# 내장형 음성 인식 시스템을 위한 심층 신경망 최적화 방법

정훈○, 최우용, 박전규
한국전자통신연구원
hchung@etri.re.kr, wychoi4@etri.re.kr, jgp@etri.re.kr

# Deep Neural Network Optimization for Embedded Speech Recognition

Hoon Chung○, Woo-Yong Choi, Jeon-Gue Park
Electronics and Telecommunications Research Institute

**요 약**

본 논문에서는 심층 신경망 기반의 내장형 음성 인식 시스템에서 음성 인식 속도를 개선하기 위한 최적화 방법에 대해 논한다. 심층 신경망 기반의 음성 인식은 기존의 Gaussian Mixture Model (GMM) 기반에 비해 좋은 인식 성능을 보이지만 높은 연산량으로 인해 리소스가 제약된 내장형 단말기에 적용하기에는 어려움이 따른다. 따라서, 본 연구에서는 심층 신경망의 계산량 문제를 해결하고자 ARM 코어에 내장된 병렬 명령어를 사용한 최적화 기법과 특이값 분해를 통해 심층 신경망 매트릭스 연산량 감소 방안에 대해 제안한다.

주제어: 음성인식, 심층신경망, 병렬연산, 특이값 분해

## 1. 서론

최근 들어 심층 신경망 기반의 기계 학습 방식이 다양한 인지 관련 응용 분야에서 좋은 성능을 보이고 있다. 특히, 음성 인식 분야의 경우 심층 신경망 방식은 기존에 사용되던 Gaussian Mixture Model (GMM) 방식에 비해 20% 이상의 오류 감소율을 보이고 있다. 그러나, 일반적으로 심층 신경망 모델은 GMM에 비해 많은 파라메타를 사용해 관측 확률을 모델링하므로 높은 연산량을 필요로 한다. 네크워크 서비스 기반의 음성 인식 시스템은 multi-core 기반의 고성능 서버나 Graphical Processing Unit (GPU) 기반의 고성능 하드웨어를 사용하여 심층 신경망의 높은 연산량 처리하는데 무리가 없지만 개인화 서비스에 중점을 둔 내장형 단말기에서는 심층 신경망의 높은 연산량이 응용 프로그램의 실용화시에 문제를 야기하게 된다.

따라서, 본 연구에서는 이러한 문제를 해결하고자 시스템 자원이 제한된 내장형 단말기에서 고속으로 심층 신경망 계산하기 위한 최적화 방법에 대해 논한다. 2장에서는 관련 연구에 대해 소개하고 3장에서는 본 연구에서 수행한 최적화 방법에 대해 설명한다. 4장에서는 본 연구에서 수행한 최적화 방식의 성능 결과를 기술하고, 5장에서는 결론을 맺는다.

## 2. 관련 연구

심층 신경망의 고속 계산과 관련된 연구는 크게 2가지 연구 방향으로 나뉜다. 첫 번째 방식은 하드웨어 최적 방식으로 multi-core나 many-core 기반의 프로세서에서 병렬 처리를 통해 계산량 분리하는 방식이다 [1][2]. 두 번째 방식은 심층 신경망을 구성하는 모델 파라메타를 성능 저하를 최소화 하면서 줄이는 방식이다. [3][4][5][6]. 하드웨어 최적화 방식은 계산량만을 하드웨어의 병렬 처리 방식으로 구현하게 되므로 성능에는 변화가 없지만 모델 파라메타를 줄이는 방식에서는 인식 성능이 저하가 발생할 수 있다.

## 3. 심층 신경망 계산 최적화 방법

본 절에서는 먼저 일반적인 심층 신경망의 구조에 대해 간단히 설명하고 고속 연산을 위한 NEON 명령어 기반의 병렬 처리 방식과 특이값 분해를 통한 연산량 감소 방법에 대해 설명한다.

### 3.1 심층 신경망 구조

입력 특징 벡터 $x_t$ 에 대해 $L$ 개의 은닉층과 $S$ 개의 출력 노드로 구성된 심층 신경망에 대한 출력 노드별 사후 확률을 구하는 과정은 다음과 같다.

$$z^0 = x_t$$

$$z^{(l+1)} = \tanh(A^{(l)} z^{(l)} + b^{(l)})$$

$$p_s(x_t) = softmax(x_t) = \frac{\exp(a_s z^{(L)})}{\sum_{s'}^{S} \exp(a_s z^{(L)})}$$

이때, $p_s$ 는 s 번째 출력 노드의 사후 확률을 의미하며 $z^{(l)}$ 은 $l$ 번째 층의 입력 벡터를 의미한다. 결국, 심층 신경망은 $A^{(l)} z^{(l)} + b^{(l)}$로 정의된 affine transform, tanh()와 softmax() 연산으로 구성된다.

### 3.2 병렬 연산을 통한 매트릭스 연산 최적화

대부분의 내장형 단말기들은 ARM 코어를 기반으로 한 Application Processing Unit (APU)를 사용하고 있으며 NEON 명령어 기반의 Single Instruction Multiple Data (SIMD)를 지원한다. NEON 기반의 SIMD 명령어에서는 하나의 명령어에 대해 4개의 부동 소수점 연산을 동시에 사용할 수 있어 데이터 병렬화를 통한 연산량 감소 방식을 제공해 준다. 본 연구에서도 심층 신경망의 연산량을 줄이기 위해 NEON 명령어를 사용하여 매트릭스 연산을 최적화 하였다. 그림 1은 NEON 기술을 사용해 4개의 내적을 동시에 구하는 과정을 도식적으로 보여준다.

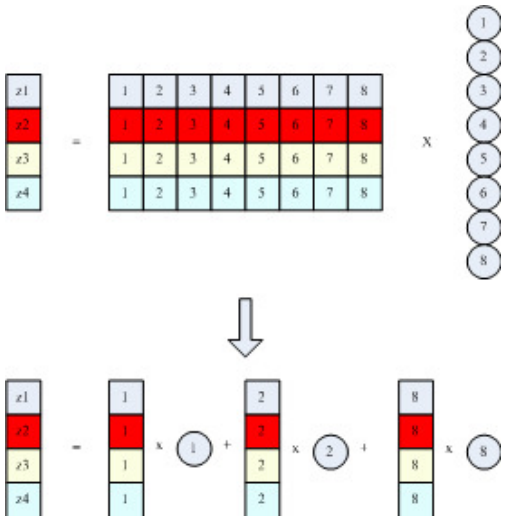

그림 1. NEON을 사용한 4x8 매트릭스 계산 예

본 연구에서는 그림 1의 방식을 확장해 16개의 내적 연산을 동시에 수행하는 코드를 작성하였다. 그 코드의 일부는 다음과 같다.

```
float dot16_neon( const float *a,
          const float *b, unsigned n )
{         float s = 0;
          vmov.f32          q2, #0.0
          ,,,
1:
          subs              %3, %3, #16
          pld [%1, #512*4]
          vld1.32           {d20-d23},          [%1]!
          vmla.f32 q2, q8, q10
          bgt               1b
          vadd.f32          q2, q2, q3
}
```

그림 2. NEON 기반의 매트릭스 연산 일부

아울러, NEON 명령어를 사용한 병렬 연산 구현 시 고려 해야 할 사항은 데이터 액세스시의 병목 현상을 줄이는 일이다. 본 연구에서는 본 연구에서는 ARM 코어에서 제공하는 메모리 pre-cache 기능을 사용하여 향후 사용하게 될 데이터를 미리 준비 할 수 있게 함으로써 연산 수행 속도를 높일 수 있었다.

### 3.3 특이값 분해를 통한 매트릭스 최적화

NEON 명령어 사용을 통한 매트릭스 연산 병렬화로 계산 속도를 개선했지만 저장 공간을 감축하지는 못했다. 또한, 실제 내장형 시스템에서 사용하기에는 반응 속도가 느린 면도 있다. 본 연구에서는 심층 신경망 모델이 차지하는 저장 공간을 줄이고 계산 속도를 좀더 개선하기 위해 특이값 분석을 통한 low-rank approximation 방식을 적용하였다. 심층 신경망 훈련은 많은 휴리스틱 정보에 기반 하여 최적화 되므로 훈련된 매트릭스내에 중복된 정보가 포함될 수 있다. 따라서, 이를 truncated 특이값 분석을 통해 제거함으로써 저장 공간과 계산 속도를 줄이고 경우에 따라서는 추가 성능 개선 효과를 볼 수 있다. 특이값 분해 과정은 다음과 같이 훈련된 MxN 매트릭스를 MxK와 KxN개 2개의 매트릭스로 분해하여 표현한다.

$$A = U\Sigma V^T = \sum_{i=1}^{r} \lambda_i u_i v_i^T$$

$$\widetilde{A} = \widetilde{U} \bullet \widetilde{V}$$
$$where\ \ \widetilde{U} = U_k\sqrt{\Sigma_k}, \widetilde{V} = \sqrt{\Sigma_k}\, V_k^T \ \ (k < r)$$

## 4. 실험 결과

본 연구에서 제안된 최적화 방식을 평가 하기 위해 사용한 심층 신경망 모델, AP의 스펙 및 인식 성능은 표 1과 같다.

표 1. 최적화에 사용된 심층 신경망 모델 및 AP

| 변수 | 설정 값 |
|---|---|
| 특징 | 필터뱅크 |
| 특징 차원 | 600 |
| LDA차원 | 600 |
| 은닉층의 수 | 5 |
| 은닉층의 노드 수 | 1024 |
| 활성화 함수 | tanh() |
| 출력층의 노드 수 | 9792 |
| 출력층의 활성화 함수 | log(softmax()) |
| 사용된 AP | 2.5GHz Snapdragon |
| 인식 평가 도메인 | 차량용 네비게이션 |
| 평가 발화 수 | 2112 |
| 인식률 | 91.21 (%) |

먼저 NEON 기술을 사용한 병렬 연산을 통한 최적화 한 경우 각 모듈별 연산량 감소는 표 2와 같다.

표 2. C코드와 NEON코드 모듈별 연산 시간 (ms)

| | Matrix | Exp() | Tanh() | LogSoftmax |
|---|---|---|---|---|
| 부동소수점 | 20.21 | 9.04 | 15.17 | 25.01 |
| NEON | 6.18 | 5.18 | 9.52 | 7.25 |

표 2에서 보는 바와 같이 NEON 기반의 명령어를 사용할 경우 매트릭스 연산 자체는 대략 3.4배 정도 개선된다. 심층 신경망에서 가장 연산량을 많이 차지하는 부분은 softmax 층의 매트릭스 연산이다. 본 연구에서 사용한 모델의 경우에도 각 은닉층은 1024x1024 매트릭스로 구성되어 있으나 softmax 층은 1024x9792로 5개의 모든 은닉층을 계산한 것보다 많은 계산량을 차기하게 된다. 따라서, 본 연구에서는 softmax 층에 대해 truncated 특이값 분해를 통한 인식 성능 및 모델 사이즈 변화를 측정했다. 표 3은 softmax 층의 랭크수 감소에 따른 인식 성능 및 모델 사이즈 크기 감소를 보여준다.

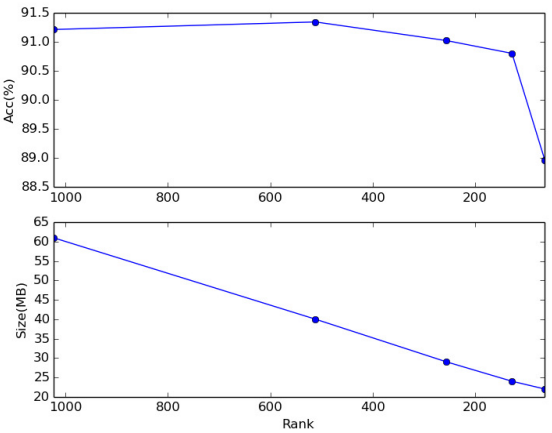


그림 3. softmax 층의 랭크수에 따른 인식 성능 및 모델 사이즈

랭크수는 심층 신경망의 전체 파라메타 수를 결정한다. 랭크수가 높아지면 모델 파라메타의 수가 많아져 메모리 사용량이 증가하나 일반적으로 인식 성능이 개선된다. 하지만, 흥미로운 것은 랭크수가 1024에서 512로 줄어든 경우에는 인식 성능이 오히려 개선되었다. 이는 심층 신경망을 훈련한 모델에 불필요하게 중복된 데이터가 포함되어 있고 이를 제거함으로써 성능이 개선되었다고 해석 될 수 있을 것 같다.

## 5. 결론

본 논문에서는 저장 공간과 연산 능력이 제한된 내장형 단말기에서 심층 신경망 기반의 음성 인식 속도를 개선하기 위한 최적화 방법에 대해 논하였다. 본 논문에서는 병렬 처리 연산을 통한 연산 방식 측면에서의 최적화와 truncated 특이값 분해를 통해 심층 신경망 모델 측면의 최적화에 대해 구현하였다. truncated 특이값 분해의 경우에는 훈련된 모델내 파라메타 중복 정도에 따라 모델 사이즈 감속 및 추가 인식 성능의 개선도 가능하였다. 향후 연구로써는 심층 신경망의 특성을 좀더 고려한 모델 파라메타 감축 방안에 대한 연구가 필요하다.

## 감사의 글

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [ R0126-15-1117, 언어학습을 위한 자유발화형 음성대화처리 원천기술 개발]

## 참고문헌

[1] K.-S. Oh, and K. Jung, "GPU implementation of neural networks," Pattern Recognition, vol. 37, issue 6, pp. 1311-1314,June 2004.

[2] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in proceedings of NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.

[3] J. Xue, J. Li, and Y. Gong, “Restructuring of deep neural network acoustic models with singular value decomposition,” Proc. Interspeech, pp. 2365-2369, 2013.

[4] Ba, Jimmy and Caruana, Rich. Do deep nets really need to be deep? In NIPS, pp. 2654?2662, 2014.

[5] Bucilu, Cristian, Caruana, Rich, and Niculescu-Mizil, Alexandru. Model compression. In KDD, 2006.

[6] Denil, Misha, Shakibi, Babak, Dinh, Laurent, de Freitas, Nando, et al. Predicting parameters in deep learning. In NIPS, 2013