딥러닝 기반 한국어 Lip Reading 시스템 개발

*김기현,**안태현

국민대학교 전자공학부

*rlgus4995@kookmin.ac.kr,**ammjy1122@kookmin.ac.kr

요 약

본 보고서에서는 입술 움직임을 통해 어떤 단어를 발화하고 있는지를 분석하는 Lip Reading 기술을 구현하여 실시간 영상에서 발화하고 있는 단어를 출력 해주는 시스템을 개발하는 프로젝트를 진행하였다. 립리딩 특성 상 시퀀스 영상 데이터를 처리해야 하고, 연산량을 고려하여 MobileNetV2 와 LSTM(Long Short-Term Memory) 순환 신경망의 결합 모델을 사용하였다. 또한, Overfitting 을 방지하기 위한 Dropout, K-fold Cross Validation, Data Augmentation 등을 사용하여 모델의 Accuracy 를 강화하고, 실시간 Lip Reading 시스템 개발에 성공하였다.

1 서론

Lip Reading 은 영상에서 입술 움직임을 분석해 음성 신호 없이도 발화된 단어를 인식하는 기술로, 소음이 많은 환경이나 청각 장애인을 위한 의사소통 수단이나 기타 소음 및 잡음이 많은 환경에서 활용할 수 있는 의사소통 수단으로 유용하다. 본 프로젝트에서는 MobileNetV2 와 LSTM 을 결합한 모델을 사용해 한국어 Lip Reading 시스템을 개발하고 웹캠을 사용해 실시간으로 detect 하는 프로그램을 개발하였다.

2 과제 수행 내용

2.1 Dataset

본 프로젝트에서 학습 및 평가에 사용된 데이터셋은 약 27 명의 발화자가 ‘아’, ‘여’, ‘마’, ‘오’, ‘애’, ‘임’, ‘으’, ‘어’, ‘워’, ‘왠’, ‘우’, ‘안’ 등의 음절을 두 번씩 발화하는 모습을 dlib 라이브러리를 이용해 입술 모양만 crop 하여 구성하였다. 이는 한국어의 특성 상 단어 혹은 구문 단위로 학습시키기보다는 모음과 양순음 음절 단위로 학습시키고 이를 합쳐 후처리 하는 과정으로 단어를 유추하는 것이 더 높은 정확도를 보일 것이라는 가정에 기반하였다. 실제로도 이러한 접근 방식은 단어 및 구문 단위로 학습시키는 것에 비해 연산량, 정확도, 확장성 면에서 우수한 성능을 나타냈다. 총 55 개의 시퀀스 데이터 중에서 41 개는 학습용, 10 개는 검증용, 나머지는 평가용으로 구성하였다.

2.2 Data preprocessing

데이터 획득 과정은 각 발화자가 한 음절씩 발화하는 동영상을 촬영한 후, 첫 10 프레임을 추출하여 dlib 라이브러리를 이용해 얼굴 랜드마크를 생성한 뒤 입술 부분만 crop 하여 데이터 셋에 추가하는 방식으로 진행하였다.

그림 1. dlib 을 이용한 얼굴랜드마크 검출 과 입술 crop

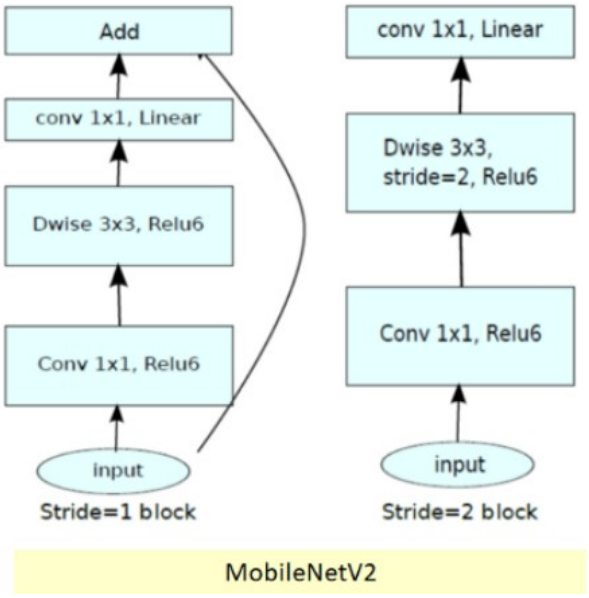
각 음절 발화 데이터는 10 장의 시퀀스 입술 이미지 데이터로 구성되어 있으며, MobileNetv2 입력을 위해 (224,224) 크기로 조정하고 RGB 채널을 정규화하였다. 데이터셋에 추가된 이미지들은 일정 확률로 좌우 반전, 약 20 도 회전, 밝기와 채도 변화, 가우시안 블러링 등의 랜덤 데이터 증강이 적용되었다.

그림 2. Random augmentation sequence data

2.3 model design

2.3.1 MobileNet v2

이미지를 학습하는 모델 을 설계하고자 했을 때, 가장 먼저 고려했던 것은 CNN 계열의 모델들 이었다. 그중에 실시간 어플리케이션에 적합한 효율적인 연산과 단순한 구조의 경량화 네트워크를가진 MobileNet V2 를 선택하였다. MobileNet V2 는 기존 MobileNet V1 이가진 Depthwise Separable Convolution 을 활용한 것과 더불어 inverted Residual 구조를 사용하여 학습 성능을 끌어올렸다. 구조는 다음과 같다.



2.3.2 LSTM

학습에 사용하는 데이터는 시간의 순서에 따른 시퀀스 이미지 데이터이기 때문에 RNN(Recurrent Neural Network)의 일종인 LSTM 을 사용한다. LSTM 은 RNN 의 단점인 시퀀스가 길어질수록 학습이 어려워지는 문제(기울기 소실 문제)등이 있고 LSTM 은 이러한 문제를 해결하기 위해 개발되었다. LSTM 은 기본적으로 셀 상태와 여러 게이트로 구성되어 있다. 셀 상태는 정보의 흐름을 조절하며, 게이트는 정보를 추가하거나 제거여 셀 상태를 업데이트 한다.

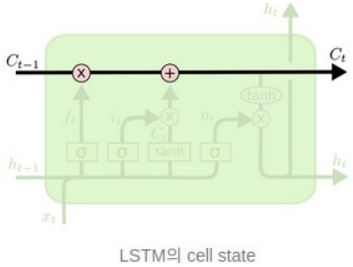


그림 3. LSTM cell state

셀 상태는 이전 상태에서 현재 상까지 유지되는 정보의 흐름을 나타낸다. 이를 통해, LSTM 은 오래된 정보를 기억하고 새로운 정보를 적절히 갱신할 수 있다. 셀 상단의 cell state 가 있으며 자기 자신에게 피드백 된다. 또한 망각 게이트와 입력 게이트를 거치고 나온 결과가 다음 입력이 된다.

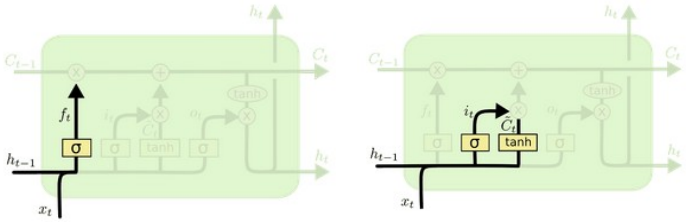
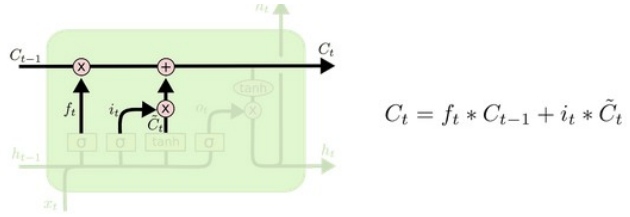


그림 4. Forget gate, input gate layer

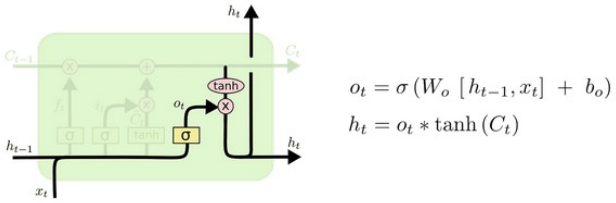
게이트는 주로 sigmoid 함수(값이 0 에서 1 사이인 출력)를 사용하여 정보를 선택적으로 통과시키는 역할을 한다. 망각 게이트는 셀 상태에서 이전 상태의 정보를 얼마나 잃어버릴지 결정하고 sigmoid 함수를 사용해, 값이 0 이면 셀 상태 정보는 0 이되어 사라지고 1 이면 그대로 전달된다.

입력 게이트는 새로운 정보를 셀 상태에 어떻게 반영할 것인지 결정하고 이러한 망각, 입력 게이트를 통해 LSTM 은 기존 정보와 새로운 정보를 적절하게 조합하여 예측 수행이 가능해진다.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

그림 5. LSTM cell state 업데이트



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

그림 6. Output layer

출력 게이트에서는 셀 상태에서 어떤 정보를 출력할지 결정한다. 값을 그대로 사용하지 않고, 출력의 중요도에 따라서 sigmoid 함수를 적용해 0~1 사이 값으로 만든 후 출력하고자 하는 신호에 곱해서 출력한다.

Lipreading 특성 상 약 10 장 가량의 연속된 입술 모양의 움직임으로 말하고 있는 음절을 예측해야 하기 때문에 학습시키기에 위와 같은 LSTM 모델이 적합하다고 생각되었다.

2.3.2 Prevent Overfitting

위에서 말한 모델을 사용하여 학습한 결과 data 의 수가 많지 않기 때문인지 아래와 같이 Trainset 에 대한 overfitting 이 발생하였다.

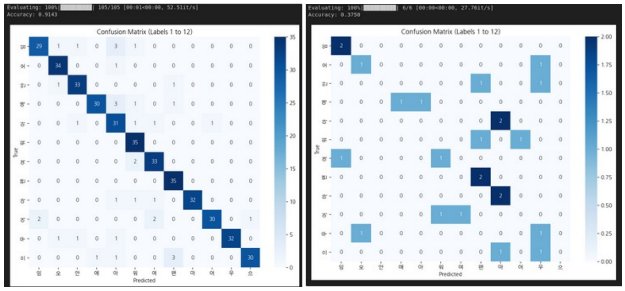


그림 7.학습 후 trainset(좌)와 testset(우)에 대한 혼동 그래프

이렇게 overfitting 되는 것을 방지 하기 위해서 다음과 같은 방법을 적용하였다.

1. Drop out 설정

Drop out 이란 모델이 과적합 하는 것을 방지하기 위해 각 layer 끼리 연결된 연결망에서 확률적으로 뉴런을 제거하는 기법이다.

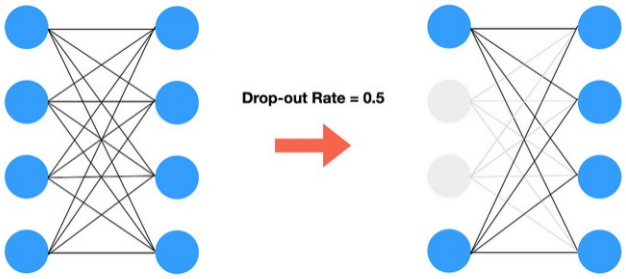


그림 8. dropout

본 프로젝트에서는 여러 확률로 실행해본 결과 dropout rate= 0.6 일 때가 가장 좋은 결과를 보여주어서 dropout rate 를 0.6 으로 설정하고 학습을 진행 하였다.

2. data augmentation

모델의 과적합을 방지하고 데이터의 수를 증강시키기 위해 그림 2 와같이 데이터를 좌우 반전,회전,색상 변경, 블러링 등의 데이터에 변형을 주는 기법을 말한다. 이도 dataset 의 transform 을 할 때 아래와 같이 Torchvision 패키지를 이용하여 확률적으로 적용할 수 있다.

```
train_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomRotation(degrees=20),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.1),
    transforms.GaussianBlur(kernel_size=(5, 9), sigma=(0.1, 5)),
```

3. 5-fold cross validation

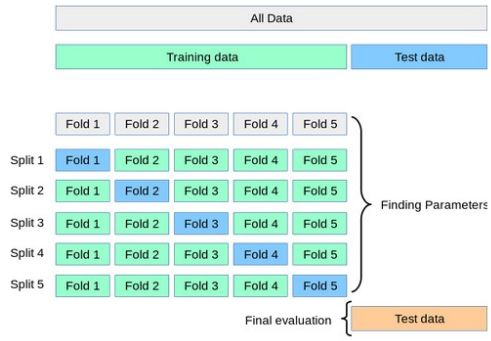


그림 9. 5-fold cross validation

데이터셋의 약 20%를 검증 데이터셋으로 사용하면 유의미한 데이터를 학습에 사용하지 못하게 되어 성능 향상이 제한될 수 있고, 과적합이 발생할 수 있다. 이를 해결하기 위해, 전체 데이터셋을 5 개의 부분으로 나누어 각 단계에서 다른 검증 데이터셋을 사용해 학습을 진행한 후, 5 번의 교차 검증이 끝나면 결과를 종합하여 베스트 모델을 선정하는 5-fold cross validation 을 적용하였다. 이 방법은 모든 데이터셋을 학습에 활용할 수 있게 해주며, 과적합 방지와 학습 성능 향상에 큰 도움을 주었다.

2.3.4 최종 학습 모델

최종 학습 모델은 MobileNet V2 와 LSTM 모델을 결합하여 사용하였고 아래와 같다.

```
# MobileNetV2 + LSTM + Dropout 모델 정의
class MobileNetV2LSTM(nn.Module):
    def __init__(self, num_classes, dropout_rate=0.6):
        super(MobileNetV2LSTM, self).__init__()
        mobilenet = models.mobilenet_v2(pretrained=True)
        mobilenet.features = nn.Sequential(
            *list(mobilenet.features),
            nn.AdaptiveAvgPool2d((1, 1))  # 추가: AdaptiveAvgPool2d로 마지막 출력 크기를 (1, 1)로
        )
        self.mobilenet = mobilenet.features
        self.lstm = nn.LSTM(1280, 256, batch_first=True)  # MobileNetV2의 출력 크기 1280
        self.dropout = nn.Dropout(dropout_rate)  # Dropout 레이어 추가
        self.fc = nn.Linear(256, num_classes)

    def forward(self, x):
        batch_size, seq_length, c, h, w = x.size()
        x = x.view(batch_size * seq_length, c, h, w)
        x = self.mobilenet(x).squeeze(-1).squeeze(-1)  # (batch_size * seq_length, 1280, 1,
        x = x.view(batch_size, seq_length, -1)
        x, _ = self.lstm(x)
        x = self.dropout(x)  # Dropout 적용
        x = self.fc(x[:, -1, :])
        return x
```

10 장의 고정된 시퀀스 이미지 데이터를 MobileNet V2 의 입력으로 받아 각 이미지에 대해 1280 차원의 출력 특징 벡터를 생성한다. 이 특징 벡터들은 시퀀스 데이터로서 LSTM(Long Short-Term Memory) 네트워크의 입력으로 사용된다. LSTM 은 시간적 의존성을 고려하여 이 특징 벡터들을 처리하고, 최종적으로 각 클래스 별로 fully connected layer 를 통해 학습을 진행한다.

이 모델 구조는 단순히 CNN 과 LSTM 을 결합한 모델(CNN+LSTM), CNN 과 RNN 을 결합한 모델(CNN+RNN), 또는 MobileNet 과 LSTM 을 결합한 모델(MobileNet+LSTM)보다 더 우수한 학습 성능을 보여주었다. MobileNet V2 의 효율적인 특징 추출 능력과 LSTM 의 시퀀스 데이터 처리 능력을 결합함으로써, 본 프로젝트의 모델은 입술 움직임의 미세한 변화를 효과적으로 학습하고 높은 정확도를 달성할 수 있었다.

2.4 Data post processing

각 모음과 양순음에 대한 음절을 학습시켰기 때문에 이를 실제 사용 가능한 시스템으로 만들기 위해선 실시간 웹캠으로 받아오고 예측한 음절들을 합쳐 단어를 유추하게 하는 후처리 과정이 필요하다. 우리는 실시간으로 받아온 음절을 리스트에 저장하고 다음과 같은 매핑 과정을 통하여하여 결과 단어를 출력하였다.

원 단어		모음화
안녕	→	안여
아마도	→	아마오
힘들어	→	임으어
놀라워	→	오아워
괜찮아	→	왠안아
누구	→	우우
좋아	→	오아
고마워	→	오마워
졸려	→	오여
사랑해	→	아아애

예를 들어 저장된 리스트에 '안' 과 '여' 가 저장 되어 있다면 '안녕'을 출력한다. 하지만 이렇게 후처리를 한다면 '안녕'을 말했는데 '안','여' 가 인식되는 것이 아닌 '아','여' 가 인식 되었을 때 올바른 결과를 도출 할 수 없을 것이다. 이를 해결하기 위해서 difflib 패키지의 SequenceMatcher 모듈을 이용해 받아온 음절 리스트와 가장 비슷한 단어를 유추하는 clustering 함수를 정의한다.

위 함수는 현재 받아온 음절 리스트와, 정답 음절 리스트, 정답 음절 리스트에 따라 출력하는 단어의 리스트를 인자로 받는다. 현재 받아온 음절 리스트를

```python
def clustering(str,list1, list2):
    score = []
    for i in range(len(list1)):
        ratio = SequenceMatcher(None, str, list1[i]).ratio()
        score.append(ratio)
    index = score.index(max(score))
    if len(list2[index]) == len(str):
        return list2[index]
```

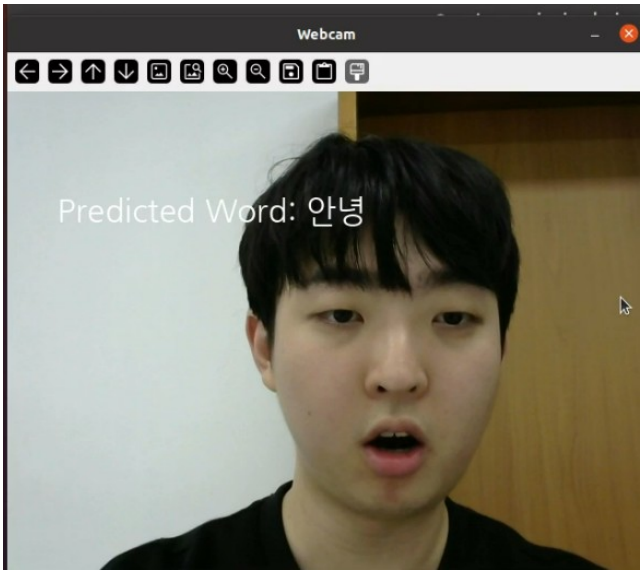
정답 음절 리스트와 SequnceMatcher 함수를 통해 비교한 뒤 가장 점수가 높은 인덱스를 반환해 그 인덱스에 해당하는 단어를 return 하는 식으로 진행된다.

```
Predicted Word: 오마워
놀라워
```

위와 같은 방식으로 진행했을 때 원래는 '오아워'가 인식되어야 '놀라워'가 출력 되지만, Sequence Matcher 함수가 가장 가까운 정답을 찾아줘 '놀라워'로 잘 도출되는 것을 확인 할 수 있다.

3 실험 결과 및 분석

지금까지 기술한 방법으로 시스템을 구성하였고 총 2 사람이 11 단어에 대해서 테스트를 진행하였다.




```
Predicted Syllable: 오
Predicted Word: 안오
안녕
```

학습시킨 모델을 평가모드로 사용해 위와같이 웹캠을 통해 실시간으로 음절을 받아온뒤 후처리하는 과정을 거쳐 최종 출력단어를 실시간으로 plot 한다.

안녕,아마도,힘들어,놀라워,괜찮아,좋아,사랑해
단어에서는 높은 빈도로 참값을 출력하는 정확도를 보여주 었지만. 누구,졸려 같은 단어에서는 높은 빈도로 참값을 예측하지 못했다. 이는 training 과정에서 '우', 와,'여' 음절의 학습이 잘되지 못한 이유로 보인다.

4 결론

본 프로젝트에서는 한국어 Lip Reading 시스템을 성공적으로 개발하였다. MobileNet V2 와 LSTM(Long Short-Term Memory) 모델을 결합하여 실시간 영상에서 입술 움직임을 분석하고 발화된 단어를 예측하는 시스템을 구현하였다. 데이터셋은 약 30 명의 발화자가 발화한 음절들을 기반으로 구성되었으며, 데이터 증강과 5-fold 교차 검증을 통해 모델의 정확도를 높였다. 최종 모델은 MobileNet V2 로부터 추출된 특징 벡터를 LSTM 에 입력하여 처리하고, Fully Connected Layer 를 통해 예측하는 방식으로 구성되었다. 실험 결과, 본 시스템은 단어 및 구문 단위로 학습하는 것보다 우수한 성능을 보였으며, 실시간 적용 가능성을 확인하였다. 이번 연구를 통해 딥러닝 기반의 Lip Reading 기술이 한국어 환경에서도 효과적으로 구현될 수 있음을 확인하였다.

참고문헌

1 LSTM 이해하기
https://dgkim5360.tistory.com/entry/understanding-long-short-term-memory-lstm-kr

2 Mobilenetvw,Inverte Residuals and Linear Bottlenecks
https://gaussian37.github.io/dl-concept-mobilenet_v2/

3
4
5
6
7
8