Recognition and Prediction for EMNIST-Letters Image Sequences Combining VGG-5 and LST

Chan Jeong Park
Dept. of Electrical and Computer Engineering, Seoul National University
ckswjd99@snu.ac.kr

Abstract

본 프로젝트에서는 문자열 이미지를 인식하여 다음으로 등장할 문자열을 예측하는 모델을 구성하였다. VGG와 LSTM 모델을 기반으로 고정 길이의 문자열, 가변 길이의 문자열을 인식하고 다음 문자열을 예측하는 모델을 설계하였으며, 성능을 개선할 수 있는 몇 가지 기법들을 시도하였다.

1. Introduction

본 프로젝트에서는 EMNIST-Letters 데이터셋으로 구성된 이미지 배열을 인식하고, 다음으로 올 문자를 예측하는 모델을 설계 및 학습한다. 그 과정에서 높은 성능을 보이는 모델을 선별함과 더불어, 어떤 점이 그 모델의 성능을 향상시켰는지 분석하는 것을 본 프로젝트의 목적으로 한다.

2. Related Works

VGG: VGGNet[1]은 2014 년 ILSVRC에서 준우승을 거둔 모델로, 네트워크의 깊이를 늘리는 것에 집중하고 각 커널의 크기는 작게 유지한 모델이다. 그 덕에 전체적인 구조가 매우 간단하면서도 EMNIST Letters 데이터셋에 대하여 약 95%의 높은 성능을 내는 것으로 알려져 있다[2]. 이러한 이유로 해당 모델의 구조를 본 프로젝트의 CNN 부분에 사용하기로 결정하였다.

LSTM: LSTM(Long Short-Term Memory)[3] 은 어떤 정보가 짧게, 혹은 길게 기억되어야 하는 경우에 대하여 유동적으로 처리하는 능력을 가진 RNN 모델이다. 가장 큰 특징은 cell state의 개념을 도입하여 현재 갖고 있는 정보 중 어느 정도를 기억하고, 새로운 입력을 얼마나 받아들이고, 출력으로 얼마나 내보낼 지 결정한다는 점이다. 본 프로젝트에서 RNN 부분은 One hot vector 형태로 표현된 문자열의 전체 구조에 대해서도 이해할 필요가 있으므로 LSTM을 사용하기로 결정하였다.

3. Methods

3.1. Dataset

Easy: Training Dataset 50,000 개, Validation Dataset 10,000 개로 이루어져 있다. 각각의 데이터는 5 개의 EMNIST-Letters 이미지와 6 개의 라벨로 구성된다. 5 개의 이미지가 어떤 문자인지에 대한 정보는 앞 5 개의 라벨을 통해 주어지며, 그 5 개 이후로 이어져야 하는 한 개의 문자에 대한 정보가 6 번째 라벨을 통해 주어진다. Label은 항상 등차수열을 이루며 그 공차는 1 이상 3 이하이다.

Normal: Training Dataset 50,000 개, Validation Dataset 10,000 개로 이루어져 있다. 각각의 데이터는 각기 다른 T개의 EMNIST-Letters 이미지와 (T+1)개의 라벨로 구성된다. T개의 이미지가 어떤 문자인지에 대한 정보는 앞 T개의 라벨을 통해 주어지며, 그 T개 이후로 이어져야 하는 한 개의 문자에 대한 정보가 (T+1)번째 라벨을 통해 주어진다. Label은 항상 등차수열을 이루며 그 공차는 -6 이상 6 이하이다

3.2. Model Architecture

Easy Dataset: Easy dataset에 대한 모델은 VGG 모델과 LSTM 모델을 연결하여 설계하였다(Figure 1). VGG 모델의 경우 (28, 28) 크기의 흑백 이미지를 input으로 받아 26 크기의 output을 출력한다. 이 출력은 softmax 레이어를 거쳐서 나오며 이미지가 각 문자일 확률에 대한 벡터이다. LSTM모델의 input은 크기 52 의 벡터, hidden state는 크기 130 의 벡터이다. Output은 hidden state 벡터에 대해 하나의 FC 레이어와 하나의 Softmax 레이어를 거쳐서 나오는 크기 26 의 벡터이다. 이는 각 문자의 다음으로 등장할 확률에 대한 벡터이다. LSTM의 레이어 수는 1 으로 두었다.

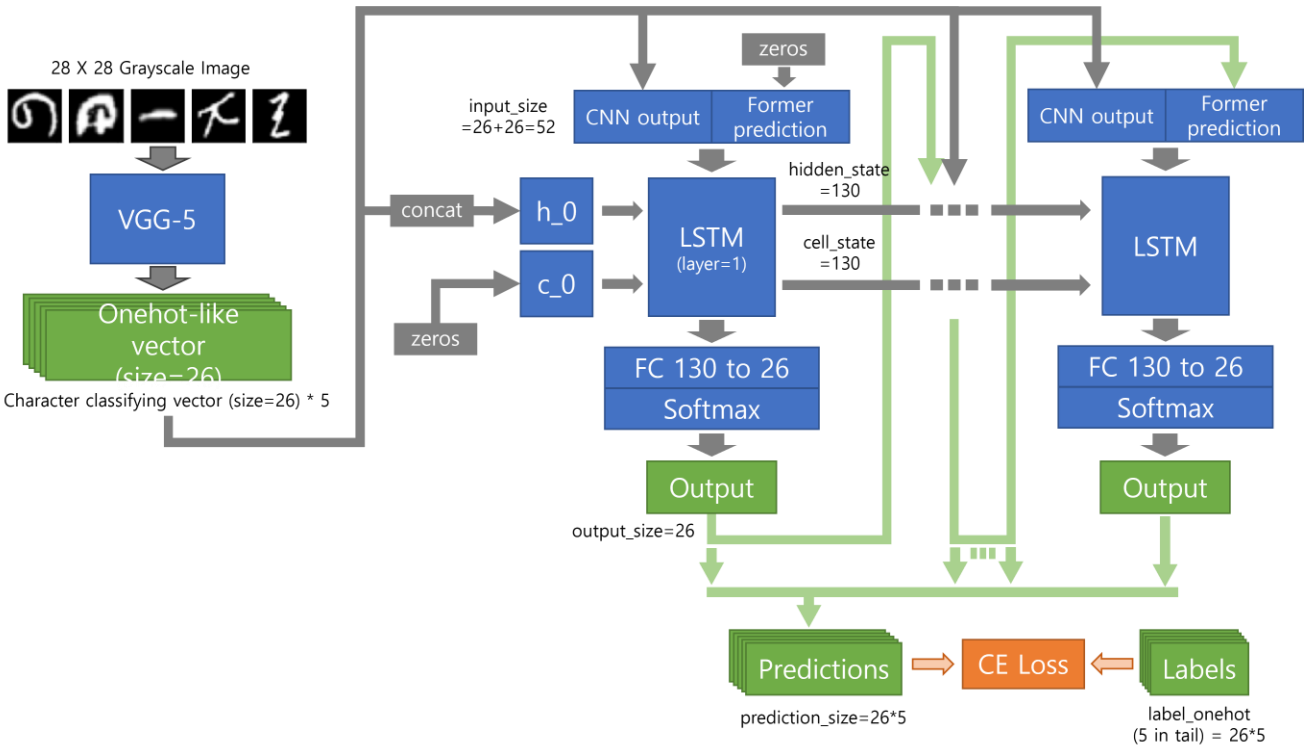

Figure 1: Easy Dataset에 대한 모델 구성

두 모델을 연결하는 과정에서 다음과 같은 기법을 사용하였다. LSTM의 input으로는 VGG에서 얻은 크기 26 의 벡터와, 이전 단계의 LSTM의 출력으로 얻은 크기 26 의 벡터를 이어 붙인 크기 52 의 벡터를 사용한다. 이는 VGG의 판단과 LSTM의 판단 사이의 선택의 자율성을 부여하기 위함이다. 또한 LSTM의 hidden state의 초기값으로 VGG에서 얻은 output 5 개를 이어 붙인 크기 130 의 벡터를 사용한다. 이는 전체 문자열에 대한 정보를 제공하기 위함이다. 이를 위해 LSTM의 hidden state의 크기가 130 으로 설정되었다.

위와 같이 모델을 설계한 뒤, 다음과 같은 조건으로 모델을 학습시켰다. Batch 크기는 32, optimizer는 Adam, learning rate는 0.001 으로 두었다. Epoch는 20 까지 학습시키고 추이를 관찰하였다. Teacher forcing 기법을 적용하여, training 과정에서 LSTM의 input 중 이전 단계의 LSTM 출력을 넣어주는 부분을 label에서 주어지는 값으로 대체하였다. Test 과정에서는 이전 단계에서 예측한 값을 사용한다. 최종적으로 LSTM이 예측한 5 개의 문자와 label의 마지막 5 개 사이의 cross entropy를 loss로 설정하였다.

Normal Dataset: Normal dataset에 대한 모델은 VGG 모델과 LSTM을 연결하는 것은 동일하나 easy dataset에서 시도한 실험적인 방법들은 최소화하였다 (Figure 2). VGG 모델의 경우 (28, 28) 크기의 흑백 이미지를 input으로 받아 256 크기의 feature 벡터를 출력한다. 정형화된 방식이 아닌, 모델이 학습하는 방식대로 이미지에 대한 정보를 표현할 수 있도록 의도하였다. LSTM의 input은 256 크기의 벡터로, VGG에서 얻은 feature 벡터만을 받는다. 이외의 정보는 input으로 사용되지 않는다. LSTM의 hidden, cell state는 크기 256 의 벡터이다. 이는 이미지의 feature 벡터의 크기와는 무관하며 충분히 큰 값을 가지도록 설정한 것이다. 초기값은 모두 0 으로 두었다. LSTM의 레이어 수는 2 로 두었다. 각 단계에서 최종 예측 결과를 얻기 위해 LSTM의 hidden state 벡터를 output size 26 인 fully connected 레이어 하나를 거치도록 하였다.

위와 같이 모델을 설계한 뒤, 다음과 같은 조건으로 모델을 학습시켰다. Batch 크기는 64, optimizer는 Adam, learning rate는 0.001 으로 두었다. Epoch는 30 까지 학습시키고 추이를 관찰하였다. Teacher forcing은 사용하지 않았다. 최종적으로 LSTM이 예측한 마지막 문자 하나와 label이 제공하는 마지막 정답 사이의 cross entropy만을 loss로 설정하였다.

Variations: Normal dataset에 대한 모델을 설계한 뒤, 몇 가지 추가적인 기법을 적용하여 기존 모델과

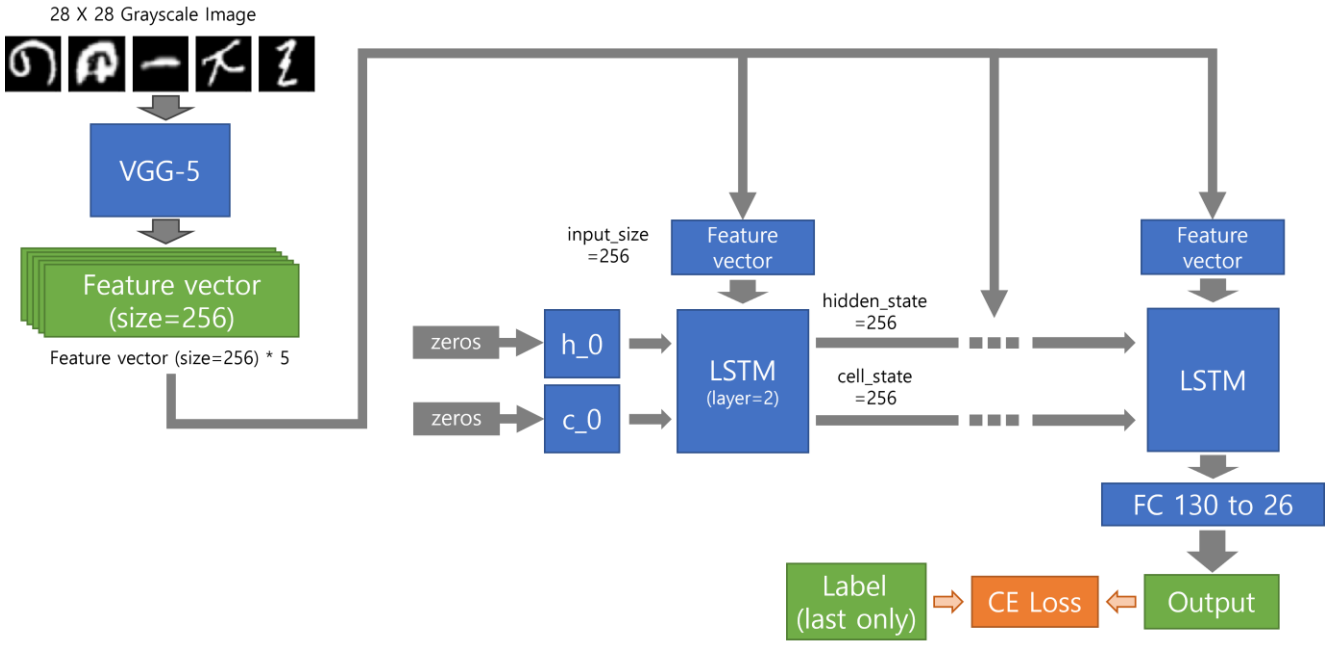

Figure 2: Normal Dataset에 대한 모델 구성

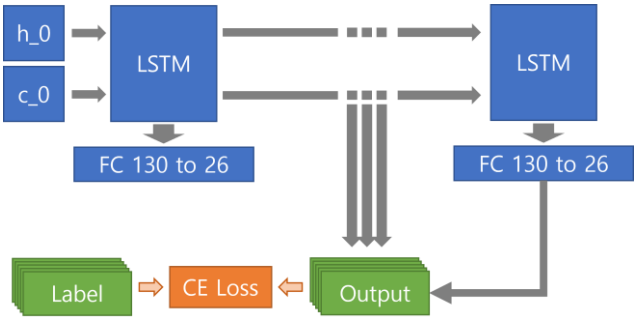

Figure 3: Except-First Loss

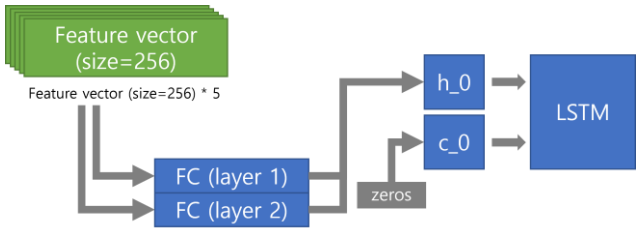

Figure 4: Hidden State Initialization

성능을 비교해보았다. 첫 번째 기법은 except-first loss이다. 기존에는 마지막 예측 결과에 대해서만 loss를 계산한 것과 달리, 첫 번째 예측 결과를 제외한 나머지 예측 결과에 대해서도 차이를 계산하여 loss에 합하였다(Figure 3). 이를 통해 data augumentation 과 비슷한 효과를 가질 것으로 기대하였다.

두 번째 기법은 hidden state initialization이다. LSTM의 hidden state를 VGG에서 얻은 feature 벡터를 사용하여 초기화하는 것으로, 문자열 중 처음 2 개의 feature 벡터를 output 크기 256 의 fully connected 레이어에 각각 통과시켜 LSTM의 첫 번째, 두 번째 레이어의 hidden state로 사용하는 것이다(Figure 4).

4. Results

4.1. Easy Dataset

Epoch 2 부터 train dataset에서는 90%를 넘었으나, test dataset에 대해서는 75~90% 사이에서 큰 폭으로 움직이는 모습을 보여주었다(Figure 5). Test dataset에 대한 최고 정확도는 epoch 5 에서의 88%였다.

4.2. Normal Dataset Approach on Easy Dataset

Easy Dataset을 위해 사용한 모델이 충분한 성능을 보여주지 못하여, normal dataset을 위해 설계한 모델을 그대로 가져와 사용하였다. Epoch 1 에서 train dataset에 대해 91.45%의 정확도를, test dataset에 대해서 98.89%의 높은 정확도를 보여주었다.

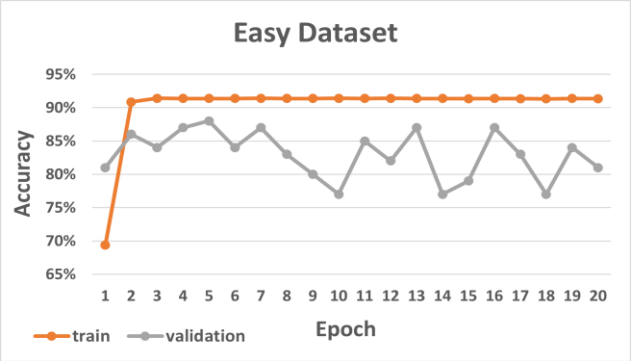

Figure 5: Easy Dataset Benchmark

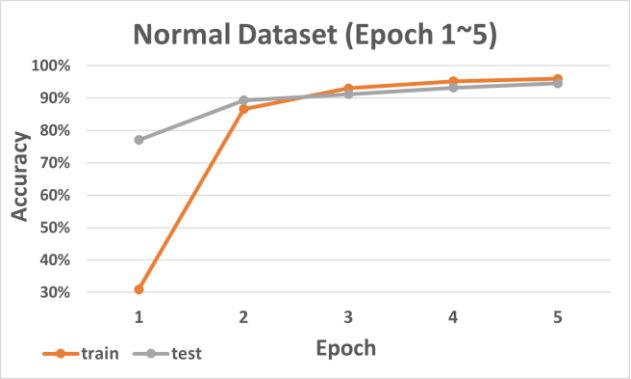

Figure 6: Normal Dataset Benchmark (Epoch 1~5)

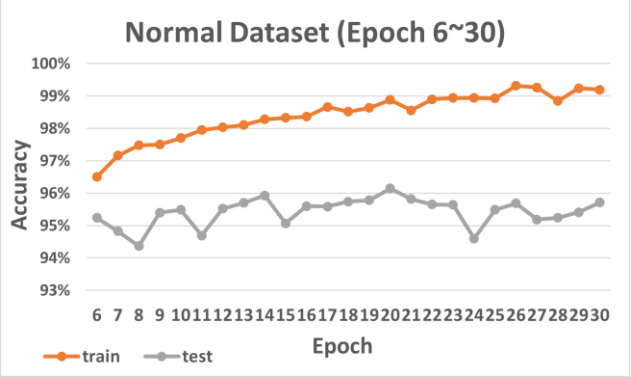

Figure 7: Normal Dataset Benchmark (Epoch 6~30)

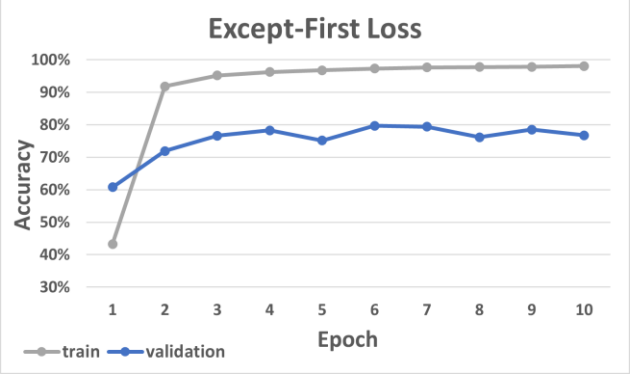

Figure 8: Except-First Loss

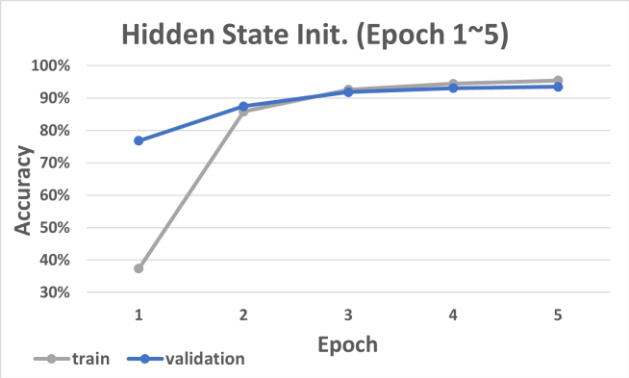

Figure 9: Hidden State Initialize (Epoch 1~5)

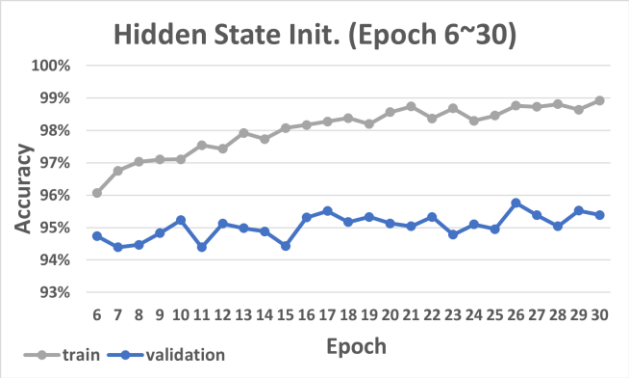

Figure 10: Hidden State Initialize (Epoch

4.3. Normal Dataset

Epoch 3 부터 train dataset과 test dataset 모두 90%의 정확도를 넘기는 빠른 학습의 모습을 보여주었다 (Figure 6). 하지만 epoch 6 이후로는 train dataset에 대한 정확도는 꾸준히 증가하지만, test dataset에 대해서는 큰 진전이 이루어지지 않는 모습을 보여주었다 (Figure 7). Test dataset에 대한 최고 정확도는 epoch 20 에서의 96.15% 였다.

4.4. Variation 1: Except-First Loss

Train dataset에 대해서는 epoch 2 부터 90%를 넘는 정확도를 보여주었으나, Test dataset에 대해서는 80% 정도의 정확도에 수렴하는 모습을 보여주었다(Figure 8).

4.5. Variation 2: Hidden State Initialization

Epoch 3 부터 train dataset과 test dataset 모두 90%의 정확도를 넘기는 모습을 보여주었다(Figure 9). 하지만 epoch 6 이후로는 train dataset에 대한 정확도는 꾸준히 증가하지만, test dataset에 대해서는 큰 진전이 이루어지지 않는 모습을 보여주었다(Figure 10). 또한 Train, Test 모두 기존의 방식보다 정확도가 낮게

측정되었다. Test dataset에 대한 최고 정확도는 epoch 26 에서의 95.76% 였다.

5. Discussion

처음에 구상한 모델은 정확도가 90%에 도달하지 못했다. 이후 Normal Dataset을 위해 설계한 모델을 사용하자 첫 epoch에서부터 굉장히 높은 성능을 보여주었다.

이러한 차이의 원인으로는 VGG의 output을 크기 26 의 벡터로 제한했던 점을 들 수 있다. VGG에서 LSTM으로 넘어가는 정보의 양을 제한할수록 학습의 상황에서 역전파를 통해 학습하는 양도 줄어들 수밖에 없을 것이다. 그러므로 normal dataset을 위한 모델과 같이 정보는 충분히 흘러갈 수 있도록 설계하고, 그 속에서의 연산은 모델의 학습에 맡기는 것이 유리하다고 할 수 있다.

또 다른 원인으로는 LSTM의 input으로 VGG의 output, 이전 단계 LSTM의 output을 이어 붙이고 teacher forcing을 적용한 점도 들 수 있다. Training 단계에서 label의 값이 그대로 LSTM에게 주어지게 되면서 불명확한 VGG의 output보다는 확실한 label을 신뢰하도록 학습될 수 있고 이것이 전체 성능의 하락으로 이어졌을 수 있다. 실제로, 실험 과정에서 easy dataset과 유사한 구조의 LSTM 모델을 normal dataset에 적용시켰을 때 training set에서는 100%에 근접한 정확도를 보여주지만 test set에서는 5%의 매우 낮은 정확도를 보여준 바 있다. 이 때 VGG의 output을 확인해보니 이미지와 무관하게 항상 동일한 output을 출력하도록 학습되었다. 즉, LSTM이 label 부분만을 신뢰하고 VGG는 무시하여 VGG의 학습이 전혀 일어나지 않은 것이다.

이러한 부분이 개선된 normal dataset 모델에서는 충분히 좋은 정확도를 보여주었다. VGG의 output 크기를 256 으로 키워 VGG와 LSTM 사이의 정보 흐름이 원활히 이루어지도록 하였고, LSTM의 input은 오롯이 VGG가 출력한 feature 벡터만을 사용하도록 하였다.

여기서 몇 가지 기법을 추가하여 성능을 향상시키고자 하였다. 첫째로, 기존에 마지막 예측에 대해서만 loss 계산을 했던 것을 수정하여, 공차를 계산할 수 없는 첫 번째 예측을 제외한 나머지 전부의 예측에 대하여 loss 계산을 진행하였다. 하나의 문자열에서 얻을 수 있는 여러 부분 문자열에 대한 예측도 학습시키는, data augmentation의 효과를 노린 것이었으나 오히려 모델의 성능이 감소하였다. 그 이유로는 같은 형태의 data에 대해 너무 많은 loss 계산이 이루어지다 보니 overfitting이 일어난 것으로 추측된다. Train dataset과 test dataset에 대한 정확도의 차이 또한 overfitting으로 설명될 수 있다.

두 번째로 적용한 기법은 LSTM의 hidden state를 initialize 하는 것이었다. 문자열의 처음 두 개의 feature vector를 LSTM의 hidden state 초기값으로 변환시켜 더 많은 정보를 제공하고자 하는 의도였지만, 초기값을 0 으로 주었을 때에 비해 약간의 성능 하락이 관측되었다.

Easy dataset에 대한 모델의 실패와 위의 두 기법을 시도해본 결과로, 충분히 간단한 모델을 사용하는 것이 좋다는 점을 재확인할 수 있었다. 인위적인 의도를 갖고 모델의 형태를 조작하여 성공하기는 쉽지 않아 보인다. 목적에 부합하는 기본적인 구조만을 설계하고 나머지는 데이터와 모델의 학습 과정에 맡기는 것이 가장 좋은 성능을 내는 모델을 얻는 방법이라고 생각된다.

References

[1] SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[2] KABIR, H. M., et al. Spinalnet: Deep neural network with gradual input. arXiv preprint arXiv:2007.03347, 2020.

[3] SAK, Hasim; SENIOR, Andrew W.; BEAUFAYS, Françoise. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.