공학석사학위논문

레지듀얼 심플 게이티트 콘보넷을 이용한 온-디바이스 음성인식

On-device Speech Recognition with Residual Simple Gated Convolutional Networks

2019년 8월

서울대학교 대학원

전기 정보 공학부

이 윤 진

공학석사학위논문

레지듀얼 심플 게이티트 콘보넷을 이용한 온-디바이스 음성인식

On-device Speech Recognition with Residual Simple Gated Convolutional Networks

2019년 8월

서울대학교 대학원

전기 정보 공학부

이 윤 진

레지듀얼 심플 게이티트 콘보넷을 이용한 온-디바이스 음성인식

On-device Speech Recognition with Residual Simple Gated Convolutional Networks

지도교수 성 원 용

이 논문을 공학석사 학위논문으로 제출함

2019년 8월

서울대학교 대학원

전기 정보 공학부

이 윤 진

이윤진의 공학석사 학위 논문을 인준함

2019년 8월

위 원 장: 최기영

부위원장: 성원용

위 원: 조남익

요 약

생활의 편의를 위한 자동 음성인식 시스템은 늘 우리의 곁에 있는 스마트폰과 더불어 실생활에 쓰이는 임베디드 장치에 필수적으로 장착된 지 오래다. 특히, 인공신경망 기반의 알고리즘은 높은 성능 덕분에 자동 음성인식 시스템에 많이 적용된다. 하지만 대부분의 자동 음성인식은 각종 회사의 서버로 보내져 처리된다. 이는 사생활 침해, 보안의 문제와 높은 레이턴시를 야기하여 사용자의 장치에서 독립적으로 처리되는 자동 음성인식 시스템의 필요성이 높아지고 있다. 장치에서 직접 인공신경망 기반의 음성인식을 처리하기 위해서는 장치의 제한된 배터리 용량 때문에 전력 소비를 최소화해야 할 필요가 있다.

지금까지 많은 인공신경망 모델이 개발되었다. 이들 중, 연속적인 문맥을 볼 수 있도록 하는 회귀적 특성 덕분에 회귀신경망(RNN) 기반의 알고리즘이 음성인식을 위해 많이 사용된다. 특히 LSTM 기반의 회귀신경망(RNN)이 유명하다. 하지만, LSTM RNN에 사용되는 모든 파라미터를 저장하기에는 임베디드 장치가 가진 캐쉬 메모리의 크기가 충분하지 않다. 그래서 많은 DRAM 접근이 일어나는데, 이는 이 신경망이 계산되는 속도를 늦춘다. 또한, 현재 계산에 과거에 계산된 값이 되먹임(feedback)되는 복잡한 회귀신경망의 특성으로 인하여 매 시간 스텝에 대한 병렬화 계산이 불가하다. 이 또한 많은 DRAM 접근을 야기하여 LSTM RNN이 계산되는 속도증가를 저지하고, 많은 전력을 소모하게 한다.

이 논문에서는 한 번에 캐쉬 메모리에 저장될 수 있는 1M 정도의 작은 파라미터 크기를 가진 레지듀얼 심플 게이티드 콘보넷(Residual Simple Gated Convolutional Network)을 실제 임베디드 장치에서 실행한 결과를 보여준다. 이 모델이 가지는 1차원 depthwise 콘볼루션은 음성 신호의 일시적인 패턴을 찾는 데 도움이 된다. 또한, 회귀신경망 대신 콘볼루션 신경망을 사용함에 따라, 시간 스텝 별 병렬화가 가능해져, 배터리 사용량이 감소하고, 이 인공신경망 모델의 연산 속도가 높아진다.

임베디드 장치에서 음성인식 속도를 높이기 위하여 기존의 심플 게이티드 콘보넷에 인셉션 레지듀얼 연결을 추가하여 정확도 저하 없이 인공신경망의 층수를 줄이려 하였으나 이는 속도 향상에는 큰 성과가 없는 것으로 드러났다.

keywords: 음성인식, 시퀀스 모델링, 회귀신경망 (RNN), 콘볼루션 신경망 (CNN), 임베디드 디바이스

student number: 2016-26378

차 례

표 차례

그림 차례

제 1 장 서론

1.1 온-디바이스 음성인식 : 이점과 문제점

근래에 들어 인공신경망을 바탕으로 한 자동 음성인식이 많이 사용되고 있다. 특히, 인공신경망을 이용한 앤드투앤드(end-to-end) 음성인식이 많은 음성인식 방법 중에서도 효과적으로 이용된다. [1, 2, 3, 4]. 앤드투앤드 음성인식에서는 인공신경망을 훈련하는 데이터(data)로 라벨(label)과 피쳐(feature) 간의 일대일 매핑(mapping)이 필요하지 않아 훈련을 위한 데이터를 이전보다 쉽게 만들 수 있도록 한다.

음성인식을 이용한 서비스들이 우리 생활에 서서히 스며들고 있는 현재에 온-디바이스 음성인식에 대한 필요성이 나날이 늘어가고 있다. 온-디바이스 음성인식은 사용자의 음성이 서버로 전해지지 않고 사용자의 디바이스 안에서 수행되는 자동 음성인식을 일컫는다. 음성인식이 임베디드 장치(embedded device)나 스마트폰에서 많이 사용되는 반면, 거의 모든 음성인식은 사용자의 음성이 서버로 전달되면 서버에서 이를 수행하게 된다. 하지만 이렇게 서버에서 진행되는 음성인식은 사용자들에게 보안과 사생활 침해에 대한 걱정을 불러일으킨다.

따라서, 온-디바이스 음성인식은 이러한 보안과 사생활 침해 걱정에 대한 중요한 해결책이다. 사용자의 음성이 서버로 전달되고 저장될 염려가 없기 때문에 사용자는 그들의 사생활이 침해될 걱정을 덜 수 있다. 또한, 거의 모든 음성인식이 개개인의 기기안에서 수행되기 때문에 데이터를 전송하고 전달받는데 드는 대기시간을 없앨 수 있다. 서비스 제공 업체들 또한 서비스 제공을 위한 서버 사용이 줄기 때문에 막대한 에너지와 이에 대한 소비 비용을 절감할 수 있다.

하지만, 디바이스에서 자체적으로 음성인식을 수행하는 데에는 몇 가지 해결해야 할 문제점이 존재한다. 첫 번째로, 온-디바이스 음성인식은 디바이스의 하드웨어적인 사양에 큰 영향을 받는다. 이전에 사용되어 온 인공신경망은 음성인식을 수행하는데 수백만 메가바이트(MB) 크기의 파라미터가 필요하다. 하지만 하드웨

어적인 사양이 다소 낮은 사용자의 디바이스 안에서 이를 수행하려면 파라미터의 크기를 대폭 줄여야 할 필요가 있다.

두 번째로, 메인 메모리(main memory)로부터 파라미터를 불러오는 빈도수가 낮아져야 한다. 인공신경망으로 음성인식을 수행하는 경우 파라미터의 크기가 굉장히 크기 때문에 캐쉬 메모리(cache memory)에 모두 저장될 수 없어 메인 메모리에 저장된다. 하지만, 메인 메모리에 자주 접근하여 정보를 가져오게 되면 인공신경망을 계산하는 시간이 오래 걸리게 된다. 이뿐만 아니라 기기에서 소비하는 에너지 또한 높아져 온-디바이스 음성인식에 좋지 않다.

서버에서 수행되는 음성인식은 배치(batch) 방식을 사용하여 여러 사용자의 음성을 받아 이를 동시에 계산할 수 있어 메인 메모리에 접근을 줄일 수 있다. 하지만 사용자 각각의 디바이스에서 음성인식을 수행하는 경우, 여러 사용자의 음성을 받아 추론하는 것이 아니기 때문에 이 방법을 적용할 수 없다. 이는 회귀적 인공신경망이 아닌 컨볼루션 인공신경망을 이용한 인공신경망 모델을 사용하여 여러 시간 스텝의 계산을 동시에 처리하여 해결할 수 있다.

이 논문에서는 실제 디바이스에서 처리되는 파라미터 수가 적고, 메인 메모리의 접근을 줄여 추론 속도를 높인 인공신경망을 보여준다. 이에 대해 설명하기 전에, 이에 사용된 모델과 앤드투앤드 음성인식에 대해 먼저 설명하고자 한다.

1.2 음성인식의 구성요소

앤드투앤드 음성인식은 크게 어쿠스틱 모델(acoustic model)과 랭귀지 모델(language model)로 이루어져 있다. 어쿠스틱 모델은 주어진 라벨에 대한 어쿠스틱 피쳐(acoustic feature)의 확률적 분포를 나타낸다. 랭귀지 모델은 단어나 문자의 확률적 분포를 나타낸다. 음성인식은 어쿠스틱 피쳐와 표현되는 철자를 올바르게 연결하는 방법을 찾는 과정이라고 볼 수 있다. 특히 어쿠스틱 모델은 어쿠스틱 피쳐를 인간이 읽을 수 있는 단어나 철자로 바꿔주는 음성인식의 중요한 부분이다. 인공신경망으로 음성인식을 수행하는 경우, 이 어쿠스틱 모델은 무조건 훈련된 인공신경망을 사용한다.

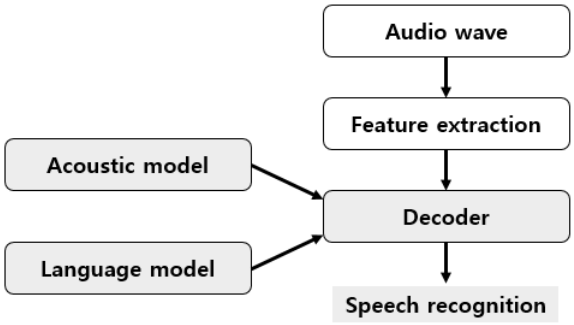

그림 1.1: 음성인식의 전체적 절차

그림 1.1은 음성인식의 전체적 절차를 보여준다.

우리는 어쿠스틱 모델을 위해 온-디바이스에 적용하기 쉽게 고안된 심플 게이티드 콘보넷 [5]을 적용한다. 이 모델은 온-디바이스 음성인식에 적합한 파라미터 크기를 가져 메인 메모리에서 필요한 파라미터를 가져오는데 걸리는 속도를 줄였고, 콘볼루션 연산을 어쿠스틱 모델에 사용하여 여러 시간 스텝을 동시에 계산하도록 해 추론 속도 또한 개선하였다. 또한, 이전 모델들보다 좋은 성능을 보인다.

1.3 회귀신경망을 사용한 어쿠스틱 모델의 단점

회귀신경망은 어쿠스틱 모델을 훈련하는데 오랜 기간 사용된 모델이다. 이는 회귀적 성질 때문에 연속적으로 관련된 정보를 처리하는 데에 있어 훌륭한 모델로 간주 되어 왔다. 여러 회귀신경망 구조가 고안되었지만, 그중 LSTM(long short-term memory)이 시퀀스 모델링(sequence modeling)을 하는 데 가장 많이 이용된다 [6]. 그 이유는 모델에 추가된 스테이트(state)와 게이트(gate)로 인하여 그레디언트(gradient)가 폭발하거나 사라지는 것을 방지할 수 있기 때문이다 [7].

$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f), \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i), \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o), \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c\right), \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t).
\end{aligned}
\tag{1.1}
$$

하지만 LSTM 회귀신경망은 여러 시간 스텝(time step)을 동시에 계산하는 데 적합하지 않다. LSTM 모델의 식은 다음 1.1과 같다. 이전의 히든 스테이트(hidden state) 값이 현재 계산되는 스테이트에 이용되기 때문에, 여러 시간 스텝의 스테이트를 함께 계산하는 것은 불가능하다. 이 경우, 매 시간 스텝을 계산할 때마다 메인 메모리에서 훈련된 파라미터를 불러와야 하기 때문에, 추론 속도가 느려짐은 물론이고, 에너지 소비까지 늘어난다.

1.4 관련 연구

1.4.1 diagonal LSTM RNN

diagonal LSTM RNN에서는 모든 회귀적 성질을 지닌 행렬이 기존의 LSTM RNN보다 작다. 즉, 행렬 $\mathbf{U}_f$, $\mathbf{U}_i$, $\mathbf{U}_o$ and $\mathbf{U}_c$의 대각선의 값을 제외하고는 모두 0이 된다. 즉, 회귀적 파라미터 행렬과 입력 행렬의 행렬 곱이 이 둘의 element-wise 곱이 된다. 식 1.2은 diagonal LSTM RNN에 대한 식이다. $\odot$은 행렬의 element-wise 곱을 의미한다.

$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \odot \mathbf{h}_{t-1} + \mathbf{b}_f), \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \odot \mathbf{h}_{t-1} + \mathbf{b}_i), \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \odot \mathbf{h}_{t-1} + \mathbf{b}_o), \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \odot \mathbf{h}_{t-1} + \mathbf{b}_c\right), \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t).
\end{aligned}
\tag{1.2}
$$

LSTMRNN에서 회귀하는 부분의 파라미터가 굉장히 큰 것이 여러 시간 스텝을 동시에 계산하는데 큰 걸림돌이다. 하지만 diagonal LSTM RNN의 경우 회귀하는 부분에 사용되는 가중치 행렬에서 얻은 대각선의 값들만 사용하므로 그 크기가 매우 작아진다. 이를 임베디드 장치에 사용하면 회귀하는 부분의 가중치를 캐쉬 메모리에 저장할 수 있기 때문에 메인 메모리에서 가중치를 불러오지 않아도 되어 여러 시간 스텝을 병렬화하여 계산하는 데 유리하다. 따라서 여러 시간 스텝의 입력을 한 번에 병렬화하여 불러 올 수 있다.

1.4.2 QRNN

QRNN에는 RNN과는 달리 컨볼루션 연산을 사용한다 [8, 9]. LSTM RNN은 오직 한 시간 스텝 입력 $\mathbf{x}_t$ 만 받아 연산하는 반면, QRNN은 k개의 $\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-k+1}$ 입력이 연속된 정보를 계산하기 위해 입력된다. QRNN에 회귀적 가중치 행렬이 없기 때문에 여러 시간 스텝을 병렬화하여 계산하는 것이 가능하다. 하지만 k 크기에 비례하여 파라미터의 크기가 커지기 때문에 k값을 잘 조절하는 것이 중요하다. QRNN의 식은 아래와 같다 1.3.

$$
\begin{aligned}
\hat{\mathbf{x}}_t &= \tanh\left(\mathbf{W}_z \mathbf{x}_t + \mathbf{b}_z\right), \\
[\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t] &= \sigma([\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o]\mathbf{x}_t + [\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o]), \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{x}}_t, \\
\mathbf{h}_t &= \mathbf{o}_t \odot \mathbf{c}_t + (1 - \mathbf{o}_t) \odot \mathbf{x}_t.
\end{aligned}
\tag{1.3}
$$

1.4.3 게이티드 콘보넷

게이티드 콘보넷 또한 회귀 신경망을 사용하는 대신 컨볼루션 신경망을 이용하여 연속적인 데이터를 모델링 할 수 있도록 하였다. 한 층의 게이티드 콘보넷은 두 콘볼루션 연산으로 이루어져 있는데, 이 콘볼루션의 필터들(filter)이 특정 길이의 문맥을 볼 수 있도록 하는 역할을 한다[10]. 게이티드 콘보넷은 회귀적 연결을 사용하는 구조가 없기 때문에 여러 시간 스텝을 쉽게 병렬화할 수 있다. 게이티드 콘보넷의 식은 아래 1.4와 같다.

$$
\mathbf{h}(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + \mathbf{b}) \odot \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c}) \tag{1.4}
$$

여기서 $*$은 콘볼루션 연산을 의미하고 $\odot$는 element-wise 곱을 의미한다. $\mathbf{W}, \mathbf{V} \in \mathbb{R}^{T \times D \times D}$ 과$\mathbf{b}, \mathbf{c} \in \mathbb{R}^{D}$은 훈련가능한 변수를 의미하며, 여기서 D는 이 네트워크의 채널 너비이다. 입력 벡터는 $\mathbf{X} \in \mathbb{R}^{N \times D}$이며 여기서 N은 시퀀스 길이를 의미한다.

1.5 논문의 개요

본 논문의 뒷 내용은 아래와 같다. 제2장에서는 심플 게이티드 콘보넷을 온-디바이스에서 처리하기 위해 파라미터의 개수를 1M개로 설정하여 훈련한 여러 결과를

보여준다. 제3장에서는 실시간 음성인식을 위하여 레이턴시를 낮춘 심플 게이티드 콘보넷의 각 레이턴시 마다 성능을 측정하였다. 또, 파라미터 양자화 방법과 이를 심플 게이티드 콘보넷 모델에 적용한 결과를 보여준다. 제4장에서는 본 논문을 끝맺는다.

제 2 장 심플 게이티드 콘보넷

2.1 게이티드 콘보넷

게이티드 콘보넷은 시퀀스 모델링을 위해 게이트 구조를 가진 컨볼루션 층을 쌓아 만들어진다[10]. 게이티드 콘보넷의 기본 구조는 아래 식2.1 과 같다.

$$\mathbf{h}^l(\mathbf{X}^l) = (\mathbf{X}^l * \mathbf{W}^l + \mathbf{b}^l) \odot \sigma(\mathbf{X}^l * \mathbf{V}^l + \mathbf{c}^l) \tag{2.1}$$

$\odot$은 element-wise 곱셈을, σ는 시그모이드 함수를, $*$는 컨볼루션 연산을 의미한다. 게이티드 콘보넷 네트워크는 L개의 층으로 $l = 0, 1, 2, 3, ..., L-1$ 과 같이 이루어져 있다. $\mathbf{W}^l, \mathbf{V}^l \in \mathbb{R}^{T \times 1 \times D \times D}$과 $\mathbf{b}^l, \mathbf{c}^l \in \mathbb{R}^D$ 은 훈련되는 파라미터이다. 여기서 T는 필터의 길이, D는 피쳐(feature)의 크기를 의미한다.

$\mathbf{W}^l$와 $\mathbf{V}^l$는 $[T, 1, D, D]$ 형태이다. 모델의 크기는 T의 길이에 비례하며, D는 모델 구상 시 정해지는 값이다. T는 각 층에서 보는 문맥의 길이라고 볼 수 있다. time-delay neural networks(TDNN) [11, 12]에서 한정된 범위의 문맥의 시퀀스를 배우는 것에 대한 연구가 진행된 바 있다. 게이티드 콘보넷에서 특정 길이의 시퀀스를 처리하는 것은 TDNN에서 일시적인 문맥을 보는 것이라고 생각할 수 있다.

음성인식 시 인접한 시퀀스 정보를 봐야만 하는 것은 불가피한 일이기 때문에 게이티드 콘보넷은 특정 길이 T를 설정하여 모델 구조를 만든다. [13]에서는 T의 범위를 13에서 28까지 정하여 실험하였지만 T에 비례하여 파라미터의 크기가 기하급수적으로 증가하므로 온-디바이스에서의 적용을 위해 T를 11정도로 설정한다,

2.2 심플 게이티드 콘보넷

심플 게이티드 콘보넷에서는 T의 길이를 1로 설정한다. 이는 각 컨볼루션 필터마다 차지하는 가중치 $\mathbf{W}^l$와 $\mathbf{V}^l$을 TDD에서 DD로 줄이는 효과가 있다. 한 층마다 $\mathbf{W}^l$와 $\mathbf{V}^l$, 두 개의 가중치를 가지고 있기 때문에 $2T$배 만큼 파라미터의 크기가 감소한다. 하지만 T를 1로 설정하면 게이티드 콘보넷은 연속된 문맥을 보지 못하게 된다. 이는 성능이 감소하는 결과로 이어지게 되어 이를 방지하기 위하여 네트워크가 인접한 정보를 볼 수 있게 하는 다른 방법이 필요하다.

이것은 1차원 depthwise 컨볼루션을 더해 해결할 수 있다. 1차원 depthwise 컨볼루션은 적은 파라미터를 사용하며 특정 길이의 시퀀스를 볼 수 있게 해준다. Depthwise 컨볼루션은 이미 비젼, 인공 번역, 음성인식에 사용되고 있다[14, 15, 16, 17]. 1차원 depthwise 컨볼루션의 수식은 아래와 같다.

$$\mathbf{h}^l_{t,1,d} = \sum_{i=\lfloor -T/2 \rfloor}^{\lfloor T/2 \rfloor} \mathbf{F}^l_{i,1,d} \mathbf{X}^l_{t+i-1,1,d} \tag{2.2}$$

식 (2.2)는 t번째 시간 스텝에서의 컨볼루션 식을 보여주고 있다. 1차원 depthwise 컨볼루션 필터의 행렬은 $\mathbf{F}^l \in \mathbf{R}^{T \times 1 \times D}$이다. 이 컨볼루션 필터 $\mathbf{F}^l$의 파라미터 크기는 TD이다. 이 컨볼루션을 기존의 게이티드 컨볼루션 모델의 앞에 배치함으로써 $2TDD$ 크기의 파라미터가 $2DD + TD$로 줄어드는 효과를 볼 수 있다.

또한 심플 게이티드 컨볼루션은 1차원 depthwise 컨볼루션과 동시에 이웃한 피쳐 차원의 방향의 정보를 얻는 방식으로 성능향상을 꾀하였다. 이웃한 피쳐를 같이 봄으로써 서로의 상관관계를 계산할 수 있다. 이 너비를 K로 설정한 1차원 depth-

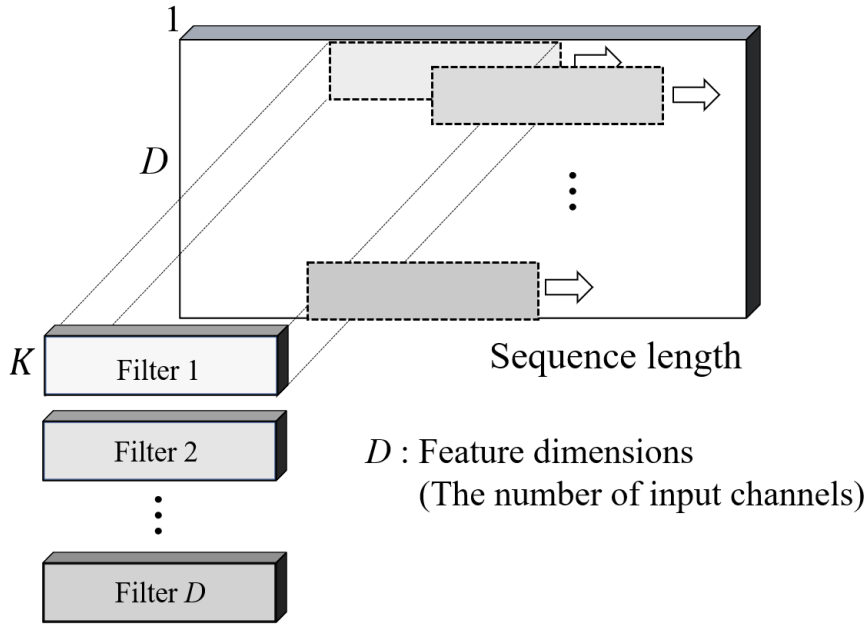

그림 2.1: 너비가 K인 1차원 depthwise 콘볼루션

wise 컨볼루션의 수식은 아래와 같다.

$$\mathbf{h}^{l}_{t,1,d} = \sum_{w=\lfloor -K/2 \rfloor}^{\lfloor K/2 \rfloor} \sum_{i=\lfloor -T/2 \rfloor}^{\lfloor T/2 \rfloor} \mathbf{F'}^{l}_{i,1,d,w} \mathbf{X}^{l}_{t+i-1,1,d+w} \tag{2.3}$$

$\mathbf{F'}^{l}$의 파라미터 크기는 1차원 depthwise 콘볼루션의 파라미터 크기 TD의K배인 TDK이다. 따라서, 한 층의 심플 게이티드 콘보넷에 필요한 파라미터의 크기는 $2DD + TDK$이다. D의 크기가 수백이기 때문에 크기 TDK는 게이티드 콘보넷 파라미터의 크기인 TDD에 비하여 훨씬 작다. 그림 2.1는 너비 K를 가지는 1차원 depthwise 콘보넷을 보여준다.

심플 게이티드 콘보넷은 게이티드 콘보넷과는 다르게 batch normalization [18]

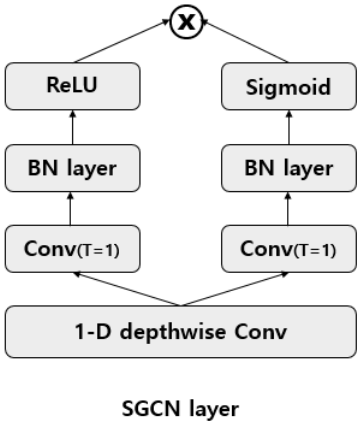

그림 2.2: 심플 게이티드 콘보넷 한 층의 구조

층이 추가된다. 이전의 게이티드 콘보넷에는 batch normalization이 거의 이용되지 않았다[10, 13, 19]. Batch normalization은 ResNet [20] 구조와 같이 컨볼루션 층과 activation 층 사이에 들어간다. 게이티드 콘보넷의 $\mathbf{X}^l * \mathbf{W}^l + \mathbf{b}^l$ 식 뒤에는 activation 층이 들어가지 않는다. 심플 게이티드 컨보넷은 $\mathbf{X}^l * \mathbf{W}^l + \mathbf{b}^l$ 뒤에 ReLU 함수 층을 추가한다. 다른 컨볼루션 $\mathbf{X}^l * \mathbf{V}^l + \mathbf{c}^l$ 에는 이미 게이트 역할을 하는 activation 함수가 있어 다른 activation 층이 필요없다. 그림 (2.2)는 심플 게이티드 콘보넷 한 층의 구조를, 그림 (2.3)는 심플 게이티드 콘보넷 모델 전체의 구조를 보여준다.

2.3 레지듀얼 심플 게이티드 콘보넷

컨볼루션 신경망의 성능은 신경망의 깊이와 각 컨볼루션 층의 너비와 비례한다고 볼 수 있다. 실제로 심플 게이티드 콘보넷 또한 깊이 쌓을 수 있는 모델의 특성이 성능향상에 큰 도움이 되었다. 하지만 깊은 층에 의하여 온-디바이스에서 음성인식을 수행하는 데 있어 추론 속도를 높이는 데 한계가 있다. 따라서, 정확도를 낮추지 않고 층수를 줄이기 위하여 ResNet[20]과 인셉션(inception) 연결[21]에서 착안한 레

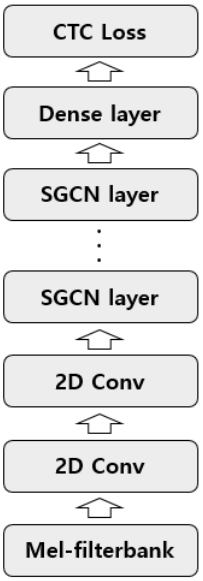

그림 2.3: 심플 게이티드 콘보넷 모델의 구조

지듀얼 shortcut 연결(그림 2.4)과 한 층의 컨볼루션과 1차원 depthwise 컨볼루션을 사용한 인셉션 레지듀얼 연결(그림 2.5)을 사용하였다.

2.4 실험결과

아래 표들은 여러 심플 게이티드 컨볼루션 구조와 다른 모델의 성능을 비교한 결과이다. 성능 측정의 척도로는 CER(character error rate)과 WER(word error rate)을 사용하였다. 모델의 구조는 $\mathbf{n}_{layer} \times \mathbf{n}_{dim}$으로 표현된다. 여기서 $\mathbf{n}_{layer}$ 은 모델의 층수를 의미하며 $\mathbf{n}_{dim}$은 LSTM에서는 hidden dimension의 크기를, 심플 게이티드 콘보넷에서는 피쳐맵의 너비를 의미한다.

표 2.1은 모델의 파라미터 개수가 1M개 이하인 모델들이다. LSTM에는 1차원 필터의 너비 K를 1로 두고, 1차원 depthwise 컨볼루션을 적용하였다. 파라미터 개수를 1M개 이하로 설정 시 심플 게이티드 콘보넷이 1차원 컨볼루션을 사용한 LSTM

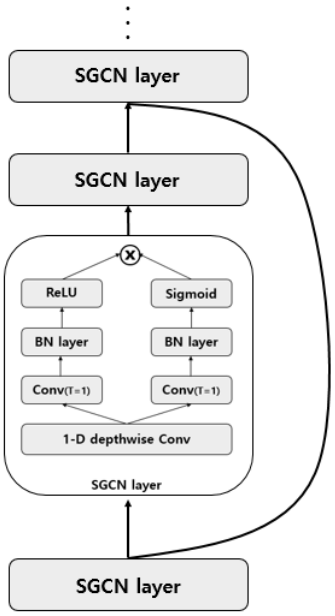

그림 2.4: 숏컷 레지듀얼 심플 게이티드 콘보넷 모델의 구조

보다 성능이 뛰어난 것을 알 수 있다.

표 2.2는 약 1M개의 파라미터를 가진 심플 게이티드 콘보넷과 인셉션 레지듀얼 심플 게이티드 콘보넷 구조들을 다양한 층수와 너비로 WSJ Si-ALL을 이용해 훈련한 결과이며, 표 2.3은 숏컷 레지듀얼 심플 게이티드 콘보넷과 심플 게이티드 콘보넷을 WSJ Si-284로 훈련한 결과이다. 실험 결과, 인셉션 레지듀얼 심플 게이티드 콘보넷이 2 3개 낮은 층을 사용하여 비슷하거나 나은 성능을 보이기는 하였으나, 모델의 너비 또한 살짝 넓어 인셉션 레지듀얼 연결만의 효과라고 보기 어렵다. 또한 인셉션 레지듀얼 연결을 음성인식 수행 시 연산하여야 하기 때문에 속도 향상에 도움을 주지 못했다.

표 2.4는 다양한 너비 K를 가진 심플 게이티드 콘보넷의 성능이다. 해당 모델들은 WSJ Si-284를 이용해 훈련하였다. [5]에서도 성능이 K에 정비례하여 높아지지 않고, 너비 K가 21 정도로 커진 경우에는 성능이 좋아지는 것을 볼 수 있었던 것과

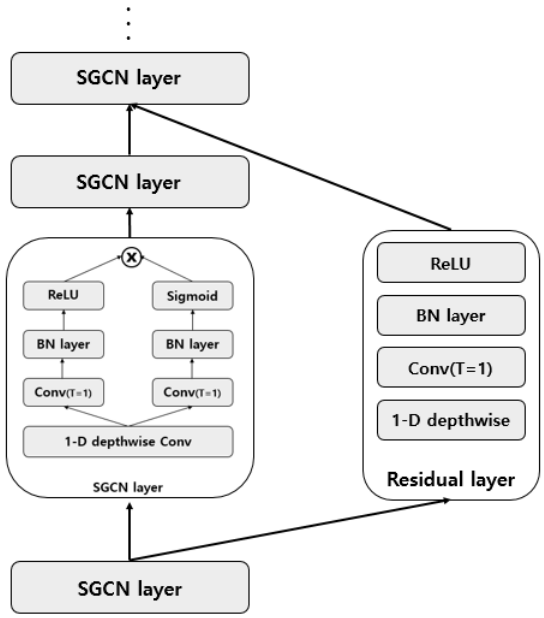

그림 2.5: 인셉션 레지듀얼 심플 게이티드 콘보넷 모델의 구조

같이 K를 1에서 11까지 설정한 너비 190의 12층 짜리 심플 게이티드 콘보넷에서는 K와 성능의 연관성을 볼 수 없다. K가 커짐에 따라 파라미터의 개수 또한 증가하기 때문에 온-디바이스 음성인식을 위한 구조에서는 K를 5로 설정하고 훈련하였다.

표 2.5는 190차원의 10층 짜리 인셉션 심플 게이티드 콘보넷의 인셉션 구조의 필터 크기를 달리하여 실험한 결과이다. 해당 모델들은 WSJ Si-284를 이용해 훈련하였다. 기본 구조에서 쓰이는 $[11, 5, 190, 1]$ 크기의 같은 필터 크기를 사용한 구조가 가장 좋은 결과를 얻었고, 이를 제외하고는 인셉션 구조가 현재 시간 만을 보도록 훈련한 $[1, 5, 190, 1]$ 크기의 필터를 이용한 모델이 좋은 성능을 보였다.

표 2.1: 1M개의 파라미터를 가진 신경망들의 WSJ eval92 CER과 WER

(a) WSJ Si-284로 훈련한 CER과 WER

		Greedy	
모델	**파라미터 개수**	**CER**	**WER**
4x160 LSTM	0.96M	10.78	37.1
4x160 LSTM, 1-D depthwise	0.97M	9.64	33.6
12x190 SGCN ($K = 5$)	0.99M	**7.32**	**26.51**

(b) WSJ Si-284 보다 큰 WSJ Si-ALL로 훈련한 CER과 WER

		Greedy	
모델	**파라미터 개수**	**CER**	**WER**
4x160 LSTM	0.96M	8.64	31.2
4x160 LSTM, 1-D depthwise	0.97M	6.72	24.1
12x190 SGCN ($K = 5$)	0.99M	**5.53**	**20.20**

표 2.2: 여러 심플 게이티드 콘보넷 구조의 WSJ eval92 CER과 WER

		Greedy	
모델	**파라미터 개수**	**CER**	**WER**
15x150 SGCN	0.80M	5.99	21.79
12x190 SGCN	0.99M	5.53	20.20
9x210 inception res SGCN	1.01M	5.33	20.20
10x190 inception res SGCN	0.92M	5.60	20.68
10x210 inception res SGCN	1.11M	**5.17**	**19.49**

표 2.3: 숏컷 레지듀얼 심플 게이티드 콘보넷 구조의 WSJ eval92 CER과 WER

		Greedy	
모델	**파라미터 개수**	**CER**	**WER**
12x190 SGCN	0.99M	7.32	26.51
12x190 shortcut res SGCN	0.99M	7.72	27.64

표 2.4: 다양한 너비 K를 가진 1차원 depthwise 컨볼루션에 대한 WSJ eval92 CER과 WER

		Greedy	
모델	**파라미터 개수**	**CER**	**WER**
12x190 SGCN ($K = 1$)	0.89M	7.27	26.21
12x190 SGCN ($K = 3$)	0.94M	7.76	28.19
12x190 SGCN ($K = 5$)	0.99M	7.32	26.51
12x190 SGCN ($K = 7$)	1.04M	7.67	28.39
12x190 SGCN ($K = 9$)	1.09M	7.77	28.73
12x190 SGCN ($K = 11$)	1.14M	7.71	28.00

표 2.5: 다양한 필터 크기를 가진 10x190 인셉션 심플 게이티드 콘보넷 모델에 대한 WSJ eval92 CER과 WER

		Greedy	
필터 크기	**파라미터 개수**	**CER**	**WER**
$[1, 5, 190, 1]$	0.901M	7.89	28.74
$[11, 1, 190, 1]$	0.903M	7.93	29.20
$[5, 5, 190, 1]$	0.908M	8.10	29.29
$[11, 3, 190, 1]$	0.911M	7.97	28.89
$[11, 5, 190, 1]$	0.920M	7.74	27.72

제 3 장 온-디바이스 심플 게이티드 콘보넷

3.1 낮은 레이턴시 심플 게이티드 콘보넷

3.1.1 음성 전처리

우리는 connectionist temporal classification (CTC) [2, 22] 알고리즘을 이용한 어쿠스틱 모델을 훈련하기 위해 심플 게이티드 콘보넷을 채택하였고, The Wall Street Journal (WSJ) Corpus를 사용하여 이를 훈련하였다. 이 corpus를 사용하려면 음성의 특징을 추출하기 위한 적합한 전처리 과정이 필요하다.

훈련을 위한 음성 추출 처리 과정은 다음과 같다. 우리는 첫 입력으로 WSJ corpus의 가공되지 않은 wav 파일을 받는다. 음성 특징을 추출하기 위해서는 음성에 불연속 퓨리에 변환(DFT)을 취해주어야 한다. 이는 25ms의 시간 동안 Hamming window를 씌워 10ms 마다 실행된다. 이 불연속 퓨리어 변환 과정을 거친 추출된 샘플들은 필터뱅크(filter-banks)를 통과하여 40차원의 log Mel-주파수 필터뱅크 특성을 지니게 된다.

이렇게 추출된 음성의 특징들은 컨볼루션 신경망을 거치며 연속된 데이터 길이가 짧아지게 된다. [4]에 따르면 컨볼루션 신경망을 거쳐 특징들의 수를 줄이는 것이 훈련 속도와 오류 비율의 관점에서 성능향상에 기여 한다는 결과가 있다. 우리는 이 특성을 적용해 40차원의 log Mel-주파수 필터뱅크 특징을 두 층의 컨볼루션 신경망의 입력으로 넣었다. 연속된 데이티의 길이는 이 두 층의 컨볼루션 층을 거치며 줄어들게 된다.

3.1.2 신경망에서의 레이턴시

컨볼루션 신경망에 입력으로 넣어져 처리되는 데이터는 단 한 순간만의 데이터가 아니라 그 전과 후의 음성과 함께 넣어져 처리된다. 하지만 이와 같은 특성은 이 구조가

추론에 사용될 시, 현재의 값을 계산하기 위하여 후에 들어올 데이터를 기다렸다가 처리하도록 한다. 이는 레이턴시(latency)를 키워 음성인식의 응답속도를 늦춘다.

이는 과거에 입력된 음성 특징만을 보고 연산하는 causal 컨볼루션 [23, 24] 을 이용하여 낮은 레이턴시를 가지도록 할 수 있다. causal 컨볼루션은 컨볼루션 연산을 수행할 때 미래 시간 스텝에서 온 음성 특징을 참고하지 않기 때문에 레이턴시를 발생시키지 않는다. 본 논문에서 채택한 구조는 1차원 depthwise 컨볼루션의 필터의 크기 T가 11이다. 이는 이 필터가 T-5에서 T+5까지의 입력을 받아 처리한다는 의미인데, 여기서 T-5는 과거의 5 스텝만큼의 입력을, T+5는 미래의 5 스텝만큼의 입력을 뜻한다. 따라서, 심플 게이티드 콘보넷의 한 causal 하지 않은 층은 5 시간 스텝만큼의 레이턴시를 발생시킨다.

한 시간 스텝이 약 20ms이기 때문에 한 층의 심플 게이티드 콘보넷이 100ms의 레이턴시를 발생시킨다. 이는 우리가 causal 하지 않은 층을 한 층씩 쌓을 때마다 레이턴시 또한 늘어난다는 것을 의미한다. 레이턴시가 늘어나면 실시간으로 음성을 인식하는 것을 상당히 방해한다. 우리는 이 causal 하지 않은 층을 causal 한 층으로 대체함으로써 레이턴시를 낮추었다.

3.2 심플 게이티드 컨보넷의 양자화

3.2.1 신경망 파라미터 최소화

근래에 들어 인공신경망이 음성인식과 이미지 분야에서 많은 알고리즘을 대체하며 훌륭한 성과를 내고 있는 것은 사실이다. 하지만 인공신경망은 이에 사용되는 파라미터의 개수가 매우 많아 많은 저장 공간을 필요로 한다. 이는 네트워크의 파라미터 개수를 줄이거나, 네트워크에 필요한 파라미터 각각의 크기를 줄여 해결할 수 있다.

네트워크의 파라미터 개수룰 줄이는 방법은 knowledge transfer learning [25] 이나 low-rank decomposition [26] 과 같은 방법을 이용하여 더 작은 네트워크로 훈련을 시키는 방법과 pruning[27]을 사용하여 불필요한 파라미터를 제거하는 방법이 있다. 하지만 온-디바이스 음성인식에 사용하는 심플 게이티드 콘보넷은 이미 비슷한 성

능을 내는 다른 네트워크 대비 많은 파라미터 개수를 줄인 네트워크이다.

따라서 심플 게이티드 콘보넷에는 네트워크에 필요한 파라미터 각각의 정확도를 고정소수점으로 바꾸어 파라미터 자체의 크기를 줄였다. 파라미터 자체의 크기를 줄이면 이를 저장하는 저장 공간이 줄어드는 것뿐만 아니라 이를 메인 메모리가 아닌 캐쉬 메모리에만 저장하고 있기에도 유리하여 메모리에서 파라미터를 불러오는 시간을 줄일 수 있다. 또한, 한정된 하드웨어에서 음성인식 추론을 위한 연산을 할 때 부동소수점이라면 한 값을 연산할 수 있는 하드웨어로 32비트에서 8비트의 고정소수점을 만들 시 네 개의 값을 한번에 연산할 수 있게 되어, 실시간 온-디바이스 음성인식을 하는데 많은 도움이 된다.

3.2.2 직접적 양자화 방법

파라미터 양자화를 수행하려면 파라미터의 범위와 양자화에 대한 민감도를 확인하여야 한다 [28]. 여러 층을 쌓아 만든 신경망에서는 이를 층별로 묶어 좀 더 간편하게 양자화한다. 각 층의 파라미터 중 바이어스는 값의 범위가 넓어 더 높은 정확도를 사용해 양자화한다. 양자화에 대한 민감도는 특정 층만 양자화해 실험해 보며 측정할 수 있다 [28]. 일정한 간격의 양자화는 그림 3.1을 참고한다.

각 파라미터에 쓰일 양자화 비트 수를 결정하면, 그 모델의 결과의 오류를 줄이는 방향으로 양자화를 진행한다. 직접적인 양자화에서 양자화로 인해 생긴 결과의 오류와 파라미터 간의 직접적 영향을 찾기가 쉽지 않아 최적의 스텝 크기 δ를 정하는 것은 굉장히 어렵다. 따라서 최적의 스텝 크기는 처음에 L2 에러 최소화 방법 3.1을 사용하여 정한 후 실험을 통하여 세밀하게 튜닝된다. 가장 좋은 결과는 보통 스텝 크기가 Lloyd-Max 알고리즘에서 정한 값을 1.2-1.6배의 값일 때이다[29]. 직접적 파라미터 양자화 방법은 아래와 같다.

1) 부동소수점으로 훈련된 파라미터를 준비한다.

2) 입력과 각 층의 파라미터를 모두 양자화한다.

3) 첫 스텝 크기는 L2 오류 최소화 방법을 사용해 정한다. 입력 데이터와 첫 층의 파라미터부터 여러 양자화한 값으로 출력을 계산하여 오류를 줄이는 방향의 값

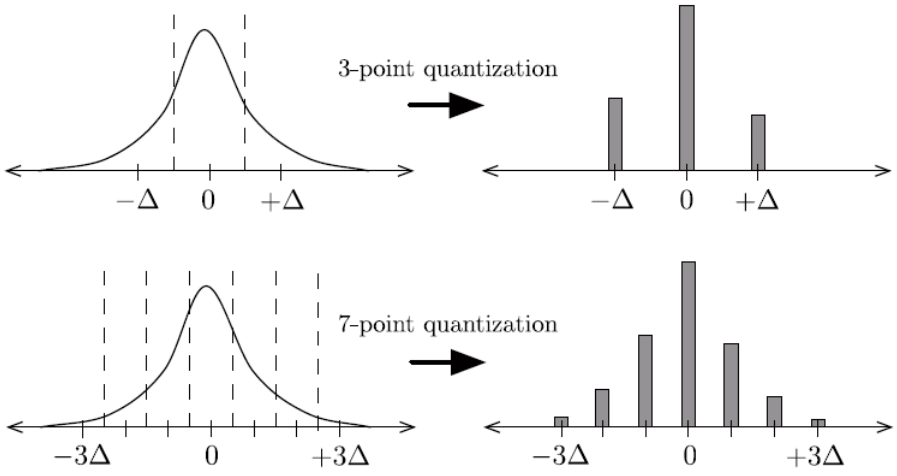

그림 3.1: 파라미터의 직접적 양자화 전과 후의 히스토그램

을 그 층의 파라미터의 스텝 크기로 정한다.

4) 세 번째 단계를 각 층별로 수행하여 마지막 층까지 수행한다.

$$
\begin{aligned}
g(t) &= sign(x) \cdot min\{\lfloor \frac{\mid x \mid}{\Delta} + 0.5 \rfloor, \frac{M-1}{2}\} \\
\Delta^{new} &= \frac{\sum_{i=1}^{N} x_i g(x_i)}{\sum_{i=1}^{N} g(x_i)^2} \\
E &= \frac{1}{2}\sum_{i=1}^{N} (\Delta \cdot g(\mathbf{x}_i) - x_i)^2
\end{aligned}
\tag{3.1}
$$

3.2.3 재훈련을 통한 양자화

3-2에서 설명한 방법으로 양자화를 하는 것은 특히 파라미터의 간격을 세 구간이나 일곱 구간 정도로 나눠 양자화했을 때 출력의 정확도가 많이 떨어진다. 이것은 양자화된 신경망의 오차역전파법(error backpropagation)을 적용해 다시 훈련 시킴으로써 성능을 향상할 수 있다.

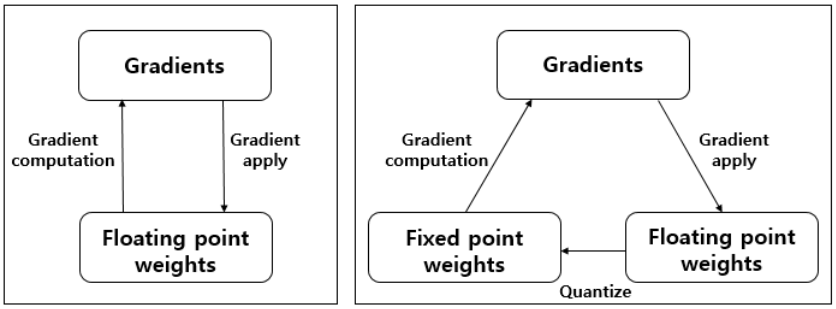

그림 3.2: 일반적인 파라미터 훈련 방법과, 양자화를 위한 파라미터 재훈련 방법

오차역전파법은 오류 신호 σ 를 출력부터 입력까지 역으로 전파하며 오류 증감률을 계산하는 출력의 오류를 줄이기 위한 기울기 하강 방법이라고 할 수 있다. 하지만 매순환 마다 변화하는 파라미터의 크기가 양자화 스텝 크기보다 작기 때문에 이 방법을 바로 양자화된 파라미터에 적용하는 것은 좋지 않다. 그래서 우리는 재훈련 시 부동소수점과 고정소수점을 모두 사용한다. 재훈련을 통한 파라미터 양자화 방법은 아래와 같다.

1) 부동소수점으로 훈련된 파라미터를 양자화한다.

2) 양자화된 파라미터를 이용해 loss를 구한다.

3) 그 loss를 기반으로 부동소수점의 증감률을 구한다.

4) 그 증감률을 부동소수점 파라미터에 더해 값을 갱신한다.

5) 위 네 가지 순서를 충분한 정확도를 얻을 때까지 반복한다.

3.2.4 Tensorflow lite를 이용한 양자화

최근 tensorflow가 모바일 하드웨어와 같은 사양이 낮은 디바이스에서도 인공신경망을 쉽고 빠르게 구현할 수 있도록 지원하는 tensorflow lite를 제공하기 시작했다. 온-디바이스에서 심플 게이티드 컨보넷을 실시간 구현하기 위하여 우리는 tensorflow에서 제공하는 8비트 양자화를 사용한다[30].

Tensorflow lite의 경우 최대값과 최소값을 이용한 균등한 양자화를 한다. 양자화

된 파라미터의 식은 다음 (3.2)과 같다.

$$
\begin{aligned}
Q(x) &= \frac{(x - mean)}{scale} + 128 \\
scale &= \frac{max(x) - min(x)}{255}
\end{aligned}
\tag{3.2}
$$

여기서 mean은 가중치나 activation 값의 평균을 의미하며, activation의 평균값이나, 최대최소값은 훈련 과정에서 분포를 통해 얻어진다. 이를 통해 파라미터들은 최소값이 0, 최대값이 255에 매핑(mapping)되는 비대칭의 균일한 양자화가 된다.

Batch normalization의 경우 scale factor와 분산을 가중치의 값에 합쳐서 양자화를 실행함으로 연산량을 줄이고, 양자화 노이즈에 더 내성이 강해지도록 하였다. 아래 식 은 가중치에 batch normalization의 파라미터가 합쳐진 수식이다.

$$
\mathbf{w_{fold}} := \frac{\gamma w}{\sqrt{EMA(\sigma_{\mathbf{B}}^{\mathbf{2}}) + \varepsilon}}
\tag{3.3}
$$

여기서 γ는 batch normalization의 scale factor를, $EMA(\sigma_B^2)$는 batch의 컨볼루션의 분산에 대한 이동 평균 값이다. ε는 수치적 안정성을 위해 사용되는 작은 상수 값이다.

3.3 실험결과

온-디바이스 음성인식을 실시간으로 처리하기 위해서는 낮은 레이턴시를 가지는 모델이 필요하다. 표 (3.1)는 심플 게이티드 콘보넷과 인셉션 레지듀얼 심플 게이티드 콘보넷 모델을 낮은 레이턴시를 가지도록 WSJ Si-284를 사용해 훈련한 결과이다. 200ms와 300ms의 레이턴시를 가지는 모델의 훈련 결과로는 100ms의 레이턴시 차이가 성능 차이와 비례한다고 보기는 힘들지만, WSJ Si-284를 사용해 훈련한

표 3.1: 레이턴시의 길이에 따른 심플 게이티드 콘보넷 구조의 WSJ eval92 CER과 WER

			Greedy	
모델	**레이턴시**	**파라미터 개수**	**CER**	**WER**
12x190 SGCN	200ms	0.99M	7.41	27.13
12x190 SGCN	300ms	0.99M	8.18	29.81
12x190 SGCN	1200ms	0.99M	7.32	26.51
10x190 res SGCN	200ms	0.92M	8.50	31.90
10x190 res SGCN	300ms	0.92M	8.24	29.61
10x190 res SGCN	1200ms	0.92M	7.74	27.72
12x170 res SGCN	200ms	1.08M	7.50	28.25
12x170 res SGCN	300ms	1.08M	8.48	31.21
12x170 res SGCN	1200ms	1.08M	6.77	25.32

레이턴시가 1200ms인 12x190 SGCN 모델의 WER이 26.51인 것과 비교하여 낮은 레이턴시를 가지는 모델이 성능이 낮다는 것을 알 수 있다.

표(3.2)는 WSJ Si-ALL로 훈련한 심플 게이티드 콘보넷과 인셉션 레지듀얼 심플 게이티드 콘보넷 모델의 가중치를 3-4에서 설명된 tensorflow lite 방식으로 양자화하여 8비트로 성능을 측정한 이용한 결과이다. WER은 모든 모델에서 32비트로 성능을 측정한 결과보다 약 1퍼센트 정도 증가하였다. 10x210 res SGCN 모델을 900MHz quad-core ARM Cortex-A7 CPU에서 32비트와 8비트로 음성인식을 수행한 결과 realtime factor가 각각 0.503, 0.226으로 실제 2배 이상의 속도향상이 있었다.

표 (3.3은 음성인식을 수행할 때 전체 음성 데이터를 받지 않고 순간의 음성 데이터를 받아 음성인식을 시작할 수 있도록 적은 프레임을 입력으로 넣어 900MHz quad-core ARM Cortex-A7 CPU에서 8비트의 고정소수점으로 실행 속도를 측정한 결과이다. 속도 측정에 사용한 데이터는 길이 총 2532초의 WSJ eval92이다. 음성

표 3.2: 여러 심플 게이티드 콘보넷 구조의 8비트를 사용한 WSJ eval92 CER과 WER

		Greedy			
모델	**파라미터 개수**	32비트		**8비트**	
		CER	**WER**	**CER**	**WER**
15x150 SGCN	0.80M	5.99	21.79	6.31	22.72
12x190 SGCN	0.99M	5.53	20.20	5.71	20.73
9x210 res SGCN	1.01M	5.33	20.20	5.74	21.53
10x190 res SGCN	0.92M	5.60	20.68	5.97	21.05
10x210 res SGCN	1.11M	5.17	19.49	5.45	20.36

데이터를 네 프레임 씩 입력으로 넣었을 때 실시간 0.5배의 실행 속도를 보인다.

표 3.3: 심플 게이티드 콘보넷에 적은 프레임 입력을 넣어 측정한 실시간 대비 속도

	4 프레임		8 프레임		16 프레임		전체 프레임	
모델	실행시간	xRT	실행시간	xRT	실행시간	xRT	실행시간	xRT
15x150 SGCN	1135.21	0.448	818.55	0.323	659.56	0.260	516.09	0.204
12x190 SGCN	1157.94	0.457	843.74	0.333	683.49	0.270	535.48	0.211
9x210 res SGCN	1168.14	0.461	847.89	0.335	680.14	0.269	535.35	0.211
10x190 res SGCN	1115.68	0.440	814.12	0.322	655.95	0.259	520.74	0.206
10x210 res SGCN	1271.23	0.502	924.03	0.365	737.57	0.291	572.41	0.226

제 4 장 결론

본 논문에서는 온-디바이스에서 실시간으로 음성인식을 실행하기 위한 방법들을 제안하였다. 같은 1M개의 파라미터 개수를 사용하였을 때 컨볼루션 층을 쌓아 만든 심플 게이티드 콘보넷이 기존 음성인식에서 좋은 성능을 내는 LSTM 모델보다 나은 성능을 보이는 것을 확인했다. 또한 심플 게이티드 콘보넷을 이용하면 시간 스텝을 병렬화하여 계산할 수 있어 실행 속도를 높일 수 있다.

또한, 숏컷 레지듀얼 연결과 인셉션 레지듀얼 연결을 통해 더 높은 속도를 내고자 하였다. 인셉션 레지듀얼 연결을 사용한 심플 게이티드 콘보넷이 2 3층을 적게 쌓은 모델에서 더 나은 성능을 보이기는 하였지만, 모델의 너비가 기존 모델보다 조금 더 넓고 음성인식 수행 시 연산해야 할 레지듀얼 연결 때문에 더 높은 속도를 내지는 못하였다. 인셉션 레지듀얼 구조의 필터 크기와 심플 게이티드 콘보넷 구조를 조절하면 속도와 성능 등 원하는 조건에 맞는 결과를 찾을 수 있으리라 판단한다.

32비트의 부동소수점을 8비트의 고정소수점으로 양자화해 음성인식을 수행한 결과 WER는 1퍼센트 정도의 증가가 있었지만, 수행속도가 2배 이상 증가하여 실시간 온-디바이스 음성인식에 효율적이라 볼 수 있다. 8비트 고정소수점으로 양자화할 때 재훈련을 사용하여 8비트 고정소수점으로 변환한다면 고정소수점 사용으로 인한 성능 하락을 방지하는데 도움을 줄 것으로 사료된다.

약 20퍼센트의 WER을 기준으로 삼고 음성 데이터를 네 프레임 씩 입력하였을 때, 추론시간이 실제 음성 시간의 0.5배 이하인 것을 보인다. 이는 greedy 디코딩이 아닌 적절한 랭귀지 모델 디코딩을 사용하면 실시간으로 더 높은 성능의 음성인식이 가능할 것이다.

참고 문헌

[1] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[2] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *International Conference on Machine Learning*, 2014, pp. 1764–1772.

[3] Yajie Miao, Mohammad Gowayyed, and Florian Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 167–174.

[4] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International Conference on Machine Learning*, 2016, pp. 173–182.

[5] Lukas Lee, Jinhwan Park, and Wonyong Sung, "Simple gated convnet for small footprint acoustic modeling," *submitted to INTERSPEECH*, 2019.

[6] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2013, pp. 1310–1318.

[8] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher, "Quasi-Recurrent Neural Networks," *International Conference on Learning Representations (ICLR 2017)*, 2017.

[9] David Balduzzi and Muhammad Ghifary, "Strongly-typed recurrent neural networks," in *International Conference on Machine Learning*, 2016, pp. 1292–1300.

[10] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier, "Language modeling with gated convolutional networks," in *International Conference on Machine Learning (ICML)*, 2017, pp. 933–941.

[11] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[12] Vijayaditya Peddinti, Yiming Wang, Daniel Povey, and Sanjeev Khudanpur, "Low latency acoustic modeling using temporal convolution and LSTMs," *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 373–377, 2018.

[13] Vitaliy Liptchinsky, Gabriel Synnaeve, and Ronan Collobert, "Letter-based speech recognition with gated ConvNets," *arXiv preprint arXiv:1712.09444*, 2017.

[14] Francois Chollet, "Xception: Deep learning with depthwise separable convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 1800–1807.

[15] Lukasz Kaiser, Aidan N Gomez, and Francois Chollet, "Depthwise separable convolutions for neural machine translation," *arXiv preprint arXiv:1706.03059*, 2017.

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[17] Jinhwan Park, Yoonho Boo, Iksoo Choi, Sungho Shin, and Wonyong Sung, "Fully neural network based speech recognition on mobile and embedded devices," in *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[18] Sergey Ioffe and Christian Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*. JMLR, 2015, pp. 448–456.

[19] Jian Kang, Wei-Qiang Zhang, and Jia Liu, "Gated convolutional networks based hybrid acoustic models for low resource speech recognition," in *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*. IEEE, 2017, pp. 157–164.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 770–778.

[21] et al. Szegedy, Christian, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[22] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *International Conference on Machine Learning (ICML)*. ACM, 2006, pp. 369–376.

[23] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio.," in *SSW*, 2016, p. 125.

[24] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al., "Conditional image generation with PixelCNN decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.

[25] Clune J. Bengio Y. Yosinski, J. and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[26] Kingsbury B. Sindhwani V. Arisoy E. Sainath, T. N. and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6655–6659.

[27] Kyuyeon Hwang Anwar, Sajid and Wonyong Sung, "Structured pruning of deep convolutional neural networks," in *ACM Journal on Emerging Technologies in Computing Systems (JETC) 13.3*, 2017, p. 32.

[28] W. Sung and K. I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," in *IEEE transactions on Signal Processing*, 1995, pp. 3087–3090.

[29] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights+ 1, 0, and 1," in *IEEE Workshop on Signal Processing Systems (SiPS)*, 2014, pp. 1–6.

[30] et al. Jacob, Benoit, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

ABSTRACT

Nowadays, many embedded devices, such as smartphones and Amazon Alexa, use automatic speech recognition (ASR) technology for the hands-free interface. Especially neural network-based algorithms are widely employed in ASR because of high accuracy and resiliency in noisy environments.

Neural network-based algorithms require a large amount of computation for real-time operation. As a result, most of today's ASR systems adopt server-based processing. However, privacy concerns and low latency bring increased demand for on-device ASR. For on-device ASR, the power consumption should be minimized to increase the operating time.

Many neural network models have been developed for high-performance ASR. Among them, the recurrent neural network (RNN) based algorithms are most commonly used for speech recognition. Especially long short-term memory (LSTM) RNN is very well known. However, executing the LSTM algorithm on an embedded device consumes much power because the cache size is too small to accommodate all the network parameters. Frequent DRAM accesses due to cache misses not only slow the execution but also incur a lot of power consumption. One possible solution to mitigate this problem is to compute multiple output samples at a time, which is called the multi-time step parallelization, to reduce the number of parameter fetches. However, the complex feedback structure of LSTM RNN does not allow multi-time step parallel processing.

This thesis presents a Residual Simple Gated Convolutional Network (Residual Simple Gated ConvNet) model with only about 1M parameters. Nowadays, many CPUs can accommodate neural networks with a parameter size of 1M in cache me-

mory. Thus, this model can run ASR very fast and efficiently without consuming much power. The developed model is also based on a convolutional neural network, thus the multi-time step processing can easily be applied. To achieve high accuracy with a small number of parameters, the model employs one-dimensional depthwise convolution, which helps to find temporal patterns of the speech signal. We also considered inception residual connections to reduce the needed number of layers, but this approach needs to be more improved. The developed Residual Simple Gated ConvNet showed very fairly high accuracy even with 1M parameters when trained on WSJ speech corpus. This model demands less than 10% of CPU time when running on ARM-based CPUs for embedded devices.

keywords: Speech Recognition, Sequence Modeling, Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), Embedded Device
student number: 2016 - 26378