통해 분류 기준을 어떻게 정의할 것인가”라는 중요한 실험적 시도였습니다. 따라서 이 설계는 당시 연구의 시대적 맥락과 실험적 철학을 잘 보여주는 사례라고 할 수 있습니다.

2. How is LeNet-5 different from modern CNN architectures?

LeNet-5는 입력을 32×32 크기의 이미지로 받아 합성곱 계층과 서브샘플링 계층을 번갈아 쌓은 뒤, 전결합 계층과 RBF 기반 출력층으로 이어지는 구조를 가지고 있습니다. 전체 깊이는 7층 정도로 얕으며, 합성곱 필터 수도 매우 제한적이어서 파라미터 수와 연산량이 적습니다. 현대의 CNN과 비교했을 때 이 단순한 구조는 당시 하드웨어 자원의 한계를

반영한 것입니다. 또한 활성화 함수로 시그모이드나 탄젠트 함수 계열을 사용했는데, 이는 깊은 네트워크에서 기울기 소실 문제를 더 심각하게 만들었습니다. 따라서 LeNet-5는 오늘날의 CNN에 비해 훨씬 단순하고 제약적인 설계를 가지고 있었습니다.

현대의 CNN은 LeNet-5에 비해 훨씬 깊고 복잡한 아키텍처를 갖추고 있습니다. 예를 들어 VGG, ResNet, EfficientNet 같은 모델은 수십에서 수백 개의 층을 가지고 있으며, 훨씬 더 많은 필터와 파라미터를 활용합니다. 또한 Batch Normalization, Dropout 같은 정규화 기법이 추가되어 학습 안정성과 일반화 성능이 크게 향상되었습니다. 활성화 함수도 ReLU나 그 변형들이 사용되어 기울기 소실 문제를 완화하고 빠른 수렴을 가능하게 했습니다. 이러한 차이들은 LeNet-5 이후 수십 년간 딥러닝 연구가 어떻게 발전했는지를 잘 보여줍니다.

출력층 설계에서도 현대 CNN은 softmax를 표준으로 채택하여 확률적 해석을 가능하게 하는 반면, LeNet-5는 RBF 기반 출력을 사용했습니다. 또한 현대 CNN은 residual connection, attention, depthwise separable convolution 등 다양한 아이디어를 도입해 더 효율적이고 강력한 모델을 만들었습니다. LeNet-5는 파라미터 수가 적고 단순한 구조 덕분에 작은 데이터셋에 적합했지만, 대규모 데이터와 복잡한 과제를 해결하기에는 한계가 있었습니다. 반대로 현대 CNN은 대규모 데이터셋과 GPU 연산을 기반으로 훨씬 확장된 성능을 보여줍니다. 결국 LeNet-5와 현대 CNN의 차이는 단순히 깊이와 성능뿐만 아니라, 적용할 수 있는 문제의 범위와 처리 가능한 데이터의 규모에서 뚜렷하게 드러납니다.

3. Why is C3 only partially connected to S2?

LeNet-5의 C3 계층은 S2 계층으로부터 입력을 받지만, 모든 맵이 전부 연결되는 것이 아니라 일부만 연결되는 부분 연결 방식을 채택했습니다. 예를 들어 C3의 16개 특징 맵 중 일부는 S2의 6개 맵 중 3~4개에만 연결되는 식으로 제한되었습니다. 이는 당시 컴퓨팅 자원이 부족해 파라미터 수를 줄이고, 학습 효율성을 높이려는 실용적인 이유가 있었습니다. 또한 완전 연결 대신 부분 연결을 도입함으로써 불필요하게 과도한 연산을 피할 수 있었습니다. 결과적으로 C3는 자원 제약과 모델 설계상의 절충을 반영한 계층이었습니다.

부분 연결은 단순히 계산량을 줄이는 것뿐만 아니라, 모델에 의도적인 제약(architectural constraint)을 주는 효과도 있었습니다. 특정 맵들이 선택적으로 연결되면서, 네트워크는 S2의 여러 특징 맵을 무조건 모두 통합하지 않고 부분적인 패턴만 조합하도록 유도되었습니다. 이렇게 하면 서로 다른 맵이 학습하는 특징이 더 다양해질 수 있고, 중복된 표현을 줄일 수 있습니다. 다시 말해, 부분 연결은 특성 추출 단계에서 더 다양한 필터링 조합을 가능하게 하여 표현력을 높이는 역할을 했습니다. 이는 당시의 네트워크 설계에서 데이터 효율성과 일반화 성능을 확보하기 위한 중요한 아이디어였습니다.

이러한 부분 연결 설계는 오늘날의 CNN 구조와 비교할 때 독특한 특징으로 남아 있습니다. 현대 CNN은 보통 합성곱 층이 모든 입력 채널에 연결되는 완전 연결 방식을 사용하지만, LeNet-5는 이보다 더 제약된 연결을 실험적으로 도입했습니다. 이는 학습 데이터가 제한적이던 시절, 파라미터 수를 줄이고 오버피팅을 방지하는 효과적인 방법이었습니다. 동시에 다양한 맵 조합을 통해 더 복잡한 패턴을 학습할 수 있도록 설계된 점에서 의미가 있습니다. 따라서 C3의 부분 연결은 자원 한계와 학습 안정성을 동시에 고려한 설계 선택이었다고 볼 수 있습니다.

4. Why is LeNet-5 still important today?

LeNet-5는 합성곱 신경망(CNN)의 기본 구조를 정립한 역사적 모델로서 여전히 중요한 의미를 갖습니다. 합성곱 계층과 풀링 계층을 번갈아 배치하는 구조는 오늘날의 CNN에도 그대로 계승되고 있습니다. 당시 손글씨 숫자 인식이라는 비교적 단순한 문제를 다뤘지만, 이미지 특징을 계층적으로 추출해 분류하는 방식은 범용적인 아이디어였습니다. 이 때문에 LeNet-5는 CNN이 실질적으로 가능하다는 것을 최초로 입증한 사례로 평가받습니다. 따라서 현대 딥러닝 연구의 출발점으로서 교과서적인 위치를 차지하고 있습니다.

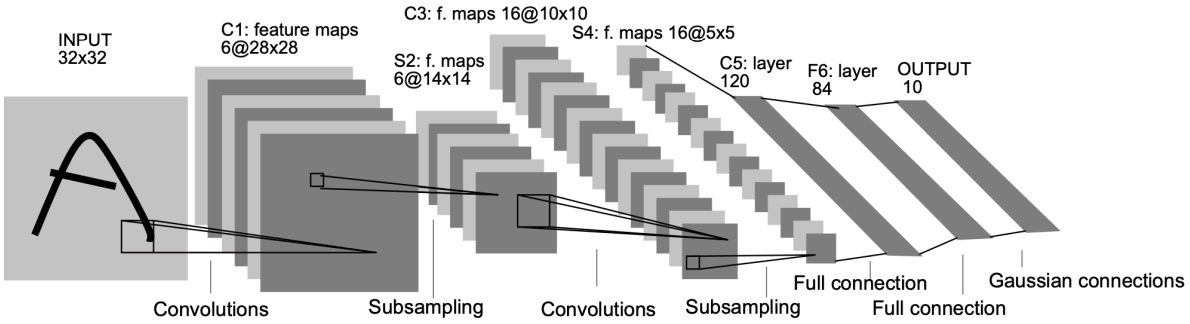
LeNet-5는 실제 산업적 응용에서도 중요한 역할을 했습니다. 예를 들어 은행권 수표(cheque) 인식 시스템에 적용되어, 매달 수백만 건의 수표를 자동으로 처리하는 데 사용되었습니다. 이는 단순한 연구 수준을 넘어 딥러닝 모델이 실제 문제를 해결할 수 있다는 가능성을 보여준 대표적인 사례입니다. 이후 이미지 처리, 음성 인식, 자연어 처리 등 다양한 분야에서 딥러닝이 확산될 수 있었던 것도 LeNet-5 같은 모델의 실용적 성공 덕분입니다. 따라서 LeNet-5는 연구와 산업 사이를 연결한 첫 번째 성공적인 CNN 사례라 할 수 있습니다.

비록 오늘날의 모델에 비하면 LeNet-5는 얕고 단순하며 성능적으로도 제한적이지만, 교육과 연구에서 여전히 학습용 예제로 활용됩니다. 초보자가 CNN 구조를 이해하고 실습하기에 적합한 크기와 복잡도를 가지고 있기 때문입니다. 또한 현대의 복잡한 모델을 이해할 때, LeNet-5를 출발점으로 비교하면 발전 과정을 쉽게 파악할 수 있습니다. 즉, LeNet-5는 단순한 옛 모델이 아니라, 딥러닝의 발전사를 설명하고 이해하는 데 반드시 거론되는 중요한 토대입니다. 결국 LeNet-5는 기술적 기원, 산업적 성공, 교육적 가치라는 세 가지 측면에서 여전히 중요한 의미를 지니고 있습니다.

Implement and train LeNet-5 on the MNIST dataset using Google Colab and Provide your training results (accuracy, loss curves, or screenshots of output) as part of your report.

Implement 설명

```python
# LeNet-5 아키텍처 구현 (원본 다이어그램에 맞춘 버전)
class LeNet5(nn.Module):
    def __init__(self, num_classes=10):
        super(LeNet5, self).__init__()

        # Feature extraction layers
        # C1: 32x32 -> 28x28 (6 feature maps)
        self.conv1 = nn.Conv2d(1, 6, kernel_size=5, stride=1, padding=0)  # 32x32 -> 28x28
        # S2: 28x28 -> 14x14 (subsampling)
        self.pool1 = nn.AvgPool2d(kernel_size=2, stride=2)                # 28x28 -> 14x14

        # C3: 14x14 -> 10x10 (16 feature maps)
        self.conv2 = nn.Conv2d(6, 16, kernel_size=5, stride=1, padding=0) # 14x14 -> 10x10
        # S4: 10x10 -> 5x5 (subsampling)
        self.pool2 = nn.AvgPool2d(kernel_size=2, stride=2)                # 10x10 -> 5x5

        # Classifier layers
        # C5: 16*5*5 -> 120 (fully connected convolution)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)  # 400 -> 120
        # F6: 120 -> 84 (fully connected)
        self.fc2 = nn.Linear(120, 84)
        # Output: 84 -> 10 (classification layer)
        self.fc3 = nn.Linear(84, num_classes)

    def forward(self, x):
        # Feature extraction
        x = torch.tanh(self.conv1(x))
        x = self.pool1(x)

        x = torch.tanh(self.conv2(x))
        x = self.pool2(x)

        # Flatten for fully connected layers
        x = x.view(x.size(0), -1)

        # Classifier
        x = torch.tanh(self.fc1(x))
        x = torch.tanh(self.fc2(x))
        x = self.fc3(x)

        return x
```



논문에서 제시하는 구조에 따라 LeNet-5를 구현했습니다. 각 층에 대한 설명은 다음과 같습니다.

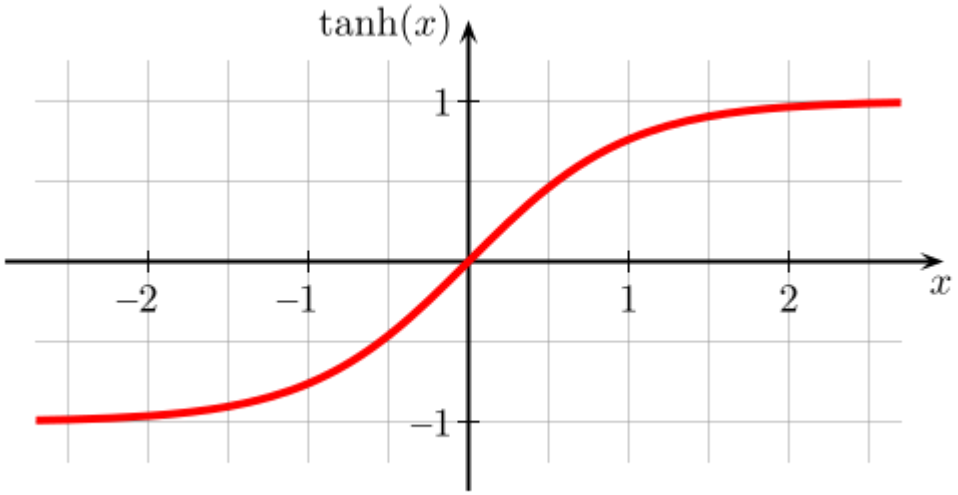
입력층은 32x32 크기의 단일 채널 이미지를 받아들이는 역할을 합니다. MNIST 데이터셋의 원본 크기인 28x28을 각 면에 2픽셀씩 패딩하여 32x32로 확장함으로써 원본 LeNet-5의 설계에 맞춰 구성됩니다. 패딩 과정을 통해 이미지의 가장자리 정보를 보존하고, 후속 Conv 연산에서 특징 맵의 크기가 급격히 줄어드는 것을 방지하는 역할을 수행합니다.

C1층은 6개의 5x5 Conv 필터를 사용하여 32x32 입력에서 28x28 크기의 6개 특징 맵을 생성합니다. 이 층의 주요 목적은 edge, 선분, 코너와 같은 저수준 시각적 특징을 추출하는 것입니다. 각 필터는 서로 다른 방향성과 패턴을 학습합니다. tanh 활성화 함수를 적용하여 비선형성을 도입하며, 총 156개의 학습 가능한 파라미터를 가집니다.

C1층의 파라미터 개수 계산 과정과 결과

- **Number of Parameters in C1**
  - (5 x 5 x 1 + 1) x 6 = 156 parameters
    (bias)
    filter size / channel size : gray scale / the number of filters.
    - ✓The size of each filter is **5 × 5**.
    - ✓Input has **1** channel
    - ✓Plus **1** bias, which is also a trainable parameter
    - ✓**6** filters (i.e., 6 output channels)

tanh 활성화함수 **-** 비선형성 도입



S2층은 2x2 평균 풀링을 통해 28x28 특징 맵을 14x14로 다운샘플링합니다. 이 과정은 공간적 해상도를 절반으로 줄여 계산 복잡도를 감소시키면서도 중요한 특징 정보는 보존하는 역할을 합니다. 평균 풀링은 지역적 변환에 대한 불변성을 제공하여 모델이 작은 위치 변화에 강건하게 만들며(invarient), 과적합을 방지하는 정규화 효과도 제공합니다.

S2층의 파라미터 개수 계산 과정과 결과

- **Number of Parameters in S2**
  - (1 weight + 1 bias) x 6 = 12 parameters

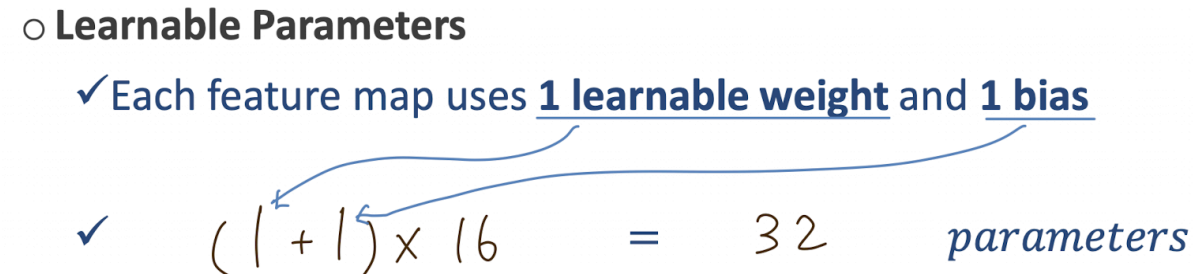
C3층은 16개의 5×5 컨볼루션 필터를 사용하여 14×14 입력에서 10×10 크기의 16개 특징 맵을 생성합니다. 이 층은 C1에서 추출된 저수준 특징들을 조합하여 더 복잡하고 추상적인 패턴을 학습하며, 숫자의 곡선, 교차점, 특정 형태 등을 인식할 수 있습니다. 총 2,416개의 파라미터(16×6×5×5 + 16개 바이어스)를 가지며, 각 출력 특징 맵은 이전 층의 여러 입력 특징 맵과 연결되어 풍부한 특징 표현을 가능하게 합니다. 이 층의 중요한 특징 중 하나는 S2층과 partially connected된다는 점입니다. 부분 연결을 통해 특성 추출 단계에서 더 다양한 필터링 조합을 가능하게 하여 표현력을 높이는 역할을 했습니다. 이는 당시의 네트워크 설계에서 데이터 효율성과 일반화 성능을 확보하기 위한 중요한 아이디어였습니다.

C3층의 파라미터 개수 계산 과정과 결과

Group	Formula	Parameters
Group 1	(5x5x3+1) x 6	456
Group 2	(5x5x4+1) x 6	606
Group 3	(5x5x4+1) x 3	303
Group 4	(5x5x6+1) x 1	151

S4층은 다시 2×2 평균 풀링을 적용하여 10×10 특징 맵을 5×5로 축소합니다. 이는 두 번째 차원 축소 단계로서 특징 맵의 공간적 크기를 더욱 줄여 후속 완전연결층에서 처리해야 할 데이터 양을 대폭 감소시킵니다. 이 단계에서 각 5×5 특징 맵은 원본 이미지의 상당히 큰 수용 영역(receptive field)을 대표하게 되어, 숫자 전체의 구조적 정보를 압축적으로 담게 됩니다.

S4층의 파라미터 개수 계산 과정과 결과



C5층은 16×5×5=400개의 입력을 120개의 뉴런으로 완전연결하는 층으로, 사실상 컨볼루션의 특수한 형태로 볼 수 있습니다. 이 층은 공간적으로 분산된 특징들을 하나의 벡터로 통합하여 고수준의 추상적 특징을 학습하며, 숫자 분류에 직접적으로 유용한 특징들을 추출합니다. tanh 활성화 함수를 사용하여 48,120개의 파라미터(400×120 + 120개 바이어스)를 가지며, 특징 추출에서 분류로의 전환점 역할을 합니다.

C5층의 파라미터 개수 계산 과정과 결과

○ **Learnable Parameters**

✓Each output feature map has $5 \times 5 \times 16 = 400$ weights

✓Plus **1 bias**

✓ $(5 \times 5 \times 16 + 1) \times 120 = 48120$ ***parameters***

F6층은 120개의 입력을 84개의 뉴런으로 연결하는 완전연결층으로, 분류를 위한 최종 특징 정제 과정을 담당합니다. 이 층은 C5에서 추출된 고수준 특징들을 더욱 압축하고 정제하여 최종 분류에 최적화된 표현을 만들어냅니다. 84차원의 출력은 원본 LeNet-5에서 ASCII 문자 인식을 위해 설계된 크기로, 10,164개의 파라미터(120×84 + 84개 바이어스)를 가지며 tanh 활성화 함수를 사용합니다.

F6층의 파라미터 개수 계산 과정과 결과

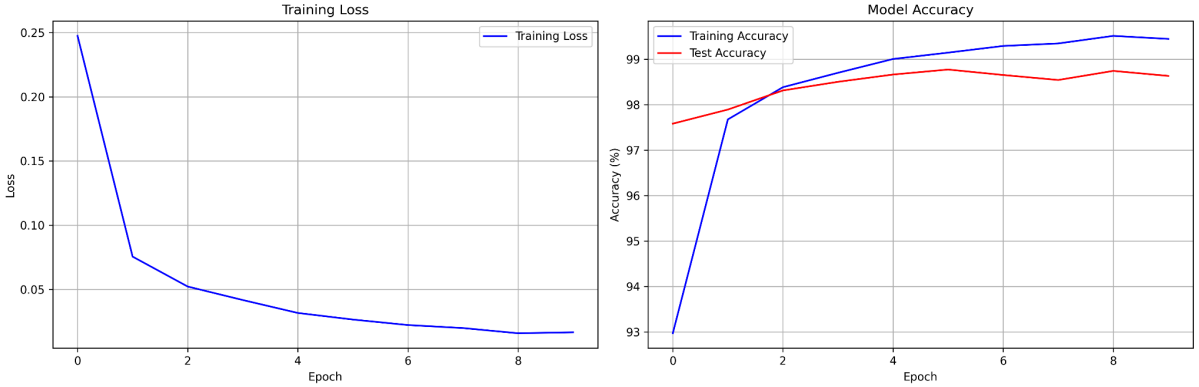
○ **Parameters**

✓Each output neuron → 120 weights + 1 bias

✓ $(120 + 1) \times 84 = 10164$ ***parameters***

출력층은 84개의 입력을 10개의 클래스(0-9 숫자)에 대응하는 출력으로 변환하는 최종 분류층입니다. 이 층은 소프트맥스 함수가 적용되기 전의 로짓(logit) 값을 출력하며, 각 값은 해당 숫자 클래스에 대한 신뢰도를 나타냅니다. 총 850개의 파라미터(84×10 + 10개 바이어스)를 가지며, 훈련 시에는 크로스 엔트로피 손실 함수와 함께 사용되어 실제 레이블과의 차이를 최소화하는 방향으로 전체 네트워크를 학습시킵니다.

Training Loss & Model Accuracy



Training Loss 곡선에서 초기 급격한 감소를 보이는 것을 확인할 수 있는데, 이는 모델이 초기에 기본적인 패턴을 빠르게 학습하고 있음을 의미합니다. MNIST와 같은 상대적으로 단순한 데이터셋에서 LeNet-5가 효과적으로 특징을 추출하고있다는 증거입니다. 이를 통해 아키텍처가 손글씨 숫자 인식에 매우 적합하게 설계되었음을 보여줍니다.

2 Epoch 이후부터는 손실이 완만하게 감소하여 최종적으로 약 0.017 수준에서 안정화됩니다. 이는 모델이 과적합 없이 올바르게 학습을 진행하고 있음을 나타냅니다. 10 Epoch에서도 손실이 계속 감소하는 추세를 보이므로, 더 많은 epoch로 훈련하면 추가적인 성능 향상이 가능할 것으로 예상됩니다.

Model Accuracy 곡선에서는 초기에 빠른 학습 속도를 보임을 확인할 수 있습니다. 이는 MNIST 데이터의 패턴이 비교적 단순하고 일관성이 있으며, LeNet-5가 이러한 패턴을 효과적으로 학습할 수 있는 충분한 표현력을 가지고 있음을 보여줍니다. 테스트 정확도는 97.6%에서 시작하여 최종적으로 98.6% 수준으로 안정화되며, 훈련 정확도와 약 0.8%p의 차이를 보입니다. 이는 올바르게 일반화됨을 나타내며 심각한 과적합이 발생하지 않았음을 의미합니다.

위 결과 곡선들을 통해 전체적으로 매우 안정적인 학습이 진행됐음을 알 수 있습니다. 손실과 정확도 모두 과적합이나 언더피팅 없이 적절한 수준에서 수렴하고 있습니다.

느낀점

지난 2년간 머신러닝 & 딥러닝에 대해 공부해오면서 많은 어려움이 있었습니다. 힐튼 교수님의 AlexNet을 필두로 딥러닝의 붐이 오기 이전부터 연구되어온 여러 주제들을 압축하여 한 번에 소화하려다보니 의도치 않게 첫 술에 배부르려고 애쓰는 학생이 됐던 것이 아닌가 싶습니다. 최근 대학원 진학 준비를 하며, 공부해왔던 내용들을 타임라인을 짚어가며 차근차근 따라가보는 시간을 가졌습니다. 그제서야 맞춰지는 퍼즐들에 그려진 연구의 흐름과 큰 그림들이 이해되는 경험을 했습니다. 그러다보니 이 과제물을 하면서 많은 즐거움을 느낀 것 같습니다. 교수님께서 설계하신 컴퓨터비전 클래스의 커리큘럼이 학생들에게 차근차근의 미학을 보여주시려는게 아닌가 하는 기대감도 들었습니다. 이 과제를 할당해주신 교수님께 감사합니다.