ISSN 2005-8063
2016. 6. 30.
Vol.8 No.2
pp.23-30

말소리와 음성과학

Phonetics and Speech Sciences

한국음성학회지

http://dx.doi.org/10.13064/KSSS.2016.8.2.023

효과적인 2차 최적화 적용을 위한 Minibatch 단위 DNN 훈련 관점에서의 CNN 구현*

Implementation of CNN in the view of mini-batch DNN training for efficient second order optimization

송 화 전** · 정 호 영 · 박 전 규

Song, Hwa Jeon · Jung, Ho Young · Park, Jeon Gue

Abstract

This paper describes some implementation schemes of CNN in view of mini-batch DNN training for efficient second order optimization. This uses same procedure updating parameters of DNN to train parameters of CNN by simply arranging an input image as a sequence of local patches, which is actually equivalent with mini-batch DNN training. Through this conversion, second order optimization providing higher performance can be simply conducted to train the parameters of CNN. In both results of image recognition on MNIST DB and syllable automatic speech recognition, our proposed scheme for CNN implementation shows better performance than one based on DNN.

Keywords: automatic speech recognition, DNN, CNN, second order optimization

1. 서론

심층 신경망(Deep Neural Network; DNN)을 이용한 음향 모델이 거의 모든 음성인식기에 기본으로 사용되는 상황에서 DNN 기반 인식기의 성능을 향상시키기 위해 CNN (Convolutional Neural Network) 및 LSTM-RNN(Long Short-Term Memory-Recurrent Neural Network) 등의 적용이 활발하게 진행되고 있다[1][2].

이 중 CNN은 음성의 경우보다 이미지 분야에서 보다 활발하게 다양한 방법들이 제안되고 있으며 LSTM-RNN의 경우는 반대로 음성/언어 분야에서 다양한 방법이 제안되고 있다. 그러나 CNN의 경우는 특징추출의 관점에서는 LSTM-RNN보다는 좀 더 직관적인 해석을 제공하며 음성신호에 대해서도 CNN을 통해 훈련된 모델 파라미터에 대한 분석이 어느 정도 가능하다. 따라서 본 논문에서도 기 개발된 DNN 기반 음성인식기의 하위층(Layer)에 보다 다양한 특징 추출 능력을 부여하기 위해 CNN 모듈을 개발하여 적용하고자 하였다.

그러나 CNN 계산량은 DNN에 비해 훈련 및 인식 시 모두 상당히 증가된다. 특히 DNN의 경우 GPU를 활용하면 병렬처리가 가능해 대용량 훈련데이터를 사용하는 경우에도 상당한 시간을 단축할 수 있는 반면에, CNN의 경우는 컨벌루션(convolution) 연산으로 인해 GPU를 효과적으로 적용하기가 쉽지 않다.

CNN의 계산을 효과적으로 처리하기 위한 방법은 [3]에서 이

* 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음(R0126-15-1117, 언어학습을 위한 자유발화형 음성대화처리 원천기술 개발)

** 한국전자통신연구원, songhj@etri.re.kr, 교신저자

Received 28 April 2016; Revised 20 June 2016; Accepted 22 June 2016

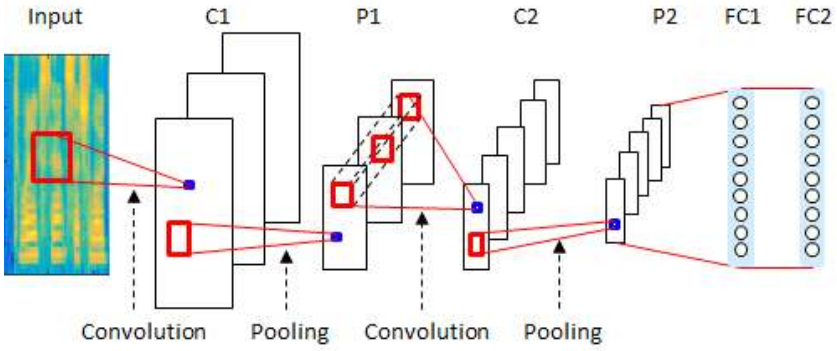

그림 1. Convolutional Neural Network 의 예
Figure 1. An example of Convolutional Neural Network

미 제안되어져 있었고, 현재 CNN 훈련을 위해 가장 많이 사용하고 있는 툴킷(Toolkit)인 Caffe [4]의 경우에도 이러한 방법을 사용하며, 빠른 GPU 계산을 위해 NVDIA사에서 cuDNN[5] 라이브러리를 Caffe 및 다양한 Toolkit과 함께 연계하여 배포하고 있다. 또한 [6]에서 [3] 방식에 대한 좀 더 구체화된 수식을 포함한 논문을 발표하였다. [7]에서는 독자적으로 구현한 CNN을 통해 입력데이터의 특징 변환 사이의 관계에 대한 다양한 실험을 수행하였다.

본 논문에서 독자적으로 구현한 CNN 구현에 대한 기본 방향은 수식적인 측면에서는 [3]의 방식과 동일하지만 해석하고자 하는 측면은 완전히 다른 관점에서 접근한다. 즉, 컨벌루션 과정과 가중치 공유(weight sharing) 특성을 가지는 CNN 의 구조가 일반적으로 minibatch 단위로 훈련을 수행하는 DNN과 동일하며, 이러한 접근법을 통해 기존에 개발한 DNN 학습 및 인식관련 모듈을 수정 없이 CNN을 훈련하고 인식하는데 사용하도록 하였다. 특히 CNN 모델 파라미터를 좀 더 효과적으로 추정하기 위한 2차 최적화 방법(second order optimization)도 minibatch 단위로 DNN 훈련을 위해 이미 개발한 것을 수정 없이 적용하도록 하였다.

이러한 방법이 쉽게 적용 가능한 기본 이유는 3차원 형태의 CNN 구조를 2차원 형태의 DNN 구조로 변경을 수행하는 것이다. 이를 통해 GPU 기반 구현 시 효과적인 행렬의 곱 형태로 CNN 표현이 가능해져 계산시간을 상당히 단축시킬 수 있다. 또한 CNN에서 사용되는 maxpooling 구조도 3차원 형태에서 2차원 maxout 구조로 변경이 가능하여 계산 효율성을 높일 수 있다. 본 논문에서는 CNN과 관련된 수식을 사용해 상세하게 서술하는 대신 그림을 사용하여 CNN과 DNN 사이에 변환 관계를 도식적으로 설명하고자 한다.

요약하자면, CNN은 일반적인 DNN과 동일한 구조로 변경될 수 있으며, DNN 훈련을 위해 구현하였던 SGD(Stochastic Gradient Descent) 기반의 오류 역전파 알고리즘(Error Back-Propagation; EBP)을 수정 없이 그대로 사용하여 CNN 과 관련된 파라미터를 훈련할 수 있어 본 논문에서 새롭게 접근한 해석방법을 통해 CNN을 위해 컨벌루션 연산 및 max-pooling 과정 등의 별다른 개발을 하지 않아도 된다.

본 논문의 구성은 다음과 같다. 2장에서는 DNN 및 CNN의 기본 구조와 CNN을 minibatch 단위로 훈련을 수행하는 DNN 구조로 변경하는 방법에 대해 살펴본다. 이러한 구조변경을 통해 2차 최적화 방법도 간단하게 적용될 수 있음을 3장에서 살펴본다. 4장에서는 구현된 CNN 코드를 검증하기 위해 MNIST DB에 대해 그 성능을 살펴보고, 5장에서 병렬 훈련에 기반한 음성인식 환경에 이를 적용한다. 6장에서 결론과 향후 계획에 대해 서술한다.

2. CNN과 DNN 사이의 관계

2.1. DNN 계산 과정

DNN의 경우 인식과 훈련 모두에서 사용하는 계산식은 식(1)과 같이 행렬과 행렬의 곱 연산과 행렬 원소(element) 각각에 대한 비선형함수의 적용이 거의 대부분이다.

$$Y = f(WX) \tag{1}$$

여기서 W는 DNN의 임의의 레이어(layer) 파라미터 행렬이고 X는 입력행렬이고 Y는 출력행렬이다. 또한 $f()$는 각 행렬의 원소마다 적용되는 비선형함수이다. 입력데이터나 출력데이터가 행렬 형태인 것은 DNN 훈련 및 인식 시 minibatch 단위로 사용하는 것이 보다 효율적이기 때문이다.

행렬 곱 연산은 BLAS (Basic Linear Algebra Subroutines) 라이브러리에 GEMM이라는 함수이름으로 아주 효율적으로 구현이 되어있어 이를 널리 사용한다. 또한 최근 GPU 환경에서도 동일한 기능의 함수를 포함한 라이브러리가 무료로 제공되며 CPU 상에서 동작하는 상용화된 병렬연산 라이브러리보다 훨씬 더 빠른 속도로 행렬 곱을 수행한다. 또한 GPU의 경우는 행렬 원소마다 병렬연산을 수행하도록 쉽게 구현할 수 있으므로 DNN과 관련된 계산을 수행하는데 있어서 최적의 환경을 제공한다.

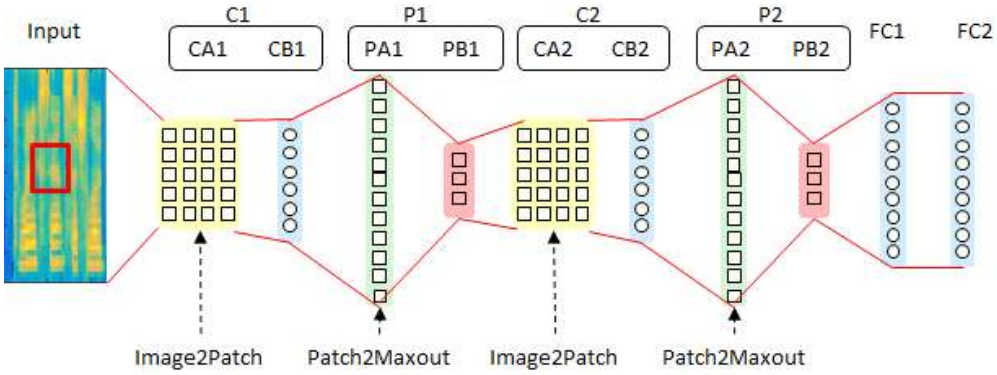

그림 2. Minibatch 훈련단위 DNN 구조의 CNN
Figure 2. CNN in view of minibatch training DNN

2.2. CNN 기본 구조

<그림 1>에 일반적으로 사용되는 CNN의 기본 구조의 한 예를 나타내었다. CNN은 다양한 특징을 추출하는 컨벌루션 층(convolutional layer; CL)과 shift-invariant 특징 추출을 위한 풀링 층(pooling layer; PL)으로 이루어져 있다. 풀링 방법은 maxpooling 방법이 효과적이라 알려져 있으며 본 논문에서도 이를 사용하였다.

<그림 1>에서 C1 층과 C2 층에서 컨벌루션이 수행된다. 보통 N개의 서로 다른 2차원 필터를 이용하여 입력데이터 - 일반적으로 특징 맵(feature map)이라고 한다. - 에 대해 계속 이동하면서 데이터에 대해 가중합을 구하는 것이다. <그림 1>의 C1 층의 경우에 사용한 필터 수는 N = 3이고, C2 층의 경우는 N = 5이다. 2차원 필터 크기는 각각의 입력데이터 특성에 맞게 적절한 크기로 정한다. C2 층의 경우는 3단 구조의 입력 특징 맵과 5단 구조의 출력 특징 맵이 구성된다. C1 층은 1단 입력 특징 맵이고 3단 출력 특징 맵을 사용한다. 다중 입력 특징 맵의 간단한 예는 컬러 이미지이고 RGB 3단 구조이다.

컨벌루션 과정 후 각각의 출력 특징 맵에 대해 풀링 과정을 수행하여 차원을 축소시킨다. 이러한 과정은 shift-invariant 특성을 데이터에 부여하는 것이다. 풀링 과정은 기본적으로 식(1)의 비선형함수처럼 행렬의 원소단위로 처리가 되지 않는다. 이는 GPU를 사용할 때 비효율적이 되지만 reduce라는 기법을 활용하여 블록(block) 단위로 나누어 병렬 계산을 수행하도록 하여 CPU를 사용할 때 보다 훨씬 효과적으로 풀링 값을 계산할 수 있다.

이러한 컨벌루션과 풀링 과정을 반복적으로 수행하여 <그림 1>과 같이 여러 층의 CL과 PL 층을 구성한 후 최종 단에서는 기존의 DNN에서 사용하는 은닉층(FC1)과 출력층(FC2)을 연결하면 전체 CNN 기반 신경망 구조가 구성된다. 여기서 'FC'는 fully connected layer를 의미하며, 일반적으로 DNN의 은닉층(hidden layer)이나 출력층(output layer)과 동일하다.

CNN 파라미터 훈련은 DNN에서 사용되는 EBP 알고리즘을 그대로 사용한다. 그러나 컨벌루션 층에서 EBP 알고리즘을 수행할 때 컨벌루션 과정을 수행해야 하므로 계산적인 측면에서는 식(1)보다 훨씬 비효율적이며 이로 인해 훈련시간이 DNN보다 상당히 길어지게 된다. 따라서 본 논문에서는 이러한 CNN에서 컨벌루션의 비효율성을 극복하기 위해 DNN을 활용하여 보다 쉽게 CNN을 구현하고 이를 더 심도 깊게 활용하고자 하였다.

2.3. CNN 의 minibatch 훈련 단위 DNN 구조로 변경

<그림 1>의 CNN 구조는 <그림 2>와 같이 컨벌루션 및 풀링 층에서 연산과정 분해를 통해 minibatch 훈련 단위를 가지는 DNN 구조로 변경을 할 수 있다.

먼저 컨벌루션 층에서 진행되는 과정은 입력데이터에 대해 현재 시점에서 필터에 대한 컨벌루션 계산을 위해 작은 이미지 패치(patch)로 분할하는 작업(CA1, CA2)과 각각의 패치와 필터 사이의 가중합 연산(CB1, CB2)으로 분할될 수 있다. 여기서 2차원 필터 크기를 $F = m \times n$ 이라고 하자. 이는 패치 크기이기도 하다. 또한 하나의 필터 대신 N개의 다중 필터를 사용하는 것이 일반적인 형식이다.

만약 입력데이터가 T개의 패치로 나누어진다고 가정하자. 이러한 일련의 과정의 한 예를 <그림 3-(a)>에 나타내었다. 즉, 필터 크기는 2×2 ($F = 4$)이고, 입력데이터의 크기가 3×3이므로 컨벌루션을 위해 분할되는 패치의 수는 4 (= T)이다.

만약 입력데이터가 minibatch 단위로 구성된다고 하자. <그림 3-(a)>에서 minibatch 크기는 2이고, 이상의 과정과 같이 각각의 입력데이터에 대해 T개의 패치로 분할하면 된다. 패치를 분할한 후 각각의 패치의 열벡터방향으로 연결을 시켜 하나의 수퍼벡터(supervector)를 구성하도록 한다. (물론 행벡터방향으로 연결을 시켜도 상관은 없다. 본 논문에서는 모든 연결방향을 열벡터방향이라고 가정한다. 이는 GPU상의 행렬곱 연산 함수가 열벡터방향만 지원하기 때문에 이를 고려한 것이다.) 따라서 minibatch 단위의 입력데이터는 최종적으로 행렬 형태로 구성되게 된다. <그림 3-(a)>에서는 M_i 로 표기하였다. 이러한 일련의 작업이 <그림 2>의 Image2Patch 층에서 수행되는 과정이다.

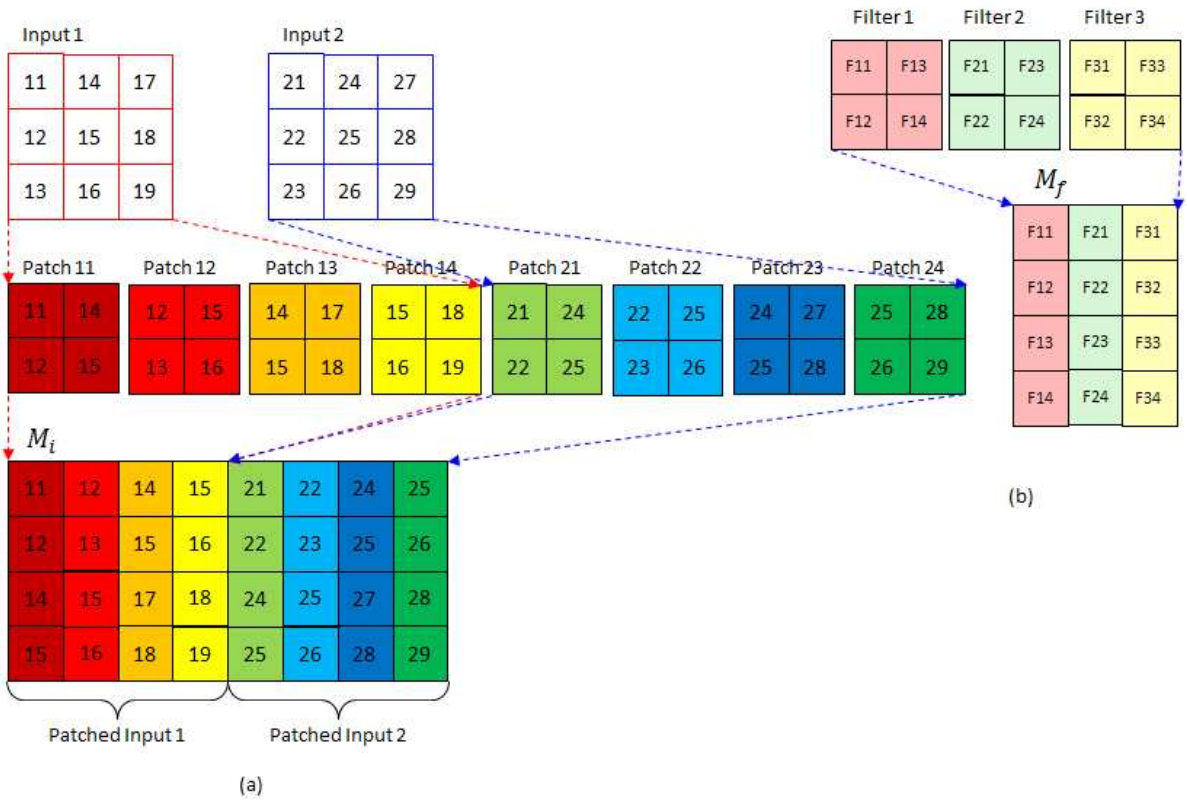

그림 3. Image2Patch 및 filter의 행렬 변환 과정 (a) Image2Patch 변환 과정 (b) filter 의 행렬 변환 과정
Figure 3. Operations of Image2Patch and matrix conversion from set of CNN filters

또한 유사한 작업을 필터에 대해서도 수행을 한다. 즉 <그림 3-(b)>와 같이 N = 3이고 2 × 2인 각각의 필터에 대해서 패치에 대해 수행한 것과 동일하게 열벡터방향으로 연결을 시켜 하나의 수퍼벡터를 구성하여 이를 행렬 M_f 와 같이 구성한다. 따라서 컨벌루션 과정은 $M_f^T M_i$ 의 행렬곱의 형식으로 변환이 되며 식(1)과 같이 DNN 수행과정과 동일하게 된다. <그림 2>의 CB1과 CB2에서 이 과정을 수행한다. 여기서 'T'는 전치행렬을 의미한다.

이상의 일련의 과정은 CNN 입력데이터의 minibatch 크기를 1이라고 하면 이러한 입력데이터는 T개의 minibatch 형식의 패치 크기 차원(F)의 새로운 입력데이터 열로 변환되는 것으로 해석할 수 있다. 즉, CNN 수식 그대로 컨벌루션 연산을 수행하도록 구현하는 관점이 아니고 minibatch 단위로 일반 DNN의 한 은닉층으로 들어오는 입력데이터를 구성하는 관점으로 컨벌루션 층 구성을 완전히 변경시키게 된다.

또한 <그림 3-(a)>와 같이 CNN에 M개의 minibatch 단위로 입력데이터를 처리하게 되면, 실제 컨벌루션 층 입력데이터는 F 차원의 M × T 개의 minibatch 내의 minibatch (Minibatch In Minibatch; MIM) 형태로 데이터 변환이 되는 과정이 발생하게 된다. 또한 F차원을 가진 N개의 필터는 F × N 크기의 은닉층 형태로 변경이 된다. 따라서 컨벌루션 그 자체의 연산이 아니라 입력데이터 형태 변환의 관점에서 식(1)과 같이 행렬곱 형태로 처리할 수 있게 된다.

기존의 논문 [3]-[6] 등에서는 단순히 컨벌루션 연산을 행렬곱으로만 해석하려고 하였고 이를 통해 일반적인 수학라이브러리에서 제공되는 GEMM 관련 함수를 사용할 수 있다는 것만 언급하였다.

그러나 본 논문에서는 MIM의 해석방법을 통해 CNN과 DNN 사이에서의 차이는 단순한 입력데이터 형태의 차이만 있을 뿐 실제 구현하는 측면에서는 완전히 동일하다는 것을 명확하게 서술하였다.

2.4. MIM 관점에서 DNN 구조의 CNN 구조로 변경

2.3절에서 설명한 것과 반대로 DNN을 MIM 관점에서 CNN 구조로 변경을 수행할 수 있다. 즉, 패치 크기가 입력데이터 차원과 동일하고(패치 개수 T=1), 은닉층의 노드 수가 필터 개수(N)가 되는 CNN과 완전히 동일하다. 따라서 CNN과 DNN의 차이는 단지 입력데이터의 형태만 변경되면 완전히 동일한 개념으로 취급될 수 있다.

이상과 같이 DNN에 대한 입력데이터의 해상도를 어떻게 다양하게 변환하여 구성하는가가 DNN의 성능을 향상시킬 수 있는 또 다른 핵심 요소라고 할 수 있으며 이러한 형태의 한가지 예가 CNN이 된다. 또한 최근 이미지 인식분야에서 가장 높은 성능을 보이는 GoogLeNet 등에서 사용하는 다중 해상도를 가지도록 CNN을 구성하는 것도 다양한 해상도를 가지는 입력데이터 구조 변화의 관점으로 해석할 수 있는 또 다른 예가 될 수 있

다.

2.5. 풀링 층에서의 분할 연산 방법

풀링 연산과정도 컨벌루션 연산과정과 같이 분할이 가능하며 이를 통해 기존의 DNN 구조에서 널리 사용되는 최적화된 maxout 연산을 그대로 적용할 수 있게 된다. 간단한 예로 <그림 1>의 P1 층이 <그림 2>에서 PA1층과 PB2층으로 연산 과정을 분할될 수 있다. 여기서 PA1층(Patch2Maxout 층)에서는 CA1층과 같은 행렬 위치 변환 과정이 이루어지며 PB2층에서는 minibatch 단위로 병렬적으로 maxout 연산이 이루어진다.

2.6. 기타 구현 시 고려할 점

CNN 구현의 측면에서는 입력 데이터에 대해 <그림 2>의 CA1 층이나 CA2 층처럼 MIM 과정을 수행하는 층만을 기존의 DNN 구조에 추가 하면 된다. MIM 변환의 경우는 각각의 데이터 어레이 사이의 맵핑 관계를 이용하면 효율적이며, 특히 GPU 환경의 경우는 데이터 위치 이동 자체도 병렬처리로 수행이 가능하다. 예를 들면 <그림 3-(a)>에서 중간의 패치를 구성하는 단계 없이 바로 입력데이터와 M_i 사이의 맵핑 관계만 사용하면 바로 데이터 구성이 병렬 처리로 가능해 진다.

CNN 구현의 측면에서는 입력 데이터에 대해 <그림 2>의 CA1 층이나 CA2 층처럼 MIM 과정을 수행하는 층만을 기존의 DNN 구조에 추가 하면 된다. MIM 변환의 경우는 각각의 데이터 어레이 사이의 맵핑 관계를 이용하면 효율적이며, 특히 GPU 환경의 경우는 데이터 위치 이동 자체도 병렬처리로 수행이 가능하다. 예를 들면 <그림 3-(a)>에서 중간의 패치를 구성하는 단계 없이 바로 입력데이터와 M_i 사이의 맵핑 관계만 사용하면 바로 데이터 구성이 병렬 처리로 가능해 진다.

3. CNN 파라미터에 대한 2차 최적화 추정 방법 적용

DNN 파라미터의 기본 학습 방법은 SGD 기반 EBP 알고리즘을 사용한다. SGD를 사용할 때 최적의 학습률(learning rate)을 추정하는 것이 중요한데 1차 최적화 방법보다는 2차 최적화 방법이 더 좋은 성능 및 훨씬 더 빠른 수렴 속도를 제공한다. 이와 같은 2차 최적화 방법은 CNN 파라미터 훈련에도 적용될 수 있으며, [8]에서 이를 적용하여 빠른 수렴과 1차 최적화 방법보다 높은 성능을 보였다. 그러나 실제 구현을 위해서는 많은 계산량이 필요로 하며 이를 해결하기 위한 다양한 방법이 제안되어져 왔다.

널리 사용되는 2차 최적화 방법으로 Natural Gradient(NG) 에 기반한 방법[9]이 있고 이에 대한 다양한 해석과 응용이 제안되었다. 특히 [10]에서 대용량 음성 데이터를 여러 서버로 분산하여 훈련할 때 보다 효과적인 훈련을 수행하기 위해 minibatch 단위로 근사적인 NG 방법을 적용하였으며, 분산 훈련에서도 상당한 안정적인 성능 향상을 보여 주었다.

본 논문에서도 개발한 CNN에 대해 대용량 데이터 기반 분산 훈련방법을 사용하였으며, MIM 관점을 통해 CNN이 DNN과 동일한 형태로 변환이 되므로 minibatch 단위로 NG 최적화 방법을 적용하여 보다 빠른 수렴과 성능 향상 및 효과적인 CNN 분산 훈련이 가능하도록 하였다. 즉, 본 논문에서 새롭게 해석한 MIM 관점 CNN은 일반 DNN과 동일하기 때문에 minibatch 단위로 DNN에서 사용 가능한 모든 훈련 알고리즘들을 쉽게 적용할 수 있다.

4. 개발한 CNN 방식에 대한 검증

본 논문에서 설명한 MIM 관점 CNN은 기개발한 ESTkDNN Toolkit[11]에 포함되었으며, 기존의 DNN 훈련 및 인식을 위해 구현한 모듈을 그대로 사용하고 단지 <그림 2>에서 설명한 몇몇 데이터 구조 변경 함수만을 추가하여 본 논문에서 수행한 모든 훈련과 평가를 수행하였다.

먼저 본 논문에서 MIM 관점으로 구현한 CNN 코드를 검증하기 위해 공개 평가환경인 MNIST DB를 사용하였다. 이는 CNN의 기본 문헌인 [8]에서 사용한 평가 환경이다. MNIST DB에 대해 제안된 다양한 알고리즘 평가에 대한 비교는 [12]에 정리가 잘 되어 있다.

또한 최근 Google에서 배포된 Tensorflow[13]에서 MNIST DB에 대한 CNN 기본 예제를 제공하고 있어 쉽게 성능을 평가할 수 있다. 본 논문에서도 Tensorflow에서 제공한 CNN 구조와 동일한 형태를 사용하여 MNIST DB에 대한 평가를 수행하였으며, 총 epoch 수는 20번으로 제한하였다.

표 1. 개발한 CNN의 MNIST DB 적용 결과
Table 1. Recognition result of developed CNN on MNIST DB

	Error (%)
MIM 관점 CNN	1.20
+ NG	0.57
World best(no distortion)[12]	0.53
Tensorflow[13]	0.80

<표 1>에 본 논문에서 개발한 CNN에 대한 성능을 나타내었다. NG 방법을 통해 CNN 파라미터를 훈련한 경우는 Tensorflow보다 더 높은 성능을 보였으며, 또한 [12]에 정리된 평가결과 중 입력데이터에 왜곡을 사용하지 않은 경우에 대해서만 비교한다면 거의 최상위권의 순위를 보였다. 평가에 사용된 CNN 환경은 아무런 조정을 거치지 않았고 전체 epoch 횟수도 20번 이하로 제한을 시켰으므로 기존보다 훨씬 효율적으로 훈련을 수행했음을 알 수 있다.

<그림 4>에 MNIST DB에 대해 훈련된 CNN 모델의 첫 번째 컨벌루션 층의 필터값을 그림으로 나타내었다. 입력된 MNIST DB의 패치 데이터에 나타나는 중요한 로컬 특징이 다양한 필터 형상으로 나타나는 것을 알 수가 있다. <그림 5>에는 두 번째 컨벌루션 층의 필터값을 나타내었다. <그림 4>보다는 좀 더 뭉쳐진 형태의 형상이 나타나는 것을 알 수가 있다. 여기서 사용한 모든 필터 크기는 5×5 이다.

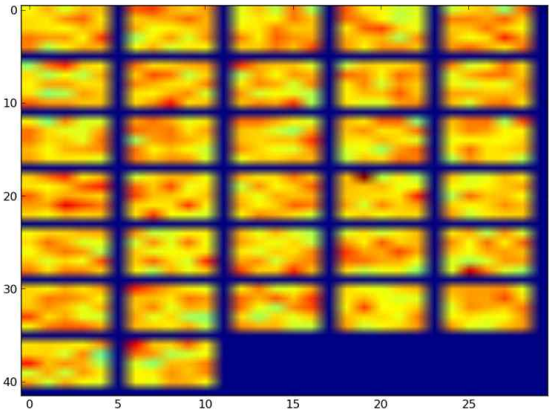

그림 4. MNIST DB에 대한 첫 번째 CL의 필터계수
Figure 4. Filter coefficients of first CL on MNIST DB

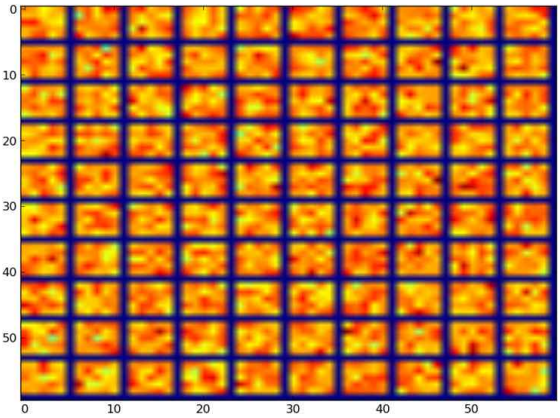

그림 5. MNIST DB에 대한 두 번째 CL의 필터계수
Figure 5. Filter coefficients of second CL on MNIST DB

5. 실험 및 결과

개발된 CNN 코드를 4장에서 검증하였고, 본 장에서는 개발한 CNN을 음성인식에 적용하였다.

인식환경은 한국어 단음절 인식 환경이다. 단음절들 사이의 비슷한 글자가 많아 혼동 가능성이 아주 높은 어려운 인식 환경이다. 그리고 단음절로 구성되므로 언어모델과 관계없이 순수한 음향모델만의 성능을 서로 비교할 수 있다는 점에서 이와 같은 환경을 선택하였다.

음절 DB는 총 120여 시간 정도의 양이며 500명의 화자가 발성한 349,996 발화로 이루어져 있다. 사용한 음절수는 한국어에 나타날 수 있는 음절 중 화자가 실제로 구별해서 발성할 수 있는 것으로 제한하여 총 1756의 단음절로 구성되도록 하였다. DNN-HMM을 훈련하기 위해 먼저 총 1136 개의 공유 상태(tied state)를 가지는 triphone 기반 GMM-HMM을 베이스라인 시스템으로 구성하였다. GMM 훈련에는 39차 MFCC를 사용하였다. 평가를 위해서는 훈련에 참여하지 않은 10명의 화자가 발성한 총 1713개 발성을 사용하였다. 평가에 사용한 1713개 발성은 훈련 시 나타난 1756 음절 중 다시 발성하기 힘든 음절은 제외한 것이고 또한 훈련용 DB와 다른 채널특성을 가지는 환경에서 수집되었다.

구성된 GMM-HMM을 사용하여 강제 정렬을 통해 DNN을 훈련을 위한 상태 레이블(label) 정보를 얻었다. DNN과 CNN을 훈련하기 위해 총 40차의 필터 뱅크(Filterbank) 출력값을 특징으로 사용하였고, 문맥 창(context window) 크기는 15 프레임(frame)을 사용하여 총 600차의 입력데이터를 구성하였다. DNN은 4개의 은닉 층과 하나의 출력 층으로 구성하였고, CNN은 하나의 컨벌루션 및 풀링 층과 총 2개의 은닉층, 1개의 출력층으로 구성하였다. 모든 은닉층의 노드 수는 1024개를 사용하였고 출력 노드 수는 공유 상태수와 동일한 1136개이다. 컨벌루션 층의 필터 개수는 총 300개를 사용하였고, 필터의 크기는 15 × 5 를 사용하였다. 즉, 주파수 축으로만 컨벌루션이 수행되도록 하였다. 또한 풀링 크기는 2 × 1을 사용하였다. DNN과 CNN 파라미터 훈련은 총 10개의 서버를 사용하여 분산 처리를 수행하였으며, 모두 NG 방법을 적용하였다. 사용한 GPU는 GTX 980계열이며 한번 epoch을 수행하는 데 소요된 시간은 DNN의 경우는 대략 13분 정도이고 CNN의 경우는 대략 20분 정도이다. 여기서 음절 평가환경에 대한 총 epoch 횟수도 MNIST DB와 마찬가지로 20번으로 제한하였다. <표 2>에 관련된 실험 결과를 나타내었다.

표 2. 음절 DB에 대한 개발한 CNN 적용 인식 결과
Table 2. Recognition result of developed CNN on Syllable DB

	Accuracy (%)
Baseline (GMM)	42.03
DNN	49.62
MIM 관점 CNN	54.17

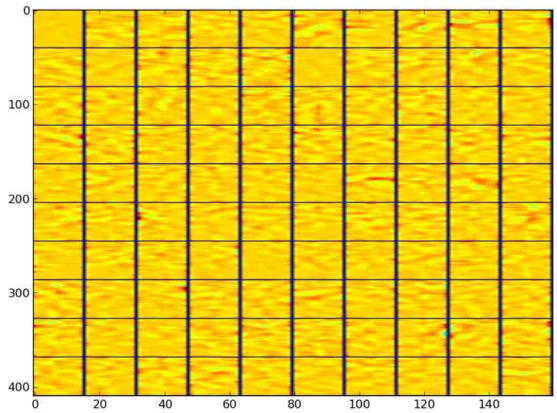

그림 6. 음절 DB에 대한 DNN 모델의 첫 번째 은닉층 파라미터 계수
Figure 6. Parameter coefficients of first hidden layer in DNN on syllable DB

훈련된 DNN 및 CNN의 파라미터 형상을 비교하기 위해 먼저 DNN의 첫 번째 은닉층 파라미터의 일부를 <그림 6>에 나타내었다. 구분된 각각의 크기는 15 프레임의 40차 필터 뱅크 열의 크기(15 × 40)이다. <그림 7>에는 CNN의 첫 번째 컨벌루션 층의

파라미터를 나타내었다. 이것은 15 × 5 크기의 300개 필터의 일부를 나타낸 것이다.

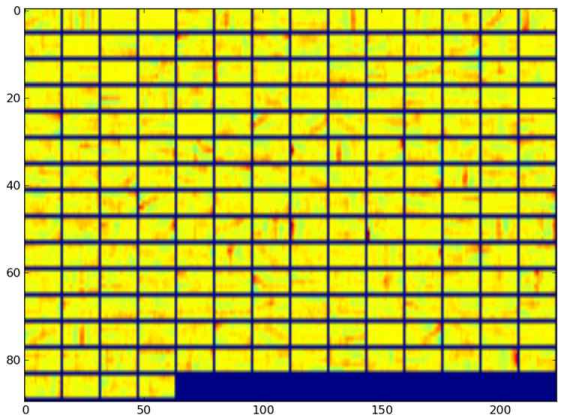

그림 7. 음절 DB에 대한 CNN 모델의 첫 번째 CL의 필터 계수
Figure 7. Filter coefficients of first CL in CNN on syllable DB

<그림 6>과 <그림 7>의 파라미터 패턴이 상당히 다름을 알 수 있다. <그림 6>의 경우는 600차 전체 입력인 15 × 40 이미지 전체의 변화 특성을 보여주고 있으며, <그림 7>의 경우는 필터의 크기가 15 × 5이므로 전체 이미지 크기인 40 × 15에서 패치로 분할된 작은 영역에 대한 중요한 변화 특성을 보여줌을 알 수 있다. 여기서 주의해야 할 부분은 DNN 경우 첫 번째 은닉층의 파라미터 개수는 600 × 1024개이며, CNN의 경우는 15 × 5 × 300개이므로 상대적으로 CNN의 파라미터 수가 적다는 사실에 주의하라.

6. 결론

본 논문은 CNN을 효과적으로 구현하기 위해 MIM 이라는 관점의 새로운 해석 방법을 사용하였으며 이를 통해 DNN과 CNN은 서로 상이한 방식이 아니라 데이터 구조 변경을 통해 서로 동일한 형태로 구성될 수 있음을 보였다. 또한 이러한 해석방식을 통해 DNN 성능을 향상시키기 위한 개발된 다양한 방법과 알고리즘들을 아무런 수정 없이 CNN에 적용 가능하며, 몇몇 예제에서 성공적인 결과를 보여주었다.

그러나 현재 DNN이나 CNN 구조를 아주 다양하게 구성하는데 유연하지 못한 부분이 있어 향후 그래프 구조에 기반을 두어 쉽게 DNN이나 CNN 구조를 변경시킬 수 있도록 개발된 시스템을 확대 적용할 예정이다.

참고문헌

[1] Abdel-Hamid, O., Mohamed, A., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, And Language Processing*, 22(10), 1533-1545.

[2] Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term recurrent neural network architectures for large scale acoustic modeling. *Interspeech 2014* (pp. 338-342).

[3] Chellapilla, K., Puri, S., & Simard, P. (2006). High performance convolutional neural networks for document processing. *Proceedings of International Workshop on Frontiers in Handwriting Recognition.*

[4] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: convolutional architecture for fast feature embedding. *Proceedings of the 22nd ACM International Conference on Multimedia* (pp. 675-678).

[5] Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., & Tran, J. (2014). cuDNN: efficient primitives for deep learning. Retrieved from http://arxiv.org/abs/1410.0759 [Computing Research Repository] on April 15, 2016.

[6] Ren, J. & Xu, L. (2015). On vectorization of deep convolutional neural networks for vision tasks, *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (pp. 1840-1846).

[7] Song, H. J., Jung, H. Y., & Park, J. G. (2015). A study of CNN training based on various filter structures and feature normalization methods. *Proceedings 2015 International Conference on Speech Sciences* (pp. 243-244).

[8] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

[9] Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10, 251-276.

[10] Povey, D., Zhang, X., & Khudanpur, S. (2015). Parallel training of DNNs with natural gradient and parameter averaging. *Proceedings of International Conference on Learning Representations 2015.*

[11] Song, H. J., Jung, H. Y., & Park, J. G. (2015). A study of DNN training based on various pretraining approaches. *Proceedings of the 2015 Spring Conference of the Korean Society of Speech Sciences* (pp. 169-170). (송화전·정호영·박전규 (2015). 다양한 Pretraining 방법에 따른 DNN 훈련 방법에 대한 고찰. 한국음성학회 2015 봄학술대회 논문집, 169-170.)

[12] Rodrigo Benenson. (2013-2016). MNIST. Retrieved from http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html on Apil 15, 2016.

[13] Google. (2015). Tensorflow. Retrieved from https://www.tensorflow.org/ on April 15, 2016.

• **송화전 (Song, Hwa Jeon)** 교신저자
한국전자통신연구원 음성처리연구실
대전광역시 유성구 가정로 218
Tel: 042-860-5836 Fax: 042-860-4889
Email: songhj@etri.re.kr
관심분야: 음성인식, 음향 모델링

• **정호영 (Jung, Ho Young)**
한국전자통신연구원 음성처리연구실
대전광역시 유성구 가정로 218
Tel: 042-860-1328 Fax: 042-860-4889
Email: hjung@etri.re.kr
관심분야: 음성인식

• **박전규 (Park, Jeon Gue)**
한국전자통신연구원 음성처리연구실
대전광역시 유성구 가정로 218
Tel: 042-860-5225 Fax: 042-860-4889
Email: jgp@etri.re.kr
관심분야: 음성인식