논문 2021-1-13 http://dx.doi.org/10.29056/jsav.2021.06.13

화자 겹침 검출 시스템의 프레임워크 전환 연구

김회남*, 박지수*, 차신*, 손경아**, 윤영선*† , 박전규***

Framework Switching of Speaker Overlap Detection System

Hoinam Kim*, Jisu Park*, Shin Cha*, Kyung A Son**, Young-Sun Yun*† , Jeon Gue Park***

요 약

본 논문에서는 화자 겹침 시스템을 소개하고 인공지능 분야에서 널리 사용되는 프레임워크에서 이미 구축된 시스템을 전환하는 과정을 고찰하고자 한다. 화자 겹침은 대화 과정에서 두 명 이상의 화자가 동시에 발성하는 것을 말하며, 사전에 화자 겹침을 탐지하여 음성인식이나 화자인식의 성능 저하를 예방할 수 있으므로 많은 연구가 진행되고 있다. 최근 인공지능을 이용한 다양한 응용 시스템의 활용도가 높아지면서 인공지능 프레임워크 (framework) 간의 전환이 요구되고 있다. 그러나 프레임워크 전환 시 각 프레임워크의 고유 특성에 의하여 성능 저하가 관찰되고 있으며 이는 프레임워크 전환을 어렵게 하고 있다. 본 논문에서는 케라스 (Keras) 기반 화자 겹침 시스템을 파이토치 (pytorch) 시스템으로 전환하는 과정을 기술하고 고려해야 할 구성 요소들을 정리하였다. 프레임워크 전환 결과 기존 케라스 기반 화자 겹침 시스템보다 파이토치로 전환된 시스템에서 더 좋은 성능을 보여 체계적인 프레임워크 전환의 기본 연구로서 가치를 지닌다고 할 수 있다.

Abstract

In this paper, we introduce a speaker overlap system and look at the process of converting the existed system on the specific framework of artificial intelligence. Speaker overlap is when two or more speakers speak at the same time during a conversation, and can lead to performance degradation in the fields of speech recognition or speaker recognition, and a lot of research is being conducted because it can prevent performance degradation. Recently, as application of artificial intelligence is increasing, there is a demand for switching between artificial intelligence frameworks. However, when switching frameworks, performance degradation is observed due to the unique characteristics of each framework, making it difficult to switch frameworks. In this paper, the process of converting the speaker overlap detection system based on the Keras framework to the pytorch-based system is explained and considers components. As a result of the framework switching, the pytorch-based system showed better performance than the existing Keras-based speaker overlap detection system, so it can be said that it is valuable as a fundamental study on systematic framework conversion.

한글키워드 : 화자겹침, 기계학습 프레임워크 전환, adam 최적화기, 학습률 변화기

keywords : speaker overlapping, machine learning framework switch, adam optimizer, learning rate scheduler

* 한남대학교 정보통신공학과
** 울산과학기술원(UNIST) U교육혁신센터
*** 한국전자통신연구원 인공지능연구소
† 교신저자: 윤영선(email: ysyun@hnu.kr)
접수일자: 2021.05.29. 심사완료: 2021.06.15.
게재확정: 2021.06.20.

1. 서 론

음성인식 시스템은 사람이 발화한 음성을 텍스트로 변환하여 컴퓨터 또는 장치로 전달하는 것을 의미하며 화자인식 시스템은 발화한 사람을 확인 또는 식별하는 기술을 뜻한다. 최근에는 발성 화자 별로 음성인식 결과를 회의록으로 작성하여 보관 또는 출력하여 콜센터, 회의, 병원 등과 같은 곳에서 사용되며[1] 화자 고유 특성을 이용하여 보안인증, 접근제어, 개인화 등에 활용하고 있다. 음성인식, 화자 인식 시스템에서 화자 간 겹침 발생은 성능 저하를 불러오는 요인 중 하나로 그 구간을 검출하는 것은 해결해야 할 중요한 문제이다.

최근 인공지능과 하드웨어의 발달로 인공신경망은 음성인식, 화자 인식 등에 접목되어 많은 연구에 사용되고 있다. 인공신경망 연구는 다양한 연구단체에서 관련 프레임워크를 공개하여 연구자들이 인공지능 관련 연구를 확산하거나 성능을 향상시키도록 지원하고 있다. 대표적으로 티아노 (theano), 텐서플로 (tensorflow), 케라스, 카페 (caffe), 토치 (torch) 등이 있다. 티아노[2]는 파이썬 (python) 기반의 최초 딥러닝 프레임워크 중 하나로 CPU, GPU의 수치계산에 매우 유용하고 저수준 라이브러리로 딥러닝 모델을 직접 만들 수 있으나 확장성이 미흡하여 다중 GPU 지원이 부족하다는 약점이 있다. 텐서플로[3]는 가장 인기 있는 프레임워크 중 하나로 여러 CPU, GPU, TPU 등 다양한 연산장치에서 사용이 가능하며 파이썬뿐만 아니라 C++과 R과 같은 다른 언어도 지원하고 딥러닝 모델을 직접 작성하거나 케라스와 같은 래퍼 라이브러리의 사용이 가능하다. 케라스[4]는 효율적인 신경망 구축을 위한 단순화된 인터페이스로 개발되었다. 파이썬 기반으로 작성되어 매우 가볍고 쉽게 몇 줄의 코드로 신경망을 만들 수 있고 티아노, 텐서플로 등에서 작동이 가능하다. 카페[5]는 표현/속도 및 모듈성을 염두하고 개발된 라이브러리 중 하나이며, 파이썬 인터페이스를 가지고 있는 C++ 라이브러리이다. 가장 큰 특징 중 하나는 'caffe model zoo'를 통해 미리 훈련된 여러 네트워크를 바로 사용할 수 있다. 토치[6]는 최대한의 유연성을 달성하고 모델을 제작하는 과정을 간단하게 만드는 것을 목표로 만들어진 프레임워크이다. 초기에는 Lua 기반의 딥러닝 프레임워크로 개발되었으며, 파이썬까지 지원하면서 사용범위가 확대되고 있다.

인공지능 개발 프레임워크는 다양한 라이브러리와 알고리즘을 제공하기때문에 개발자는 핵심적인 알고리즘만 개발하여 시간과 비용을 단축할 수 있다. 하지만 각각의 목적 및 특성에 따라 기능상의 차이가 발생하고 사용자들은 적용하고자 하는 분야에 따라 목적에 맞는 프레임워크를 사용하여야 하므로 프레임워크를 전환하고자 하는 경우 많은 어려움이 따른다.

본 논문에서는 화자 겹침 검출 시스템을 텐서플로우 기반에서 파이토치 프레임워크로 전환하는 과정에서 고려할 사항 및 과정을 기술하여, 다른 연구자들의 인공지능 프레임워크 전환 시에 도움이 될 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 화자전환이나 음성인식 등에서 인식 저하의 요인 중 하나인 화자 겹침을 검출하는 연구를 소개하고, 3장에서는 인공지능 개발 프레임워크 전환 시 고려할 사항을 살펴본다. 4장에서는 화자 겹침 시스템의 프레임워크 전환 과정, 실험 및 결과를 확인하며 5장에서 결론 및 요약으로 마무리한다.

2. 화자 겹침 검출 연구

화자 겹침은 상대방의 대화에 호응하거나 대

화 중 끼어들기와 같이 참여하거나 오랜 침묵 이후 다수의 화자가 동시에 말하는 경우로 관찰되었다[7]. 이러한 경우들은 화자 전환 검출이나 화자인식 등의 방해요소로 적용되어 화자 겹침 구간을 검출하는 연구가 필요하다.

화자 겹침 검출 시스템의 연구로는 단일 화자 발성 구간, 화자 겹침 발성 구간과 그 외의 구간을 구별하여 처리하는 것이 일반적이다[8]. 그러나 화자 겹침 용 데이터 셋이 충분하지 않기 때문에 음성인식에서 널리 사용되는 TIMIT[9], ETAPE[10] 등을 활용하여 인공적인 시료를 생성하여 사용하는 방법 등이 적용되었다[11-13].

3. 프레임워크 전환 시 고려 사항

3.1 인공지능 개발 프레임워크

프레임워크란 응용프로그램 개발을 위한 여러 라이브러리나 모듈 등을 효율적으로 사용할 수 있도록 하나로 묶은 일종의 패키지 또는 라이브러리라고 할 수 있다. 인공지능 특히 심화학습(deep learning) 프레임워크는 이미 검증된 수많은 라이브러리와 사전학습까지 완료된 다양한 인공지능 알고리즘을 제공함으로써, 개발자가 쉽게 인공지능 연구를 진행할 수 있도록 지원한다.

하지만 다양한 인공지능 개발 프레임워크는 각각의 목적이나 특성에 따라 제공되어 차별화된 기능을 제공하므로 사용자는 목적에 맞는 프레임워크를 선별하여 사용하여야 한다. 특히 하나의 프레임워크를 모든 분야에서 사용하기에는 성능 차이와 구현 및 적용의 어려움이 존재하기 때문에, 본 연구는 프레임워크 전환시 고려해야할 사항과 대표적인 전환 기법을 살펴보았다. 본 연구에서 사용하는 기본 시스템은 텐서플로 2.x 기반의 케라스 인터페이스를 사용하였으며, 전환할 대상은 파이토치 프레임워크이다. 본 절에서는 전환대상으로 삼은 인공지능 개발 프레임워크를 소개한다.

가) 텐서플로

텐서플로는 다양한 하드웨어 환경에서 인공신경망 모델을 쉽게 생성하고 학습할 수 있는 인터페이스를 제공하며 여러 가지 장점이 존재한다. 먼저 인공신경망 모델을 학습하는데 필요한 하드웨어의 자세한 조작에 신경 쓰지 않고 모델 구조에만 집중할 수 있다. 다음으로 추상화(abstraction) 기능이 제공되어 알고리즘 구현의 세부적인 면이나 함수와 함수 간의 입/출력의 적절한 방법 등의 세부적인 면을 신경 쓸 필요 없이 전체적인 논리에만 집중할 수 있다[14].

텐서플로는 딥러닝 프레임워크 중 가장 많이 활용되고 있으며, 현재는 약점으로 지적되었던 정적 그래프 모델링을 개선하여 텐서플로 2.x으로 업데이트 되었다. 기존의 텐서플로는 다른 딥러닝 라이브러리에 비해 익히기가 어려웠지만 텐서플로 2.x은 단순성과 편의성에 초점을 두며, 즉시 실행 (eager execution)을 이용한 쉬운 모델작성과 단순화된 API, 강력한 실험법 등의 특징을 갖는다. 텐서플로 2.0 부터 핵심기능의 일부로 케라스를 채택하였으며, 케라스에 적합하도록 최적화하여 모든 기능을 지원하고 있다[15].

나) 파이토치

파이토치는 토치 및 카페2 프레임워크를 기반으로 한다. 파이썬을 스크립팅 언어로 사용하며 진화된 토치 C/CUDA 백엔드를 사용한다. 파이토치는 속도를 극대화하기 위해 인텔 MKL, 엔비디아, NCCL과 같은 라이브러리를 통합했으며 구현이 간결하고 텐서플로보다 익히기 쉽다. 정의 후 실행 (define-and-run) 방식의 텐서플로와 다르게 동시 정의 실행 (define-by-run) 방식으로 동작하며 각 전파 (propagation)마다 새로운

계산 그래프를 정의해서 이용한다.

다) 케라스

케라스는 파이썬으로 작성되었으며 신경망 모델 구축을 위한 고수준 프론트엔드 프레임워크이다. 다양한 분야에 필요한 모델에 대해 직관적인 API를 제공하며 텐서플로, CNTK, 티아노, MXNet, PlaidML 등의 백엔드 딥러닝 프레임워크를 지원하여 새로운 문법의 학습없이 동일한 방식으로 케라스를 적용할 수 있다. 케라스는 배우거나 모델을 구축하기 쉽고 폭넓은 도입과 광범위한 프로덕션 배포 옵션 지원 및 GPU와 분산 학습을 지원하는 강점이 있다. 이러한 장점으로 인하여 구글, 마이크로소프트, 애플, 등의 대규모 기업들이 지원하고 있으며, 사용자층도 확산되고 있다.

3.2 구성 레이어 종류

레이어 (Layer, 층)는 신경망을 구성하는 기본 구성 블록을 말한다. 초기의 신경망 개념에서는 입력층, 출력층, 은닉층과 같이 동일한 연산방식을 취하는 블록 또는 묶음을 레이어로 기술하였으나, 최근에는 다양한 형식의 기능을 구현하는 블록을 레이어로 구성하고 있다. 본 논문에서는 발표된 모든 레이어를 전환하기보다는 주 연구대상인 화자 겹침 검출 시스템에서 이용한 레이어를 대상으로 한다. 본 연구에서 고려한 대표적인 레이어에 관련된 내용을 소개한다.

가) 완전 연결 레이어 (fully connected layer)

완전 연결 또는 밀집 (Dense), 선형 (Linear) 레이어 등의 용어로 지칭되는 레이어는 하단 (lower 또는 previous) 레이어를 구성하는 모든 노드와 연결된 가중 값 행렬 연산을 실행한다. 최근에는 확장 (dilation) 또는 샘플링의 개념 등을 적용하여 가중 값의 공유 또는 과적합 예방 등을 시도하기도 한다. 케라스에서는 Dense 레이어로 지칭하고 있으며 파이토치에서는 Linear 레이어로 선언하고 있다. 케라스의 Dense 레이어는 커널 또는 바이어스의 초기화나 정규화 (regularization) 함수, 활성화 함수 등을 선언할 수 있으나 파이토치의 Linear 레이어는 순수하게 입력에 대한 가중 값 행렬 연산에 집중하고 있다.

나) 컨볼루션 레이어 (convolution layer)

컨볼루션 레이어는 입력 값에 대하여 시간적 또는 공간적 컨볼루션 연산을 실행하는 레이어이다. 컨볼루션 연산은 두 개의 함수가 주어졌을 때 한 함수가 다른 함수의 형태를 변경하는 과정을 말하며, 일반적으로 대상 함수와 컨볼루션 연산을 시행하는 함수로 구성된다. 시계열 공간에서의 컨볼루션은 주파수 공간에서의 필터 곱에 해당되기 때문에, 후자의 컨볼루션 함수를 필터 함수라고 지칭하기도 한다. 컨볼루션은 시계열 패턴과 공간적 패턴 또는 그 이상의 차원에 대하여 적용할 수 있으며, 각각 1차 컨볼루션, 2차 컨볼루션 등으로 구현한다. 본 연구는 음성 신호를 대상으로 하기 때문에 1차 컨볼루션 연산을 고려하며, 신경망의 레이어에서도 1차 컨볼루션 레이어만을 고려한다. 케라스와 파이토치의 1차 컨볼루션 레이어의 구현 방식의 가장 큰 차이점은 결과 값의 차원을 표현하는 방식이다. 케라스에서는 필터 값으로 결과 값의 차원을 정의하지만, 파이토치에서는 출력 채널 값으로 정의하며 입력과 출력 형식도 차수와 시계열 길이가 전치 (transpose)되는 차이가 존재한다.

다) 활성화 레이어 (activation layer)

활성화 함수 (activation function)는 초기 인공신공망의 선형 분류 한계를 극복하기 위한 방법으로 가중 행렬과 입력 벡터의 곱을 비선형 시스템으로 변환하는 함수이다. 케라스나 파이토치

모두 활성화함수를 구성 레이어로 제공하고 있으며, 케라스의 경우 밀집 레이어나 컨볼루션 레이어에 활성화 함수를 지정할 수 있다. 초기의 활성화 함수는 시그모이드 함수를 많이 사용하였으나 기울기 (gradient) 소멸 현상을 극복하기 위하여 양의 값에 대해서는 값을 보존하고 음의 값에 대해서 0으로 설정하는 ReLU (Rectified Linear Unit)가 제안되었다. 그 후 음의 값에 대해서도 0으로 설정하지 않고 일정 기울기를 적용하는 Leaky RelU가 제안되었으며, 본 연구에서도 Leaky ReLU 활성화 함수를 사용하였다.

라) 배치 정규화 레이어 (batch normalization layer)

각 레이어의 연산 후에 결과 값들을 평균 이동하고 배율 척도를 조절하여 인공신경망이 빠르게 수렴하고 안정되도록 제어하는 정규화 레이어이다. 정규화 과정은 크게 입력값 그룹(배치)과 무관하게 해당 레이어에 대하여 정규화하는 레이어 정규화와 배치 값별로 정규화하는 배치 정규화 방법이 많이 사용된다. 학습데이터가 많은 경우에는 레이어 정규화 방법을 사용할 수 없으므로 학습데이터 크기에 따라 적절한 방법을 사용한다. 본 연구에서는 배치 정규화 방식을 사용하였다. 케라스와 파이토치의 정규화 관련 기본값이 차이가 있으며, 정규화된 값과 입력값을 고려하는 모멘텀 값의 적용 방식이 다르므로 함수 호출 시에 주의가 필요하다.

마) 드롭아웃 레이어 (dropout layer)

학습데이터에 인공신경망이 과적합 (overfit)되지 않도록 다양한 방법 등이 제안되었으며, 드롭아웃은 가중치 행렬 연산에서 일부 노드 값들을 누락시키는 방식이다. 일부 노드 값 또는 연결선을 누락시키는 방법 등이 많이 사용되나 노드 값 누락은 노드에 연결된 연결선들의 가중치를 갱신하지 않기 때문에 연결선 누락보다는 좀 더 간단한 방식으로 많이 사용되는 방식이다. 케라스나 파이토치의 경우 드롭아웃에 대한 기능은 유사하며 활용 형태 또한 큰 차이가 없다.

3.3 최적화기 (optimizer)

최적화기는 인공신경망이 최고의 성능 또는 최소의 손실 값을 갖도록 연결 가중치나 학습률을 변화시키는 알고리즘을 말하며, 일반적으로 특정 함수가 최소화 또는 최대화 값을 가지도록 문제를 해결한다. 가중치의 조절 방법에 따라 sgd (stochastic gradient descent, 확률적 경사하강법), momentum, adam (adaptive moment estimation) 등으로 구분되며 본 연구에서는 adam 방식을 사용하였다.

가) sgd

sgd [16, 17]는 gd (gradient descent, 경사하강법)의 과도한 업데이트 시간을 감소시키고자 제안되었다. gd는 전체 데이터를 모두 검토하여 손실 값을 구하고 가중치 업데이트를 반복하는 과정에서 많은 시간을 요구한다. 초기의 sgd는 데이터 1개를 분석하고 바로 가중치를 조절하는 방식을 적용하였다. 가중치 업데이트는 전체 데이터 n 개에 대해 각각 실행되어 총 n번의 업데이트를 진행 후 데이터 섞음 (shuffle)을 통해 각각의 데이터는 서로 다른 순서로 업데이트가 이루어진다. n번의 많은 업데이트 횟수를 줄이기 위해 데이터를 B개로 묶어 미니배치 (mini-batch)를 생성하고 손실 값의 평균을 이용하여 업데이트하는 방식이 제안되었다. 미니배치 방식을 사용하면 gd의 간편함과 sgd의 빠른 업데이트 시간의 장점을 모두 가져올 수 있으며 GPU를 이용하여 병렬처리가 가능하다. 미니배치 방식이 보편화되어 최근의 sgd는 미니배치를 이용한 방식을 의미한다. sgd는 데이터를 서로 다른 순서로

업데이트를 진행하는 방식을 채택하였기 때문에, 매 업데이트의 크기와 방향성이 일정하지 않다는 특성이 있다.

나) 모멘텀 (momentum)

모멘텀 [16, 18]은 gd 방식에서 업데이트를 통해 이동하는 과정에 관성을 추가하는 방식이다. 현재 데이터의 업데이트를 통해 이동하는 방향과 별개로 이전에 이동했던 방향의 일정 비율만큼 추가로 이동한다. 관성의 정도를 조절하는 모멘텀 항 값을 추가로 사용하며 일반적으로 0.9 정도의 값을 사용한다. 모멘텀은 sgd의 진동 현상(oscillation)을 해결할 수 있으며, 상대적으로 적은 횟수의 이동으로 더 빠르게 최소의 손실 값에 접근할 수 있다. 또한 관성을 이용하여 같은 방향에 대해 더 많은 거리를 이동하므로 극소 값에 빠지는 문제를 방지할 수 있다. 반면 모멘텀은 현재의 업데이트를 포함하여 이전의 데이터별 업데이트 값까지 기억해야 하므로 기존의 최적화 방법보다 두 배에 해당하는 메모리가 필요하다.

다) adam

sgd와 모멘텀은 모두 각각의 업데이트에 대해 같은 학습률 (learning rate)을 적용했다. adam 방식[16, 19]은 이전 업데이트의 관성을 적용하되 미래의 상황을 판단하여 다른 크기의 학습률을 적용하여 각각 다른 크기의 업데이트를 적용하는 방식이다. 이전의 관성을 참고하여 방향에 대한 업데이트를 크게 하면서도 최솟값에 가까워질수록 세밀한 학습을 통해 안정적으로 하강이 가능하다는 특성이 있다. 대부분의 딥러닝 모델에서 적합하게 작동하며 다른 방식의 최적화기에 비해 상대적으로 성능이 우수하다고 알려져 있다.

3.4 학습률 변화기 (scheduler)

학습률은 매개 변수를 조정할 때 변경하는 비율을 의미한다. 학습률을 잘못 설정하면 학습이 진행되지 않는 등 학습률의 설정은 성능에 큰 영향을 미친다. 학습 시작부터 종료까지 같은 학습률을 적용하거나 학습 과정에서 학습률 변화기를 이용하여 학습률을 조절할 수 있다. 처음은 학습률을 크게 설정하여 빠르게 최적화를 진행하고 최솟값에 가까워질수록 학습률을 작게 설정하여 미세하게 학습하는 것이 성능이 잘 나온다고 알려져있다[20]. 현재까지 널리 사용되는 학습률 변화기는 전체 학습횟수 (epoch)에 따라 변화를 주는 방법과 전체 학습횟수와 상관없이 규칙적으로 변화하는 방식 등이 사용되고 있다. 최근에는 단조식으로 일정하게 학습률이 감소되는 방식이외에도 증가하였다 다시 감소하는 식과 같이 학습 시 모델의 변이를 증가시킨 후 안정화시키는 방식 등도 적극적으로 검토되고 있다.

4. 전환 구현 및 실험

4.1 기준 시스템

화자 중첩 검출을 위하여 다층 신경망 (MLP; Multi-layer Perceptron) 구조와 유사 X-vector (Pseudo X-vector) 구조를 설계하여 케라스로 구현하였다. 다층 신경망 구조는 입력과 출력 레이어를 포함하여 완전 연결 방식으로 구성하였으며, 유사 X-vector 방식은 특징 표현에 해당하는 전반부(하단부) 레이어를 컨볼루션 레이어로 구성하고, 분류 기능에 해당하는 후반부(상단부) 레이어를 완전 연결 방식으로 표현하였다. X-vector는 변량 길이를 갖는 음성신호를 고정 길이의 화자 특성 벡터 (임베딩 벡터)로 변환하여 화자인식에 사용하는 방식으로 기존 방식에 비하여 우수한 성능을 보인 것으로 알려져 있다[19]. 원 X-vector는 전반부와 후반부 모두 완전 연결 레이어로 구성하고, 각 레이어의 노드 연결

표 1. 기준 시스템의 구성
Table 1. Architectures of base systems

MLP	Pseudo X-vector
Dense (1024) leaky_relu Dropout Dense (512) leaky_relu Dropout Dense (256) leaky_relu Dropout Dense (64) leaky_relu Dense (1) Sigmoid	Conv1D (512, 9) leaky_relu BatchNormalization Dropout Conv1D (512, 5) leaky_relu BatchNormalization Dropout Conv1D (512, 1) leaky_relu BatchNormalization Dropout Conv1D (256, 1) leaky_relu Conv1D (256, 1) leaky_relu Dense (256, 1) leaky_relu Dropout Dense (256, 1) leaky_relu Sigmoid

선을 확장하는 방식으로 구성하나, 본 논문에서 사용한 방식은 전반부를 컨볼루션 레이어로 구성하고 한꺼번에 입력으로 받는 입력 특징 프레임의 수를 원 논문에서 제시한 15프레임보다 적은 11프레임으로 사용하였기 때문에 유사 X-vector라 명명하였다. 두 시스템 모두 입력 벡터로 40차 로그 멜 필터뱅크에너지를 11프레임씩 그룹화하여 적용하였다. 두 기준 시스템의 구성을 <표 1>에 정리하였다. 괄호 안의 숫자는 특징 차수 D 및 한번에 전달되는 특징 그룹 수 또는 문맥 정보량 (context length) N이 된다. MLP의 경우에는 입력 특징 $D \times N$가 DN의 펼친 형태로 전달되며., 유사 X-vector의 경우에는 2차원 벡터 (D, N) 형태로 입력된다. 표에 기술된 각 레이어의 세부 명칭은 4.4절의 레이어 변환 부분에서 자세히 설명한다.

4.2 데이터 준비

화자 겹침 검출 시스템을 평가하기 위하여 음성인식에서 널리 사용하는 TIMIT corpus를 이용하였다. 음성 인식 전용의 데이터 집합은 음성 겹침 구간이 존재하지 않아 인위적으로 화자 전환 과정에서 겹침이 발생하도록 생성하였다. TIMIT corpus는 630명의 화자에 대하여 각각의 화자별로 10개의 문장 발성 데이터를 제공하며 462명은 학습용, 168명은 평가용으로 분류된다. 화자 겹침 검출용 음성데이터는 서로 다른 화자에 대하여 독립적인 문장 음성을 임의로 500ms ~ 2s 길이만큼 겹치게 결합하였다. 최종적으로 학습 데이터의 경우 462명의 화자에 대하여 문장별로 (총 4,620문장) 10개의 음성 데이터를 선택한다. 각 문장 데이터는 461명 중에서 임의로 1개의 화자를 선택하여 총 46,200 문장으로 결합된다. 동일한 방식으로 평가용 데이터 16,800 문장의 연결된 음성 데이터를 제작하였다.

4.3 데이터 로더

학습데이터를 신경망의 입력데이터로 전달하고, 미니배치의 형태로 구성하는 방법은 크게 두 가지로 고려할 수 있다. 첫 번째 방식은 전체 파일의 일부분(배치)에 해당하는 모든 특징을 메모리에 로드하여 학습을 진행하는 방법이다. 미니배치 방식의 경우에는 모든 배치마다 동일한 크기를 가져야 하므로 최대 배치크기를 설정하고 배치크기보다 작은 배치의 경우는 0 패딩

표 2. 데이터 로더 구현 비교
Table 2. Comparison of two data loader implementation

single batch	nested batch
for epoch: for *batch* from $F_b \cdot (N_{F_b}, D)$: model.train(*batch*)	for epoch: for file_batch from F_b: for *feature_batch* from (N_{F_b}, D): model.train(*feature_batch*)
F_b : selected from file list with given batch size (N_{F_b}, D) : feature set saved in F_b files	

(zero-padding) 방식을 적용한다. 두 번째 방식의 경우는 중첩 배치(nested batch)를 적용하는 것으로 외부 배치 반복 (outer batch loop)의 경우에는 파일 이름을 선택하고 내부 배치 반복(inner batch loop)의 경우에는 선택된 파일에 저장된 특징 파일을 메모리에 적재하여 학습에 적용하는 방식이다. <표 2>에 두 방식의 차이를 간략하게 정리하였다. 첫 번째 방식의 경우 동일한 배치크기 제약을 고려한 0 패딩에 의한 메모리 낭비와 전체 학습 데이터를 처리하기 때문에 느린 수렴 속도 등이 취약점으로 판단될 수 있다. 반면 두 번째 방식은 부분 데이터 집합을 이용하여 지엽적인 학습을 진행하기 때문에 단방향성 수렴이 아닌 지그재그 수렴 현상을 보일 수 있다. 그러나 메모리 이점과 0 패딩을 적용하지 않는다는 장점과 수렴 결과를 쉽게 확인할 수 있는 특징으로 인하여 본 연구에서는 두 번째 방법을 채택하였다.

4.4 레이어 변환

화자 겹침 검출 연구를 위해 인공지능 프레임워크를 변경하기 위해서는 신경망을 구성하는 각 레이어의 차이를 파악하고 적절하게 변환하여야 한다. 본 절에서는 화자 겹침 검출 연구에서 사용된 케라스의 각 레이어 특성을 살펴보고 해당하는 파이토치의 레이어로 변경하는 과정을 설명한다.

가) 완전 연결 레이어

케라스와 파이토치의 완전 연결 레이어는 큰 차이가 없으나 케라스의 Dense() 함수에는 커널의 초기화 방식 및 활성화 함수를 지정할 수 있다. 따라서 파이토치의 Linear() 함수로 변경 후에 glorot uniform 초기화 방식과 동일한 초기화 함수를 호출하여 각 레이어를 초기화시켰으며, 직접 활성화 레이어를 구축하였다.

나) 컨볼루션 레이어

케라스와 파이토치 레이어 중에서 입력 형태가 다른 레이어 중의 하나로, 파이토치에서 1차원의 입력데이터를 처리하기 위해서는 입력 특징을 전치하여 적용하여야 한다. 케라스의 입력 형식은 (배치, 입력 열의 길이, 입력 차수)가 되며, 파이토치의 경우에는 (배치, 입력 차수, 입력 열의 길이)가 된다. 따라서 파이토치에서는 입력 차수에 해당하는 차원을 채널로 지정하여 컨볼루션 레이어를 구성하여야 한다. 초기화 및 활성화 함수는 완전 연결 레이어와 동일하게 파이토치에서는 독립 레이어로 구성하였다.

다) 활성화 레이어

활성화 레이어 또는 함수는 신경망의 선형 연

산을 비선형 시스템으로 변환하는 단계라고 설명하였다. 다양한 형태의 활성화 함수가 존재하나 본 연구에서는 음의 영역에서도 일정 기울기로 값을 보정하는 leaky relu 방식의 활성화 함수를 선정하였다. 활성화 함수를 적용하기 위하여 기본 값을 조사한 결과 음의 값에 적용하는 기울기가 텐서플로 0.2, 케라스 0.3, 파이토치 0.01로 모두 다르다는 것을 확인하였다. 기준 시스템에서 텐서플로의 leaky_relu() 함수를 채택하였기 때문에 텐서플로의 기본 alpha=0.2로 파이토치의 negative_slope 값을 지정하여 변환하였다.

라) 배치 정규화 레이어

인공신경망을 안정화시키고 입력 값들의 변이를 줄이는 배치 정규화 방식은 케라스와 파이토치의 매개변수 지정에서 큰 차이가 있다. 기본 변수로 두 프레임워크의 정규화 과정을 적용하는 경우에는 큰 차이가 없으나, 본 논문의 연구와 같이 프레임워크 변경을 위한 과정에서는 동일한 이름의 파라미터가 다른 특성으로 인하여 적절한 변환이 이뤄지지 않을 수 있다.

케라스는 BatchNormalization() 함수 이름으로 관련 기능을 제공하지만, momentum 항은 0.99, epsilon 항은 0.001의 기본 값을 갖는다. 파이토치는 BatchNorm1d() 이름으로 momentum 0.1, eps 1e-5로 설정되어 있다. 케라스의 epsilon과 파이토치의 eps는 분산으로 정규화하는 과정에서 0 나눔 오류 (divide-by-zero)를 회피하기 위하여 더해주는 작은 상수이다. momentum 항은 케라스와 파이토치 모두 동일한 이름을 가지지만 적용 대상은 반대 의미를 갖는다. 두 프레임워크 모두 momentum 항은 현재에 수집된 통계적 특성에 새로운 배치 특성을 적용하는 데 사용하는 가중치로서 케라스의 경우 수집된 통계적 특성에 곱하지만 파이토치의 경우 새로운 배치 특성에 곱한다. 즉, 케라스의 momentum 항은 파이토치의 1-momentum에 해당하는 값이다. 본 연구에서는 momentum의 기본 값이 여러 연구를 바탕으로 설정되었을 것으로 판단하여 케라스와 파이토치 기본 값을 수정없이 사용하였다 (즉 케라스는 0.99, 파이토치는 0.1로 설정하였다).

이외에도 케라스와 파이토치의 배치 정규화 레이어의 가장 큰 차이점 중 하나는 단일 특징 벡터에 대한 처리 부분이다. 케라스의 경우에는 단일 특징이 입력되더라도 작은 난수에 의하여 정규화작업 효과를 갖으나, 파이토치의 경우에는 단언(assert) 예외를 발생시킨다. 따라서 전환과정에서는 GPU 수 * 2개 미만의 개수를 갖는 특징 벡터는 제외하고 사용하였다.

마) 드롭아웃 레이어

학습되는 신경망이 학습데이터에 과적합되지 않도록 각 레이어의 노드들을 일정 비율로 제거하여 연산하는 방식이다. 두 프레임워크에서 제공하는 Dropout() 함수는 기본 값을 이용하는 경우에는 큰 차이가 없다. 다만 케라스에서는 입력 형태에 적용하는 mask 패턴을 지정할 수 있으며, 드롭아웃 레이어마다 별도의 난수 기준 (seed)을 지정할 수 있다는 점이 차이라 할 수 있다.

4.5 최적화기

최적화기는 인공신경망이 최고의 성능 또는 최소의 손실 값을 갖도록 연결 가중치나 학습률을 변화시키는 알고리즘을 말하며, 일반적으로 특정 함수가 최소화 또는 최대화 값을 가지도록 문제를 해결한다. 대표적으로 많이 사용되는 방식으로는 sgd와 adam을 들 수 있다. sgd는 기울기가 감소하는 방향으로 최적화하는 방식인데, 전체 데이터를 이용하지 않고 일부 데이터나 무작위로 선택하여 적용한다. 반면 adam 은 sgd 종류의 개선된 방식으로 각 매개변수에 서로 다른 학습률을 선택하는 방식이다. adam이 다른

최적화 방식과 비교해 (특히, sgd) 특정 상황에서 낮은 성능을 보이지만 최근의 많은 딥러닝 연구에서는 안정성이나 수렴 속도 면에서 adam 최적화기를 추천하고 있다[22]. 본 연구에서도 기준 시스템에 adam 최적화기를 채택하였다. 그러나 프레임워크 전환 과정에서 케라스나 파이토치의 adam 기본 설정 매개변수가 달라 케라스를 이용한 기준 시스템의 adam 최적화 기본 값으로 파이토치 adam 최적화기를 초기화하였다.

4.6 학습률 변화기

최초에 케라스 기반의 화자 겹침 검출 시스템을 파이토치 기반의 화자 겹침 검출 시스템으로 전환하는 과정에서 기본 구축 방식으로는 유사한 결과를 얻지 못하였다. 일반적인 프레임워크 전환 연구는 이론적 연구라기보다는 실제적 구현 문제로 판단되기 때문에 관련 문헌이나 논의가 미흡하다. 그러나 포럼이나 github 등에서는 케라스 또는 텐서플로에서 adam 최적화기를 이용한 시스템을 파이토치 기반으로 전환하는 과정에서 동일한 문제가 공유되고 있었으며, 관련 논의 과정에서 학습률의 변경에 따른 비교 가능한 성능을 얻을 수 있었다고 보고하고 있다[23]. 학습률을 변화시키는 방법 중에서 epoch에 따른 추가 학습을 고려하여 StepLR과 lambdaLR를 고려하였으며, 연속적인 값의 변화와 일반화 특성이 높은 lambdaLR을 적용하였다.

4.7 실험 및 결과

케라스 기반의 화자 겹침 검출 시스템을 파이토치 기반으로 전환하기 위한 기준 시스템의 성능을 <표 3>에 정리하였다. 성능을 측정하기 위하여 정확도 (precision)와 복원율 (recall)을 확인하였고, 정확도와 복원율의 조화 평균인 F1을 산출하였다. F1 점수는 데이터 분포가 불균형일 때, 모델의 성능을 정확하게 평가할 수 있으며, 하나의 값으로 성능 비교를 할 수 있기 때문에 최종 비교에 사용하였다. 완전 연결 구조와 드롭아웃 레이어로 구성된 MLP에 비하여 복잡한 구조를 보이는 유사 X-vector모델의 성능이 높아서 유사 X-vector를 파이토치 모델로 전환하여 그 결과를 비교하였다.

표 3. 케라스 기반 기준 시스템 성능
Table 3. Performance of Keras-based Reference system

MLP			Pseudo X-vecotor		
Prec.	Recall	F1	Prec.	Recall	F1
86.20	87.52	86.86	94.47	86.92	90.54

전환대상인 유사 X-vector는 초기 학습률 0.001에서부터 감쇄상수 λ와 반복 횟수 epoch에 따라 매 epoch마다 λ^{epoch}만큼 감쇄하도록 설정하였으며, $\lambda = 1$인 경우에는 학습률 변화기를 적용하지 않은 시스템과 동일하다. <표 4>는 감쇄상수 λ에 따른 전환 시스템의 성능 변화를 정리하였다.

표 4. 파이토치 기반 전환 시스템 감쇄 상수에 따른 성능 변화
Table 4. Performance Comparison of Pytorch-based Converted system along to lambda constant

	λ	Prec.	Recall	F1
fixed	1.0	84.42	79.79	82.04
lambda	0.95	93.69	78.69	85.54
	0.8	94.37	86.46	90.24
	0.7	93.97	88.06	**90.92**

전환 결과 감쇄상수 $\lambda = 0.7$인 경우에 케라스 기반의 기준 시스템보다 성능이 향상된 것을 알 수 있다. 본 연구의 목적이 인공신경망의 초매개변수 (hyper-parameter)를 수정하여 최고의 성능을 얻는 것이 아니라, 서로 다른 프레임워크에서의 성능 차이 없이 시스템을 전환하는 것이기 때문에, 실험결과 성공적으로 케라스 프레임워크에

서 파이토치 프레임워크로 연구 시스템을 전환시켰다고 할 수 있다.

5. 결 론

화자 겹침 검출 연구는 대화 녹취 시스템 구성 과정에서 화자가 전환되는 시점을 파악하는 중요한 기초연구 중의 하나이다. 두 사람 이상의 화자가 동시에 발성을 하게 되면 화자 전환 시점을 검출하는데 어려움이 따르기 때문이다. 본 연구에서는 화자 겹침 검출 시스템에 대하여 간단하게 소개하고, 케라스 기반으로 구축된 시스템을 다른 인공지능 프레임워크인 파이토치로 전환할 때 고려할 요소와 그 과정을 기술하였다.

서로 유사하지만 각 고유 특성을 갖는 인공지능 프레임워크는 고유의 영역을 가지고 연구에 많은 도움을 주고 있다. 파이토치는 텐서플로 기반의 플랫폼보다 많은 유연성과 사용자 정의 기능을 추가할 수 있으므로 최근 그 사용 빈도나 영역이 확대되고 있다. 텐서플로 기반의 케라스로 구축된 화자 겹침 검출 시스템을 파이토치로 전환하는 과정은 이론적인 기여보다는 실제적인 문제를 해결하는 과정에서 고려해야 할 많은 사항이 존재한다. 본 연구진 외에도 텐서플로 기반의 adam 최적화기를 파이토치 기반으로 전환하는 과정에서 많은 어려움이 보고되었기 때문에 본 연구의 결과가 플랫폼 전환을 고려하는 연구에 도움이 되기를 바란다.

플랫폼 전환 결과 감쇄 상수를 고려한 람다 학습율 변화기를 같이 사용하는 경우 케라스 기반의 시스템과 파이토치 기반의 시스템이 유사한 성능을 얻는 것으로 확인되었다. 물론 인공신경망 기반의 많은 연구에서는 모델 구조보다 초매개변수의 설정에 따라 성능이 변화할 수 있다. 이점을 고려하여 기본적인 플랫폼 전환에서 구성요소와 전환 과정을 염두에 둔다면 다양한 플랫폼으로의 시스템 이식 등에 본 연구가 많은 도움이 될 것으로 판단한다.

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2019-0-01376, 다중화자간 대화 음성인식 기술개발).

참 고 문 헌

[1] Reynolds, Douglas A., and P. Torres-Carrasquillo, “Approaches and applications of audio diarization”, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05), Vol. 5. IEEE, 2005. DOI: https://doi.org/10.1109/ICASSP.2005.1416463

[2] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., & Bengio, Y., “Theano: a CPU and GPU math expression compiler”, Proceedings of the Python for scientific computing conference (SciPy), Vol. 4, No. 3, pp. 1-7. 2010. DOI: https://doi.org/10.25080/Majora-92bf1922-003

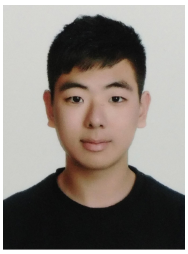
[3] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., & Zheng, X., “Tensorflow: A system for large-scale machine learning”, 12th USENIX conference on Operating Systems Design and Implementation, pp. 265-283. 2016. DOI: https://doi.org/10.5555/3026877.3026899

[4] Ketkar, N. Introduction to Keras. In: Deep learning with Python, pp. 97-111, Apress, Berkeley, CA. 2017. DOI: https://doi.org/10.1007/978-1-4842-2766-4_7

[5] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., & Darrell, T., “Caffe: Convolutional architecture for fast feature embedding”, In Proceedings of the 22nd ACM international conference on Multimedia, pp. 675-678, 2014. DOI: https://doi.org/10.1145/ 2647868.2654889

[6] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., & Chintala, S., “Pytorch: An imperative style, high-performance deep learning library”, arXiv preprint arXiv:1912.01703. 2019.
[7] Adam, A. G., Kajarekar,S. S., & Hermansky, H., “A new speaker change detection method for two-speaker segmentation”, In 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 4, pp. IV-3908, IEEE, 2002. DOI: https://doi.org/10.1109/ICASSP.2002.5745511
[8] Bullock, L., Bredin, H., & Garcia-Perera, L. P. “Overlap-aware diarization: Resegmentation using neural end-to-end overlapped speech detection”, ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 7114-7118, IEEE. 2020. DOI: https://doi.org/10.1109/ICASSP40776.2020.9053096
[9] Garofolo, John S., et al., TIMIT Acoustic-Phonetic Continuous Speech Corpus, LDC93S1. Web Download. Philadelphia: Linguistic Data Consortium, 1993. DOI: https://doi.org/10.35111/17gk-bn40
[10] G. Gravier et al, “The ETAPE corpus for the evaluation of speech-based TV content processing in the French language”, Proceedins of the 8th LREC, pp. 114-118, 2012, ELRA.
[11] Sajjan, N., Ganesh, S., Sharma, N., Ganapathy, S., & Ryant, N., “Leveraging LSTM models for overlap detection in multi-party meetings”, In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5249-5253, IEEE, 2018. DOI: https://10.1109/ICASSP.2018.8462548
[12] Andrei, V., Cucu, H., & Burileanu, C., “Detecting Overlapped Speech on Short Timeframes Using Deep Learning”, INTERSPEECH, pp. 1198-1202, 2017. DOI: https://10.21437/INTERSPEECH.2017-188
[13] Kazimirova, E., & Belyaev, A., “Automatic Detection of Multi-speaker Fragments with High Time Resolution”, In INTERSPEECH, pp. 1388-1392, 2018. DOI: https://10.21437/Interspeech.2018-1878
[14] Bahrampour, S., N. Ramakrishnan, L. Schott, and M. Shah, “Comparative Study of Deep Learning Software Frameworks”, arXiv:1511.06435v3, 2016.
[15] Effective Tensorflow 2.0 Guide, https://www.tensorflow.org/guide/effective_tf2?hl=en, 2021.05.29
[16] Ruder, Sebastian. “An overview of gradient descent optimization algorithms”, arXiv preprint arXiv:1609.04747, 2016
[17] Bottou L., Lechevallier Y., Saporta G., “Large-Scale Machine Learning with Stochastic Gradient Descent”, Proceedings of COMPSTAT’2010, 2010. DOI: https://doi.org/10.1007/978-3-7908-2604-3_16
[18] Ilya Sutskever, James Martens, George Dahl, Geoffrey Hinton, “On the importance of initialization and momentum in deep learning”, Proceedings of the 30th International Conference on Machine Learning, PMLR 28(3), pp. 1139-1147, 2013.
[19] Z. Zhang, “Improved Adam Optimizer for Deep Neural Networks”, 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), pp. 1-2, 2018. DOI: https://10.1109/IWQoS.2018.8624183
[20] Aitor Lewkowycz, “How to decay your learning rate”, arXiv:2103.12682v1, 2021.
[21] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey and S. Khudanpur, “X-Vectors: Robust DNN Embeddings for Speaker Recognition”, 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 5329-5333, 2018. DOI: https://10.1109/ICASSP.2018.8461375.
[22] V. Bushaev, Adam – latest trends in deep learning optimization, https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c, 2021.01.11.
[23] Suboptimal convergence when compared with TensorFlow model, https://discuss.pytorch.org/t/suboptimal-convergence-when-compared-with-tensorflow-model/5099/38, 2021.02.22.

저 자 소 개



김회남(Hoinam Kim)

2019.08 한남대학교 정보통신공학과 졸업
2019.09-현재 : 한남대학교 정보통신공학과 석사과정
<주관심분야> 음성인식, 화자인식, 인공지능 등

박지수(Jisu Park)

2017.2 건양대학교 의료it공학과 졸업
2019.2 한남대학교 정보통신공학과 석사
2019.8-현재 : 한남대학교 박사과정
<주관심분야> 음성인식, 음성변환, 화자인식, 인공지능 등

차신(Shin Cha)

1995년 KAIST 전산학과 박사
1986-2000 LG전자기술원 책임연구원
2000-2013 (주)IA 멀티미디어통신사업부 사업본부장
2013-2015 (주)슈어소프트테크 고신뢰검증센터 센터장
2016-현재 한남대학교 교수
<주관심분야> 소프트웨어 신뢰성, 안전성공학, IoT 보안, 화자인식 등

손경아(Kyung A Son)

2003.2 한양대학교 교육공학박사
2003.11-2014.1 한국방송통신대학교 책임연구원
2005.9-2008.8 한양대학교 컴퓨터교육과 겸임교수
2014.2-2015.12 국가평생교육진흥원 국정과제추진단 부단장
2015.1-현재 UNIST U교육혁신센터 연구교수
<주관심분야> 에듀테크, 컴퓨터교육, 멀티미디어, 학습관리시스템(LMS), 이러닝 개발 및 운영 등

윤영선(Young-Sun Yun)

2001.2 KAIST 전산학과 박사
2006.4-2007.2 한국전자통신연구원 초빙연구원
2012.8-2013.7 University of Washington 방문학자
2001.3-현재 : 한남대학교 교수
<주관심분야> 음성인식, 음성변환, 화자인식, 인공지능, 저작권침해, 유사도, 완성도 감정, 오픈소스 등

박전규(Jeon Gyu Park)

1987년 한국외국어대학교 전산학과 졸업
1989년 한국외국어대학교 전산학과 석사
2010년 배재대학교 정보통신공학 박사
1991-1999 한국전자통신연구원 선임연구원
2000-2001 L&H Korea Head of Speech Div.
2001-2002 Carnegie Mellon Univ. 객원연구원
2002-2004 동아시테크(주) 이사/연구소장
2004-현재 한국전자통신연구원 책임/인공지능연구소 복합지능연구실장
<주관심분야> 음성언어처리, 자연어처리, 인공지능 등