편집위원 : 강신각(한국전자통신연구원)

인공지능 오픈소스 라이브러리 텐서플로우(tensorflow)와 인공지능 응용 소프트웨어 개발

조만석
래블업

요 약

본고에서는 인공지능 딥러닝(Deep Learning) 기술은 최근 2012년 신경정보처리시스템 학회(NIPS)에서 Alex-net[1]을 시작으로 크게 발전하였고, 2016년 바둑에서 이세돌 9단과 인공지능 알파고(AlphaGo)의 대국에서 4:1로 알파고가 승리함으로 크게 세간의 주목을 받으며 학술 연구에서는 물론, 산업과 예술 작품에까지 응용이 확산되고 있다. 딥 러닝(Deep Learning)은 인간의 뇌를 모방한 인공 신경망(Artificial Neural Network) 기술을 사용합니다. 인공지능 딥러닝을 활용한 컴퓨터 비전 및 자연 언어 처리 · 음성 인식 등의 기초 연구가 진행되어 감에 따라 이를 응용하여 다양한 산업 분야에서 활용이 폭 넓게 진행되는 것이다. 본 고에서는 이러한 기술 진보에 필요한 딥러닝(Deep Learning) 개발 소프트웨어 도구의 종류와 가장 인기있는 텐서플로우(TensorFlow), 그리고 어떤 인공지능 응용 어플리케이션 연구가 진행되는지에 대하여 논한다.

Ⅰ. 서 론

대규모 기계 학습 시스템, 텐서플로우 (Tensorflow)[2]는 2015 년 11 월에 구글(Google, 미)이 오픈 소스로 공개 한 기계 학습 인공지능 소프트웨어이다. 구글 브레인 팀(Google Brain Team)은 구글이 가진 대용량의 데이터를 활용하여 구글 서비스 사용자가 더욱 사용하기 쉽도록 다양한 영역에 적용할 수 있도록 사내 소프트웨어 라이브러리를 개발했다. 구글 브레인 팀은 2011년에 릴리즈하였으며 그 이름을 한 디스트빌리프(DistBelief)라고 하였다. 구글의 디스트빌리프는 기계학습 연구 개발 및 구글의 다양한 서비스 제품에 적용을 목적으로 하였다.

디스트빌리프는 구글 제품을 개발하는 소프트웨어 라이브러리로 내부에서만 제한하여 사용했다. 구글은 디스트빌리프를 이용하여 인터넷검색엔진(Google Internet Search Engine), 유튜브(YouTube),구글자동번역(Google Translate), 등을 개발했다.

구글은 대용량의 데이터를 처리하기 위한 요구 조건으로 첫 번째 복잡하고, 거대한 데이터 세트를 훈련 할 수 있어야 하고, 둘째, 데이터 연산에 필요한 계산 자원을 최대한 활용할 수 있을 것, 그리고 누구나 사용하기 쉬운 인터페이스로 유연성과 확장 성이 있어야 할 것이었다. 이러한 조건을 모두 만족하며 기계학습 도구로 구글의 자체 분산 처리를 위한 라이브러리 DistBelief의 경험을 바탕으로 확장 개발하였으며 오픈 소스로 공개한 기계 학습을 위한 분산처리 기반 심층학습 (Deep Learning) 소프트웨어가 텐서플로우이다. 인공 지능 연구자는 텐서플로우를 사용하여 인공 신경망 구조를 설계하고, 분산 컴퓨팅 기술을 이용하여 딥러닝 학습을 통해서 암과 같은 질병을 진단[3]하거나 사람처럼 회화를 할 수 있는 인공지능 서비스를 할 수 있다.

Ⅱ. 본 론

1. 인공 신경망 (Artificial Neural Network)

인공신경망(Artificial Neural Network, ANN)은 생물학의 신경망, 동물의 중추신경계 중 뇌에서 영감을 얻어 기계학습과 인지과학에서 정의한 통계학적 학습 알고리즘이다. 동물의 중추 신경계는 신경 세포(Neuran) 이라는 단위로 구성되며 신경세포는 서로 시냅스라는 결합을 통해서 신호를 전달한다. 신경세포 네트워크를 수학으로 모델을 만든 것이 인공 신경망이다. 이것은 인공 뉴런의 시냅스 결합으로 네트워크를 형성하고 각 시냅스의 결합 세기(weight)를 변화시켜 문제 해결 능력을 가지는 모델을 가리킨다.

인공 신경망은 통계학적 모델의 집합이 첫째 조정이 가능한 가중치의 집합으로 학습 알고리즘에 의해서 조정이 가능한 숫자로 표현된 매개 변수로 구성되어 있고, 둘째 입력의 비선형 함수를 유추할 수 있으면 인공 신경이라고 칭한다.

인공신경망은 학습 모델에서 원하는 정답 데이터에 의해서 각 시냅스의 결합 세기를 최적화하는 교사 학습(Supervisor Learning)과 정답 데이터를 필요로 하지 않는 비 교사 학습(Non-Supervisor Learning)이 있다. 데이터로부터 학습한다는 점에서 기계학습 알고리즘과 공통점을 가지지만 인공신경망을 응용한 심층 학습은 규칙 기반의 기존 알고리즘이 풀기 어려운 컴퓨터 비전 또는 음성 인식처럼 다양한 분야에서 차별화된다.

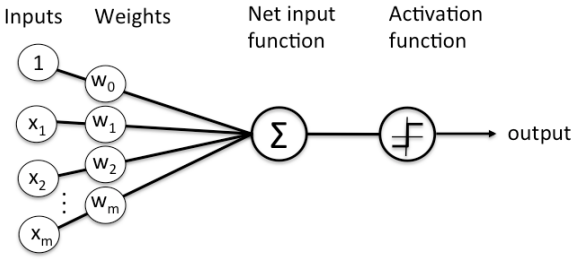

그림 1. 인공 신경 세포의 정의

2. 인공 신경망의 역사

1940년대 후반 심리학자 도널드 헤비안(Donald Hebb)[4]은 신경가소성의 원리에 근거한 자율 학습 네트워크를 만들었다. 팔리(Farley)와 웨슬리 클라크(Wesley A.Clark)(1954)[5]는 헤비안 네트워크를 모의 실험하기 위해서 계산학 모델(Computer)를 사용하였다. 프랑크 로젠블라트(Frank Rosenblatt)(1958)[6]는 간단한 덧셈과 뺄셈을 하는 이층 구조의 학습 컴퓨터 망, 퍼셉트론(Perceptron)으로 패턴 인식을 위한 알고리즘을 만들었다. 또한 그는 배타적 논리합 회로(exclusive-or circuit)를 퍼셉트론 알고리즘으로 만들었다. 폴 웨어보스 (Paul Werbos, 1975)[7]은 배타적 논리합 문제를 오차 역전파법으로 효과적으로 처리할 수 있게 되었다. 하지만 배타적 논리합 회로를 계산하기 위해서는 컴퓨터가 충분히 빨라지고 효과적으로 처리할 때까지 인공 신경망 연구는 침체되었다. 스위스 AI연구실 IDSIA에서 위르겐 슈밋흠바 (Jürgen Schmidhuber). (2009-2012)[8]의 연구 그룹은 재귀 신경망과 심화 피드포워드 신경망을 만들어 여덟 번의 패턴 인식과 기계학습 국제 대회에서 우승하였다. 복잡한 신경 세포들의 네트워크에 영감을 받은 조프 힌턴(Geoff Hinton)의 표준 비전 아키텍쳐는 자율학습 방법으로 인간과 어깨를 나란히 하거나 넘어서는 결과를 보여주었다.

3. 인공 신경망 시스템 (Artificial Neural Network System)

개별 인공 신경은 서로 연결된 네트워크로 구성된다. 여기서 신경 네트워크란 서로 여러 층의 신경 층으로 구성한다. 인공신경망은 첫째 각 층의 신경들 사이의 연결 패턴과 둘째 신경 연결의 가중치를 계산하는 학습 과정, 그리고 마지막으로 신경의 가중 입력을 활성화 정도에 따라 출력으로 바꾸어 주는 활성화 함수로 정의한다.

인공 신경망의 한 층은 여러 개의 노드로 구성된다. 심층 신경망(Deep Neural Network) 이란 여러 개의 층으로 이루어진 신경망을 의미한다. 각 신경은 여러개의 입력으로 구성되며 일정 크기 이상의 자극을 받으면 반응을 하는데, 그 반응의 크기는 입력 값과 각 입력의 계수(또는 가중치, weights)를 곱한 값와 대략 비례합니다. 일반적으로 노드는 여러 개의 입력을 받으며 입력의 개수 만큼 계수를 가지고 있습니다. 따라서 이 계수를 조절함으로써 여러 입력에 다른 가중치를 부여할 수 있습니다. 최종적으로 곱한 값들은 전부 더해지고 그 합은 활성함수(activation function)의 입력으로 들어가게 됩니다. 활성함수의 결과가 노드의 출력에 해당하며 이 출력값으로 자율 학습된다.

단일 층 신경망과 심층 신경망(Deep Neural Network)의 차이는 신경 망의 층의 깊이이다. 심층 학습 이전의 신경망은 하나는 입력과 하나는 출력 층을 사용하였으며 혹은 하나의 은닉층(Hidden layer)을 사용했고, 그래서 이러한 신경망을 얕다(Shallow Neural Network)라고 하였다. 심층 신경망은 입력과 출력 층을 포함하여 3개가 넘는 층, 즉 2개 이상의 은닉층을 가지기 때문에 깊은 학습(deep learning) 이라고 한다.

4. 오픈소스 인공지능 소프트웨어 라이브러리 텐서플로우 (Tensorflow, An open-source software library for Machine Intelligence)

구글 브레인 팀(Google Brain Team)은 구글 제품에 사용되는 기계학습(Machine Learning) 소프트웨어 라이브러리(Software Library)이다. 문서, 사진, 동영상 등의 대용량의 빅데이터를 연구연구하고 제품 서비스 개발을 위한 목적으로 2011년 디스트빌리프(DistBelief)를 내부에서 공개해서 사용하였다. 디스트빌리프는 구글이 서비스하거나 연구로 사용하는 검색, 음성인식, 광고, 사진 서비스, 지도서비스, 번역, 유튜브 등 기계학습 연구 개발 및 구글의 다양한 서비스 제품에 적용되었다.

텐서플로우(Tensorflow)는 구글 브레인 팀이 디스트빌리프를 발전시켜 2015년 11월 오픈소스로 공개한 두번째 기계학습을 위한 소프트웨어 라이브러리이다. 테 자사의 인공지능 소프트웨어 라이브러리, 텐서플로우는 맥오에스(maxOS), 윈도우즈 10의 데스트탑과 워크스테이션, 우분투 리눅스, 레드햇 리눅스등 서버 시스템에서 안드로이드, 아이오에스(iOS) 같은 스마트폰까지 지원을 한다. 텐서플로우의 인터넷 홈페이지에는 텐서플로우를 이용하는 유명한 기업의 로고가 다음과 같이 쭉 늘어서있다.

그림 2. TensorFlow를 이용하는 기업

오픈 소스가 공개되는 깃허브(GitHub.com)에서 2015년 11월 공개 후 공개된지 1년만에 가장 주목 받는 기계 학습 라이브러리 중 하나가 되었으며 2년이 지난 지금 많은 가장 많은 호응(74,000여개 Star, 36,000여 Fork)를 받았다.

텐서플로우는 오픈 소스 소프트웨어 라이브러리로 소스 코드와 활용 모델이 공개되어 480명 이상의 개발자가 참여하여 소프트웨어 품질을 높이며, TFLearn과 TensorLayer, PrettyTensor 같은 래퍼 라이브러리도 등장하였다. 또한 구글 사내에서는 이미 60 개 이상의 프로젝트에서 텐서플로우가 TensorFlow이 채용되었다[11]. 이러한 실적도 텐서플로우가 가장 인기있는 기계학습 라이브러리가 된 하나의 요인 일 것이다.

5. 딥 러닝 라이브러리 (Deep Learning Library)

소셜 네트워크 서비스(Social Network Service, SNS) 사용자들에 생산되는 다량의 자료와 태그 정보들은 많은 양의 데이터가 생성되며, 스마트 디바이스와 사물 인터넷(Internet Of Things)등 개인에 요구에 맞춤한 다양한 서비스를 제공하고 여기서 생성된 대량의 데이터를 수집하고 정형화 , 데이터 베이스로 정제하여 분석하기 위한 분산 컴퓨팅 기술의 발달로 딥 러닝의 기술 한계점이던 느린 학습시간은 분산 컴퓨팅 시간이 크게 단축되었으며, 그래픽 처리기(Graphic Processing Unit, GPU)는 심화 신경망 학습에 필요한 복잡한 행렬 연산에 소요되는 시간을 줄여주었다.

분산 컴퓨팅, GPU등을 지원하는 인공지능 딥 러닝 라이브러리를 주요 특징별로 분류하여 표로 정리 한 것이다.

표 1. 딥 러닝 라이브러리 비교

라이브러리	제작사	프로그래밍 스타일	디바이스 대응	CUDA	분산 대응	주요언어
Tensorflow	Google	선언형	GPU, Mobile	O	O	Python, C++
Torch7	Facebook	명령형	GPU, FPGA	O	O	Lua,C
Theano	Univ.Montreal	선언형	GPU	O	O	Python
Caffe	Univ.Berkeley	명령형	GPU	O	O	C++, Python
Chainer	Perferred Networks	명령형	GPU	O	O	Python
MXNet	DMLC	선언형+명령형	GPU, Mobile	O	O	Python, C++
CNTK	Microsoft	선언형	GPU	O	O	Python, C++

딥 러닝 라이브러리의 주된 차이점으로 첫째 프로그래밍 스타일, 둘째 분산 대응 멀티 GPU 여부, 세째 프로그래밍 언어를 꼽을 수 있다. 신경망 네트워크를 프로그래밍하는 스타일은 명령형 구현 방법과 선언형 구현 방법이 있다.

6. 선언형 프로그래밍 (Declarative Programming)

프로그램이 어떤 방법으로 해야 하는지를 나타내기 보다는 무엇과 같은지를 설명하는 것을 선언 형 프로그래밍이라고 한다. 이것은 전통적인 프로그래밍 언어인 포트란, C, 자바와 같은 명령형 프로그래밍 언어와 다른 접근 방식이다. 문제를 해결하는 알고리즘을 프로그래밍 언어로 명시하고, 실행 목표에 대하여 명시하지 않는 것이 명령형 프로그래밍 언어였다면, 실행 목표는 명시하지만 알고리즘은 명시하지않는 것이 선언형 프로그래밍 언어이다. 선언형 프로그래밍의 예를 들면 SQL 쿼리는 "어떤 데이터를 갖고 싶은가"를 기술한다. 하지만 구체적으로 “이진 나무 검색 (binary tree search) 작업으로 데이터 베이스에 액세스한다”라는 알고리즘은 명시하지 않는다. 이 기술은 FORTRAN, C 언어, Java와 같은 명령형 프로그래밍 언어와는 전혀 다르다. 명령형은 목적을 실현하는 알고리즘을 그 순서에 따라서 서술해야한다.

즉, 명령형 프로그래밍은 목적을 달성하기위한 방법을 "절차"로 나타내는 반면 선언적 프로그래밍은 달성해야 할 목표(출력)을 기술하고, 그것을 실현하는 절차는 시스템에 맡기는 셈이다 .

7. 데이터 흐름 그래프 아키텍처와 선언형 프로그래밍 모델 (Data flow graph architecture & Declarative programming)

TensorFlow 소스 코드가 공개되어있는 GitHub에 는 TensorFlow을 다음과 같이 소개하고있다.

"Computation using data flow graphs for scalable machine learning "

데이터 흐름 그래프는 데이터와 계산에서 유효 그래프것이다. 특히 TensorFlow에서는 데이터 가져 오기, 전처리, 계산, 상태, 출력까지 일관되게 데이터 플로우 그래프를 구축하여 처리하는 것을 염두에 두고 설계되었다.

TensorFlow의 핵심 기본 개념은 데이터 파이프 라인을 구축하는 것이다.

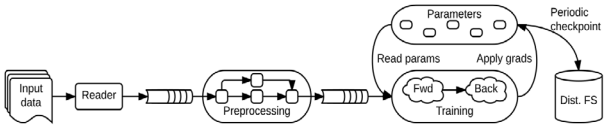

그림 3. 데이터 파이프 라인

데이터가 막대하게 되고, 매개 변수가 많아서 학습 모델이 복잡 해짐에 따라 하나의 시스템에서 신속하게 학습 수렴하지 않는 문제가 있다.

예를 들어, Google DeepMind이 바둑에서 이세돌 씨에 승리한 AlphaGo는 CPU 서버를 1902 개, GPU 서버를 176 개 사용하여 싸운 것이다.(참조2)

즉, 명령적인 스타일은 어떻게 (How) 계산이 행해지는지를 지정하는 한편, 선언적인 스타일은 어떻게 있어야 할 것인가 (What)을 기술한다.

명령형 프로그래밍은 기계 학습을 위한 알고리즘을 프로그래밍하고 데이터는 실행할 때 구축된 알고리즘에 따라서 데이터를 처리하는 과정을 거쳐서 알고리즘의 결과를 얻는다. 이와 대비되는 선언형 프로그래밍은 입력 데이터에 의한 데이터 처리는 계산에 대한 결과를 선언하여 입력 데이터부터 각 수치연산, 상태변수, 저장까지 일관된 과정을 그래프로 선언하여 계산 파이프 라인을 구축해서 실행한다.

8. 데이터 흐름 그래프 (data flow graph)

선언형 프로그래밍에 의해서 단위 인공 신경을 구현하고 각 단위 인공 신경을 데이터 흐름 그래프로 확장하여 구성한다. 데이터 흐름 그래프는 노드(node)와 에지(edge) 로 구성된다. 노드는 일반적으로 산술 연산 또는 변수를 나타내며 데이터를 읽거나 계산 결과를 기록 할 수 있다. 가장자리(엣지, edge)는 어떤 크기의 다차원 배열 또는 텐서를 나타낸다. 노드는 실제로 계산하는 장치가 할당되며 병렬/비동기 병렬로 실행된다.

예시로 동등한 배열의 입력(input, x) 과 가중치(weight, W) 초기화하고 활성화 함수(activation function) ReLU를 구성을 텐서 플로우의 파이썬 코드로 예로 보자.

```
import tensorflow as tf

# 100 차원 벡터를 0 으로 초기화
b = tf . Variable ( tf . zeros ([ 100 ]))
# 784x100 행렬을 -1 에서 1 로 임의의 값(random) 으로 초기화
W = tf . Variable ( tf . random_uniform ([ 784 , 100 ], - 1 , 1 ))

x = tf . placeholder ( name = "x" )
relu = tf . nn . relu ( tf . matmul ( W , x ) + b )
```

[코드: 단위 신경망을 텐서플로우 파이선으로 구현한 코드]

예시의 코드는 행렬 입력 X를 가중치 W의 곱과 바이어스 B의 덧셈, 그리고 활성화 함수(ReLU)를 적용하여 계산 흐름 그래프를 적용하면 세개의 노드가 데이터의 입출력을 사용한다.

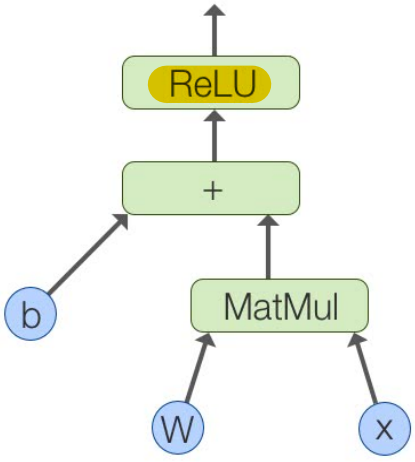

그림 4. 단위 신경망을 데이터 흐름 그래프로 묘사

9. 분산 학습(distribute computation)

텐서플로우는 다양한 환경에서 작동하는 것을 목적으로 개발했다. 라즈베리파이 같은 작은 장치부터 iOS 나 Android 등 모바일 장치, 스마트 허브, 구글 홈등 가정 자동화 기기, 데스크탑, 워크스테이션과 분산된 서버 환경까지 사용할 수 있다. 분산된 서버환경에서는 데이터와 학습을 병렬 처리할 수 있다. 하드웨어로 인텔의 x86 과 ARM의 CPU(Central Processing

Unit)과 함께 계산 용도로 특화된 GPU(Graphic Processing Unit) 그리고 구글에서 기계학습을 위하여 제작한 TPU(Tensor Processing Unit)을 지원한다. 사용자 환경으로 리눅스, 윈도우즈 10, 맥오에스(macOS) 등 다양한 운영체계를 지원한다.

또한 Google처럼 대량의 학습 데이터를 보유한 기업이 개발하고있는 것도 있고, 그런 빅 데이터를 대량의 컴퓨팅 리소스를 사용하여 학습 할 수 있다

10. 분산 실행 모델 (Distribute Execution Model)

텐서플로우는 선언형 문법에 따라 프로그래밍을 통하여 데이터 플로우 그래프를 정의한 후 사용자(client)가 계산 작업 세션(Session)을 실행(run)하면 데이터 흐름 그래프가 구성되어 실행된다. 인공 신경망이나 기계학습을 위한 많은 양의 계산을 분산 처리하기 위해서 마스터(master)와 실행자(worker)에 나누어지며 실행자는 각 데이터 흐름 그래프 작업을 장치(CPU, GPU, TPU)에 작업을 할당한다.

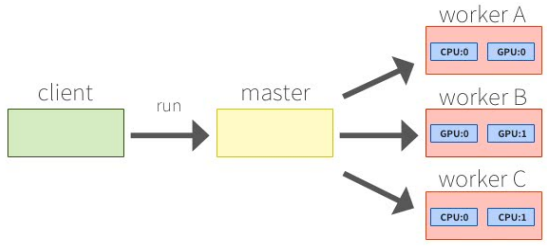

그림 5. 데이터 플로우 그래프를 분산 처리

마스터 프로세스는 데이터 흐름 그래프 작업을 나누어서 각각의 개별 계산 작업을 실행자에 할당 한다. 각 실행자는 할당된 계산 장치(CPU, GPT, TPU)의 실제 계산 및 제어를 한다.

11. 매개변수 서버(Parameter Server)

여러 대의 컴퓨터에 분산 학습할 때 각각의 매개 변수의 상태와 클러스터 정보 등 기계 간의 통신 매개 변수를 공유 할 필요가 있다. 이러한 공유 상태를 저장하는 데이터베이스가 매개변수 서버(Parameter Server)이다.

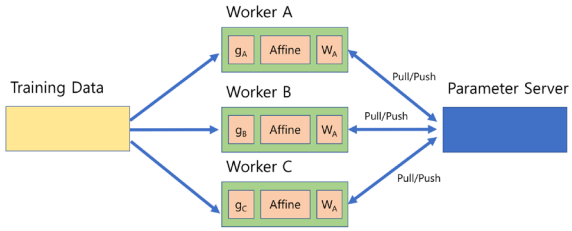

그림 6. 매개변수 공유

매개변수 서버의 동작을 확률적 경사 강하 법(Gradient Descent)의 계산으로 예를 들면, 마스터 노드는 데이터를 처리하기 위해서 미니 배치(mini batch) 알고리즘으로 실행자가 계산을 할 수 있도록 잘게 쪼개어진 데이터 플로우 그래프를 실행자에 할당합니다. 쪼개어 할당할때 여러 실행자에서 분산되어 실행되기 때문에 구별을 할 수 있는 키값을 함께 생성하여 실행자에게 보냅니다(Key & Value Pull). 실행자는 자신의 계산 장치(CPU, GPU, TPU)에 작게 쪼개어 나누어진 경사 강하 법의 부분 계산 하여 그 결과 값과 할당으로 받은 키를 매개변수 서버에 메시지로 업데이트 요청을 보냅니다 (Key & Value Push). 마스터가 분산 계산할 데이터 흐름 그래프가 있다면 계속해서 실행자는 계산에 필요한 매개변수 값을 가져와서 (Pull) 실행자가 가진 매개 변수(Key parameter)를 업데이트합니다.

텐서플로우에서 제공하는 매개변수 서버는 일관성(Consistency) 및 장애 허용(Fault Tolerance)에 유연한 확장성을 겸비한 아키텍처로 구성한다.

12. 장애 허용(Fault Tolerance)

여러 대의 컴퓨터에서 학습을 분산했다고 해도 학습 모델을 훈련하기 위해서는 일반적으로 몇 시간에서 며칠, 길면 몇 달 같은 시간을 들여 계산해야 한다. 학습을 분산 실행하면 통신 실패 및 장비의 고장 등 장애가 발생할 수 있다. 텐서플로우는 이러한 장애를 감지하면 전체 계산 그래프의 실행을 중지하고 처음부터 다시 구축 메커니즘이 있다. 학습 중에 파라미터의 상태 체크 포인트를 써 두는 것으로 재구성 다시 때 마지막으로 저장 한 저장소에서 상태를 복원 할 수 있다.

13. 시각화 도구 텐서보드(TensorBoard)

복잡한 깊은 학습의 모델이 오면 계산 그래프의 노드와 매개 변수는 늘어 디버깅이 어려워진다. 예를 들면 LSTM(Long Short-Term Memory)를 사용한 RNN(Recurrent Neural Network) 모델을 텐서플로우로 만들면 약 15,000 개 이상의 노드가 데이터 흐름 그래프가 사용됩니다. 영상 인식에 사용하는 CNN(Convolution Neural Network) 중에서 2014년 구글이 발표한 인셉션(Inception) 모델은 약 36,000 개 이상의 노드가 필요합니다[11].

이러한 네트워크가 제대로 구축되어 있는지를 확인하고 매개변수의 폭발이나 손실이 발생하지 않았는지 확인하기 위해 텐

서플로우는 텐서 보드라는 강력한 시각화 도구를 제공한다. 텐서 보드는 다음의 시각화 기능을 제공한다.

- 계산 그래프
- 스칼라 값의 그래프
- 이미지
- 음성
- 임베디드 표현

14. 그래프 시각화 (Graph Visualization)

깊은 학습의 네트워크 구성은 상당히 복잡하고 제대로 구현되어 있는지를 코드만 보면서 결과에 대한 확증을 갖는 것은 매우 어려울 수 있다. 이런 점을 효과적으로 이해를 돕기 위해서 계산 그래프를 시각화하여 코드뿐만 아니라 그래프를 보고 실수를 고치는데 도움을 받을 수 있다.

다음 예시는 영상인식에 주로 사요되는 CNN(Convolutional Neural Network) 모델을 텐서플로우로 정의한 파이썬 소스 코드의 예이다.

```
def cnn(x, y):
    x = tf.reshape(x, [-1, 28, 28, 1])
    y = slim.one_hot_encoding(y, 10)

    net = slim.conv2d(x, 24, [5,5],
scope='conv1')
    net = slim.max_pool2d(net, [2,2],
scope='pool1')
    net = slim.conv2d(net, 56, [5,5],
scope='conv2')
    net = slim.max_pool2d(net, [2,2],
scope='pool2')
    net = slim.flatten(net,
scope='flatten3')
    net = slim.fully_connected(net, 512,
scope='FC4')
    net = slim.fully_connected(net, 10,
          activation_fn=None,
scope='FC5')

    prob = slim.softmax(logits)
    loss =
slim.losses.softmax_cross_entropy(logits,
y)

    train_op = slim.optimize_loss(loss,
slim.get_global_step(),
          learning_rate=0.001,
          optimizer='Adam')

    return {'class': tf.argmax(prob, 1),
'prob': prob},\
          loss, train_op
```

텐서플로우가 CNN 모델을 실행될 때 텐서 보드로 시각화하면 다음의 그림처럼 된다. 입력은 맨 아래 입력(Input)라는 이름이 붙었다. 입력이 있고, softmax라는 함수 부분까지 일직선으로 연결된 것을 그림으로 알 수 있다.

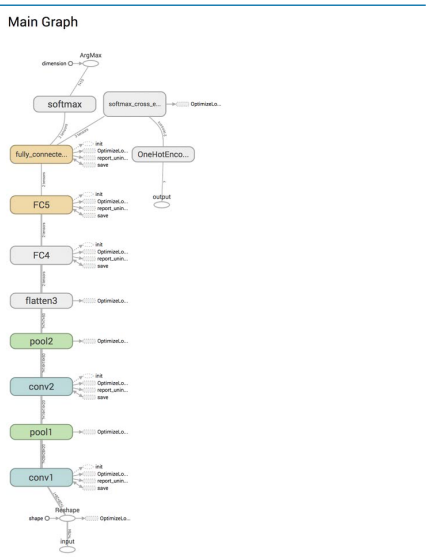

그림 7. Convolution Neural Network을 데이터 흐름 그래프로 구성하여 시각화

예시로 든 인공 신경망CNN을 구성하는 코드가 잘못했다면 어떻게 알 수 있을까. 코드를 다음과 같이 수정하여 실행 하여 텐서보드로 데이터 흐름 그래프를 시각화하면 다음 그림처럼 된다.

```
def cnn(x, y):
    x = tf.reshape(x, [-1, 28, 28, 1])
    y = slim.one_hot_encoding(y, 10)

    net = slim.conv2d(x, 24, [5,5],
scope='conv1')
    net = slim.max_pool2d(net, [2,2],
scope='pool1')
    net = slim.conv2d(net, 56, [5,5],
scope='conv2')
    net = slim.max_pool2d(net, [2,2],
scope='pool2')
    net = slim.flatten(net,
scope='flatten3')
    net = slim.fully_connected(net, 512,
scope='FC4')
    net = slim.fully_connected(x, 10,
          activation_fn=None,
scope='FC5')

    prob = slim.softmax(logits)
    loss =
slim.losses.softmax_cross_entropy(logits,
y)

    train_op = slim.optimize_loss(loss,
slim.get_global_step(),
          learning_rate=0.001,
          optimizer='Adam')
```

인공 신경망 CNN 함수에서 출력 계층으로 slim.flatten 함수를 사용했다. 이 함수의 첫 번째 인수를 net 로 해야 하지만 시각화 오류 예시에선 x 라고 했다. 이런 코드를 작성할 때 발견하지 어려울 수 있다. 많은 변수가 사용되기 때문에 코드에서 수정하는 것은 어렵다. 못하고 데이터 흐름 그래프가 실행하면 즉 학습이 실행 되면 예상하지 못한 결과를 얻을 뿐만 아니라 원인을 찾는데 많은 시행 착오가 있을 수 있다. 데이터 흐름을 선언하고 그것은 텐서 보드를 통해서 시각화하면 이러한 오류를 줄이는데 크게 도움을 받을 수 있다.

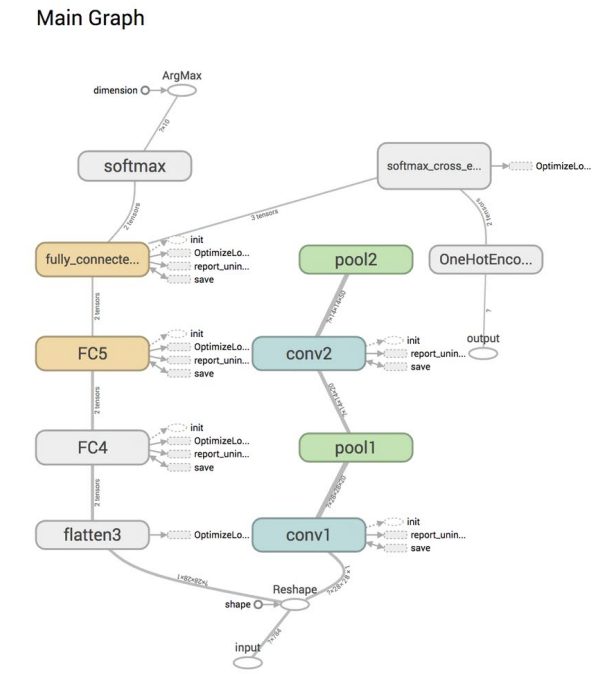

그림 8. 인공 신경망 CNN의 slim.flatten 함수의 입력을 net에서 x로 잘못 입력하였을 때 텐서보드에서 시각화한 데이터 흐름 그래프

시각화된 그래프를 보면 Convolutoin 층과 Pooling 층이 전체 결합 층과 연결되어 있지 않았다. 즉 코드가 아닌 시각화된 그래프의 도움을 받아서 자신의 코드 실수를 즉시 알 수 있다.

또한 텐서플로우의 시각화 도구는 임의의 값도 로그 값의 선 그래프나 로그 값의 히스토 그램을 시각화 할 수 있다.

텐서플로우의 학습 로그를 시각화함으로 해서 학습이 진행 중일때 일정한 시간마다 텐서 값과 계산 로그를 시각화 해주며 히스토그램의 분포를 통해서 시각화된 값을 확인할 수 있다.

다음의 그래프는 학습중인 오차가 떨어지는 모습과 정답률이 올라가는 모습을 그래프로 그린 것이다. 학습이 진행됨에 따라 오차와 정답률을 그래프 값과 경향을 자세히 볼 수 있으므로, 매개변수를 조정하여 튜닝 하거나 학습 결과를 확인하는데 사용할 수 있다.

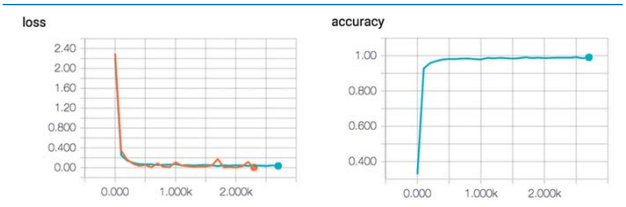

그림 9. 텐서보드의 학습 오차와 정답률 표현

텐서보드는 로그 뿐만 아니라 음성과 화상을 시각화 할 수 있다. 이를 이용하여 데이터 확장과 데이터를 가공 할 때 제대로 처리 할 수 있는지 여부를 확인할 수 있다. 다음은 인공 신경망 CNN의 입력 이미지를 가시화 한 예이다.

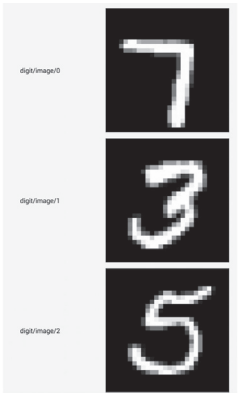

그림 10. 텐서 보드의 입력 이미지를 가시화

텐서보드의 시각화로 신경망뿐만 아니라 추천 시스템 자연 언어 처리도 이용할 수 있으며 하여 t-SNE와 PCA 방식으로 차원 줄일 수있다.

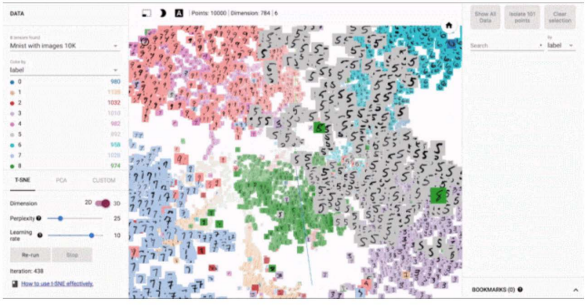

그림 11. 텐서보드로 MNIST를 시각화

15. 인터페이스의 단순화

텐서플로우는 데이터 플로우 그래프를 처리하는 구조로 설계되어 명령형 언어에 비하여 함수가 적고 연구자가 사용하기

에는 아직은 인터페이스가 친숙하지 않기 때문에 이런 것을 보완하고자 텐서플로우를 백엔드로 한 TFLearn, TensorLayer, skflow 같은 래퍼 라이브러리가 개발되었으며 연구자 및 프로토 타입 개발에 응용할 수 있다, 또한 분산 학습을 지원하기 간단하고 사용하기 쉬운 인터페이스를 통해 사용할 수 있도록 개발을 도와 준다.

TF.Learn는 기계 학습 라이브러리 scikit-learn 인터페이스를 텐서플로우에서 사용할 수 있도록 라이브러리를 개발한 것이다. 예시로 3 층 DNN을 보면 다음과 같이 몇 줄로 설명 할 수 있다.

```
classifier = tf . contrib . learn .
DNNClassifier ( feature_columns =
feature_columns , hidden_units = [ 10 ,
20 , 10 ], n_classes = 3 , model_dir =
"/tmp/learn_model" ) # Fit model.
classifier . fit ( x = train_X , y =
train_Y , steps = 2000 )
```

Ⅲ. 결 론

구글에서 인공 신경망 개발과 기계학습을 위해서 오픈소스로 소프트웨어 라이브프러리 를 공개하였다. 텐서플로우는 딥 러닝 라이브러리 중에서는 가장 인기가 있으며 많은 인공지능 논문들이 텐서플로우를 이용해서 연구하여 그 소스 코드를 오픈소스로 공개하는 등 인공지능 연구 개발을 위한 큰 생태계가 완성되고있다.

텐서플로우의 데이터 흐름 표현이 매개 변수 서버 시스템에 대한 기존 작업을 포함하며 사용자가 생산 작업과 새로운 접근 방법을 실험하기 위해 대규모 이기종 시스템을 활용할 수 있는 통일 된 프로그래밍 모델을 제공한다.

텐서플로우를 오픈 소스 소프트웨어로 출시 한 이래로 8,000 명이 넘는 사람들이 소스 코드 저장소를 포크했고, 바이너리 배포판은 50 만 번 다운로드 되었으며 사용자들은 텐서플로우를 사용하는 수십 가지의 기계 학습 모델을 발표했다.

시스템 차원에서 자동 배치, 커널 융합, 메모리 관리 및 스케줄링을위한 알고리즘을 적극적으로 개발하고 있습니다.

인공 지능 연구에서 이를 구현하는 라이브러리는 데이터 흐름 그래프을 이용하여 심층 강화 학습과 같은 알고리즘을 계산 구조가 동적으로 전개되는 경우에도 분산 리소스를 투명하고 효율적으로 사용하는 시스템을 제공한다.

텐서플로우는 오픈 소스로 기술이 공개되어 이를 개선하고, 오류를 수정하는 활동 모두가 연구 커뮤니티와 개발이 협업함으로써 분산 시스템 및 기계 학습에 대한 추가 연구가 계속 진행 중이다. 오픈 소스 소프트웨어 생태계를 통하여 딥러닝 라이브러리의 학습 모델을 포함한 교육 및 기계 학습 자원 개발에 대한 피드백을 받을 수 있는 것도 오픈 소스 기반 인공지능 생태계가 가진 장정이라고 할 수 있다.

참 고 문 헌

[1] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton "ImageNet Classification with Deep Convolutional Neural"Avances in Neural Information Processing Systems 25, NIPS 2012 (https://papers.nips.cc)

[2] TensorFlow: A system for large-scale machine learning (https://arxiv.org/abs/1605.08695)

[3] "Putting Watson to Work: Watson in Healthcare". IBM. Retrieved November 11, 2013.(http://www-03.ibm.com/innovation/us/watson/watson_in_healthcare.shtml)

[4] Hebb, Donald (1949). 《The Organization of Behavior》. New York: Wiley.

[5] Farley, B.G.; W.A. Clark (1954). "Simulation of Self-Organizing Systems by Digital Computer". 《IRE Transactions on Information Theory》 4 (4): 76-84. doi:10.1109/TIT.1954.1057468.

[6] Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain". 《Psychological Review》 65 (6): 386-408. PMID 13602029. doi:10.1037/h0042519

[7] Werbos, P.J. (1975). 《Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences》

[8] http://www.kurzweilai.net/how-bio-inspired-deep-learning-keeps-winning-competitions 2012 Kurzweil AI Interview with Jürgen Schmidhuber on the eight competitions won by his Deep Learning team 2009-2012

[9] Geoffrey E. Hinton (2009), Scholarpedia, 4(5):5947.

[10] A Tour of TensorFlow: https://arxiv.org/abs/1610.01178

[11] ImageNet Classification with Deep Convolutional

Neural Networks (https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf)

[12] Mastering the game of Go with deep neural networks and tree search: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

[13] TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems : https://arxiv.org/abs/1603.04467

약 력

조 만 석

2001년 한양대학교 전기전자컴퓨터공학사
2005년 한양대학교 의용생체공학과 대학원 석사
2004년~2014년 히타치 선임연구원
2014년~2016년 KOSS Lab 소프트웨어 개발자.
2017년 중앙대학교 의료보안연구소 연구위원
2017년~현재 레블업.AI 연구원
관심분야: 의료 데이터 분석, 인공지능 모델 개발