CNN(convolutional neural network)-GRU(Gated recurrent units) 모델을 활용한 Hand-written letter의 next arithmetic sequence 추측

서울대학교
공과대학
전기정보공학부
2017-18203
서준표

초록

CNN-GRU 아키텍처를 기반으로 teacher forcing, label smoothing을 활용해서 같은 모델로 Easy set과 Noraml set에 학습을 진행했고 각각의 validation set에 대해서 validation accuracy를 99.75 %, 95.25% 으로 확인했다.

1. 서론

이번과제는 아래와 같이 주어진 그림 시퀀스가 해당되는 문자의 인덱스에 대해서 등차수열의 규칙을 만족할 때, 다음 문자로는 어떤 문자가 등장하게 될지를 예측하는 모델을 생성하는 것이 목표라고 할 수 있다. 아래의 예시에서는 MNIST 이미지 sequence 3개가 +1의 등차수열의 규칙을 만족하므로 25(Z)를 예측해야된다.

22 23 24

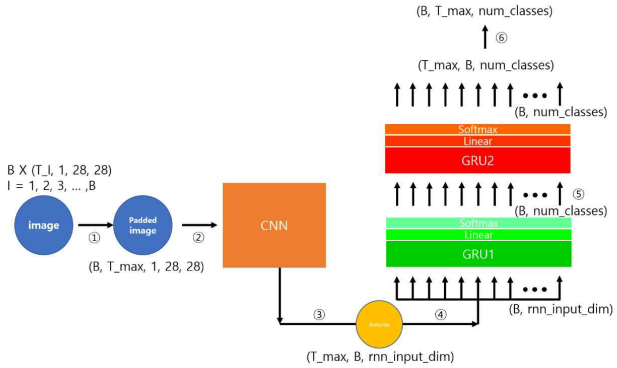
W X Y

easy task와 normal task는 아래와 같은 차이가 있다.

	Easy	Noraml
배경색	검정색	검정색, 하얀색
등차	+1 ~ +3	-6 ~ +6
시퀀스 길이	5개로 고정	두개 이상의 개수
데이터 총 개수	50000	50000

모델은 CNN 와 GRU를 활용해서 encoder-decoder 형식의 모델 아키텍쳐를 구성했다. insight를 sketch하는 과정에서 인간이 다음의 문자를 예측하는 방법론을 도입했다. 인간이 image sequence를 보고 다음에 어떤 문자가 올지 예측하기 위해서는 가장 먼저 주어진 문자들이 어떤 것들인지 파악하는 것이 가장 우선시 되어야한다. 그 뒤엔 문자들의 좌우 관계를 긴밀히 파악해서 마지막 문자를 도출해낸다. 이러한 직관적인 방법론을 활용해서 모델의 구조에 그대로 적용을 해봤다. 가장 먼저 image data를 feeding 시켜 CNN으로 features(CNN)를 뽑아서 이미지가 무엇인지 시각적으로 인지한다. 그 뒤, features의 좌우관계를 살펴서 주어진 문자열들이 무엇을 나타내는지에 대한 정보를 논리적으로 파악(GRU1)한 다. 마지막으로, 파악한 문자열 전부를 다시 한번 feeding시켜 좌우관계를 살피고(GRU2) 다음에 어떤 문자열이 등장할지를 예측하는 구조로 설계를 하였다.

2. 모델 구조



모델의 전체적인 구조는 위의 그림과 같이 구성을 했다. 배치 크기(B)만큼의 텐서 리스트를 입력으로 받고 난 뒤, 배치 크기(B)만큼의 시퀀스 별로 probability distribution을 내뱉는 CNN-GRU 구조를 설계하였다. 세부적인 사항은 아래와 같다.

① B X (T_I, 1, 28, 28) 리스트를 torch 내부의 rnn.pad_sequence 유틸을 활용해서 T_max(T_I 중

최대) 만큼의 차원을 확장시키고 zero padding을 해서 (B, T_max, 1, 28, 28) 텐서를 GPU에 로딩한다 이는 GPU를 십분 활용해서 모델이 training time을 줄이기 위함이다.

② padding 된 tensor를 dim=(0,1) 방향으로 직렬화해서 (B * T_max, 1, 28, 28) 텐서로 reshape를 한다. 이렇게 생성된 텐서를 여러 convolutional layer로 구성된 CNN을 통과시켜 features를 추출한다. 이때의 생성된 features는 다시 dim=(0,1) 방향으로 병렬화해서 (B, T_max, rnn_input_dim) 텐서로 reshape한다.

③ GRU1 layer에 통과시키기 위해서 dim=(0,1)를 transpose하여 (T_max, B, rnn_input_dim)로 features의 shape을 바꾼다.

④ Features를 GRU1, linear, softmax layer 에 통과시켜서 num_class만큼의 probability distribution을 얻는다. 이때, (T_max, B, rnn_input_dim) tensor를 feeding하여 (T_max, B, num_classes=26) tensor를 생성했다. 여기서의 distribution은 CNN-GRU1이 주어진 문자열을 보고 생성한 문자들의 probablity distribution이다. (그림에서 화살표의 개수는 T_max개 이다.)

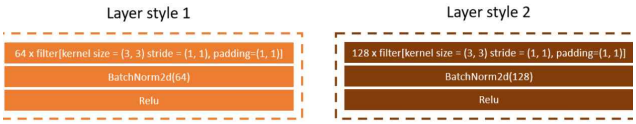
⑤ GRU1 layer에서 probablity distribution를 다시 한번 GRU2 layer에 통과시켜 다음 sequence들에 대해서 probability distribution을 얻는다. 이때, (T_max, B, num_classes) tensor를 feeding하여 (T_max, B, num_classes) tensor를 생성했다.

⑥ GRU2 layer에서 나온 (T_max, B, num_classes)을 (B, T_max, num_classes)으로 transpose한다.

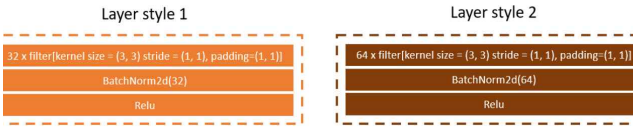
앞에서 언급한 insight sketch의 견지에서 살펴보면 GRU1 layer의 output은 시력에 해당하고 GRU2 layer의 output은 사고력(수열의 등차를 추론해내고 다음 문자열을 출력)에 해당한다. 등차의 값이 음이 될 수도, 양이 될 수도 있으므로 bidirecitonal GRU unit을 활용했다. 또한, 경험적으로 lstm보다 GRU가 learning의 속도나 generalization에서 advantages가 있어서 GRU를 RNN layer로 채택했다.

한편, CNN에서의 gradient vanishing 문제를 최소화를 위해서 resnet 기반의 skip connection을 응용했다. skip connection을 구현할 때는 각 레이어의 끝단에서 연산이 완료된 값을 세 블락(resnet 기준) 뒤의 의 Relu의 input과 연결을 시켰다. conv layer은 3 X 3만을 구현하였고 stride와 padding을 적절히 선택해서 dimension이 유지되도록 했으며 max_pool layer를 통과하고 filter의 차원이 절반으로 줄어들고 난 뒤에는 filter의 개수를 늘려 features를 잘 추출할 수 있도록 설계하였다. Easy task, Normal task에서의 전체적인 아키텍처는 완전히 동일하다. 하지만, CNN의 filter size와 rnn_input_dim에서 차이가 있다. CNN의 세부적인 구조는 아래 그림과 같다.

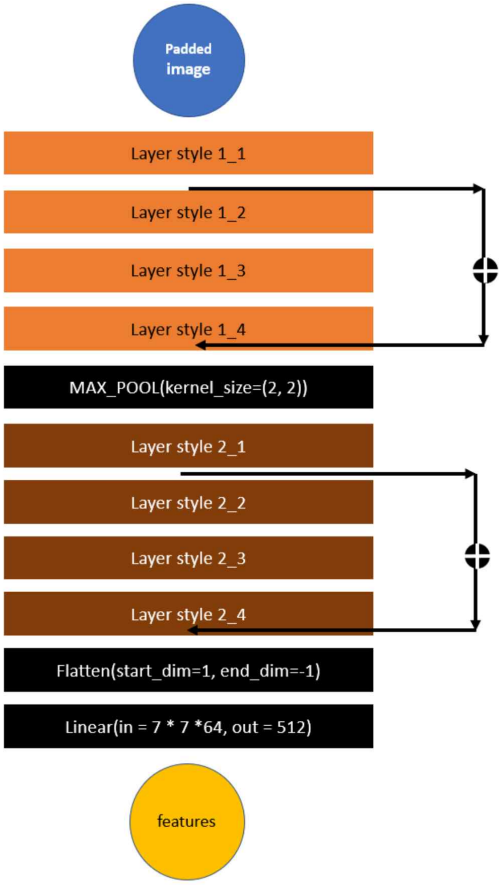
- CNN for easy task



- CNN for normal task

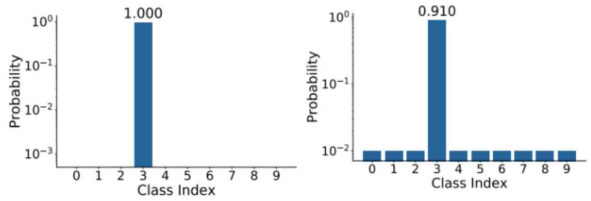


- CNN whole architecture for both

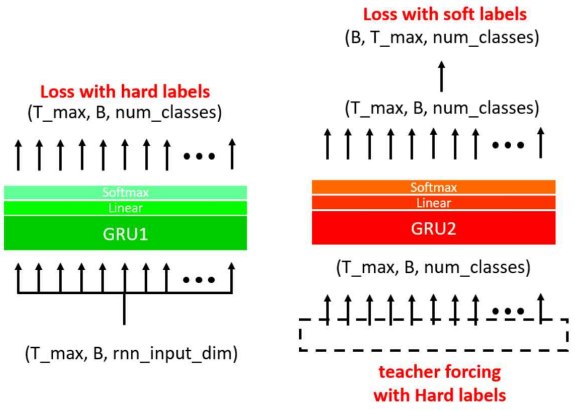


3. 학습

모델의 학습을 위해서 GRU2 layer에서 ground truth value를 hard label를 assign 하였고 GRU2에서 ground truth value를 soft label(label smoothing을 활용)으로 assign했으며 hard label로 teacher forcing을 했다. label smoothing을 통해 normal task에서 보다 더 validation accuracy가 높았으며 generalization을 hard label보다 더욱 잘 할 수 있는 특징을 엿볼 수 있었다. soft label을 통해서 probability distribution의 noise에 대한 고려가 충분히 되어 이와 같은 결과를 얻을 수 있었다. label smoothing은 one_hot 텐서를 기준으로 soft_label = one_hot * (1 - 0.2) + (0.2 / 26)으로 assign 하였다.



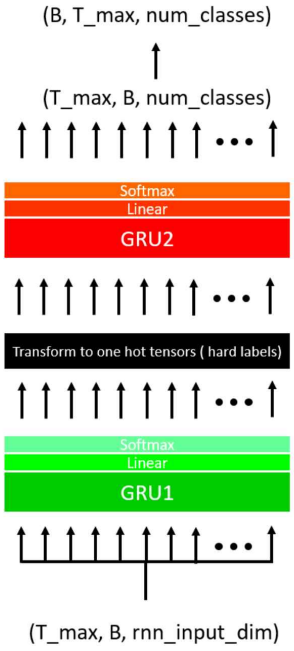
[hard label과 soft label]



loss function으로는 mean squared error loss function을 활용했다. GRU2에서만 soft label을 활용한 이유는 경험적으로 GRU2에서 수열의 추론이 과적합 되었다고 판단했기 때문이다. 최종 loss의 값은 GRU1에서와 GRU2에서의 loss 합의 평균이다.

4. 추론

CNN-GRU1 layer의 학습을 hard margin으로 진행하였으므로 추론시에도 CNN-GRU1 layer의 output을 one-hot tensor로 바꾸어준 뒤에 다음 레이어로 feeding 한다. 그 과정은 아래 그림과 같다.



5. 하이퍼파라메터 튜닝

```
kwargs = {
    'cnn_input_dim': 1,
    'cnn_hidden_size': 32,
    'rnn_input_dim': 512,
    'rnn_hidden_size': 256,
    'rnn_num_layers': 2,
    'rnn_dropout': 0.00
}
```

모델의 구조와 optimizer, loss function, learning rate, batch size을 hyperparameter 로 설정하였고 learning의 속도가 느리면 learning rate 값을 튜닝하였고 overfitting이 발생하면 모델의 구조를 좀 더 복잡하게 만들기위해 rnn_hidden_size 혹은 rnn_num_layers를 더 높은 값으로 설정하며 튜닝했다.

```
del_optim = torch.optim.Adam(model.parameters(), lr=0.000
ss_func = nn.MSELoss()
loss_func = nn.CrossEntropyLoss()
```

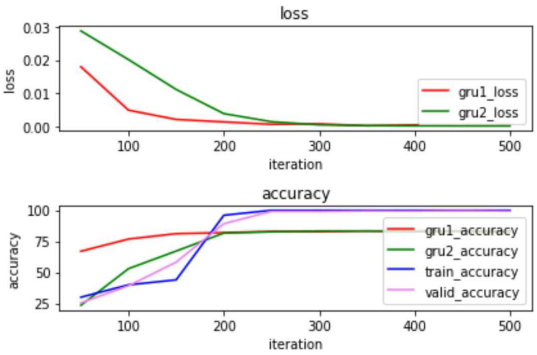
batch size는 easy는 100에서 normal은 40에서 훈련을 시킨 결과가 가장 훌륭하게 나왔다. easy는 set의 variation이 크기 않아서 batch size를 크게 잡아도 훈련이 잘 되는 반면 normal은 T_I 값이 배치의 element마다 다르기 때문에 작게 설정해줘야 모델이 더 쉽게 learning 할 수 있는 것으로 보인다.

6. 결과

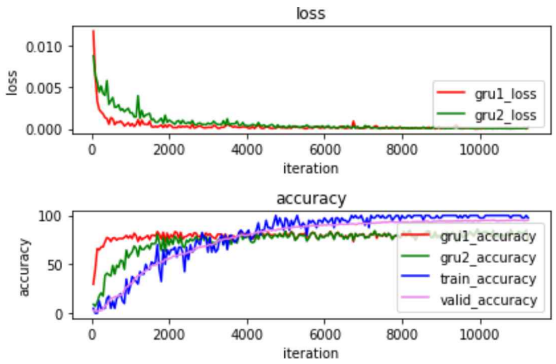
최종 모델의 결과분석을 위해서 정의한 변수는 아래와 같다.

① gru1_loss: ground truth hard label(train set)과 gru1의 loss
② gru2_loss: ground truth soft label(train set)과 gru2의 loss
③ gru1_accuracy: ground truth seqeunce(train set)와 gru1 layer에서 max value를 취한 estimated sequence에 대한 accuracy
④ gru2_accuracy: ground truth seqeunce(train set)와 gru1 layer에서 max value를 취한 estimated sequence에 대한 accuracy
⑤ train_accuracy: ground truth seqeunce(train set)의 가장 마지막 value와 gru2 layer에서 max value를 취한 estimated sequence의 가장 마지막 value에 대한 accuracy
⑥ valid_accuracy: ground truth seqeunce(valid set)의 가장 마지막 value와 gru2 layer에서 max value를 취한 estimated sequence의 가장 마지막 value에 대한 accuracy

(①, ②, ③, ④, ⑤은 print 할때의 batch에 대해서만 계산했다.)



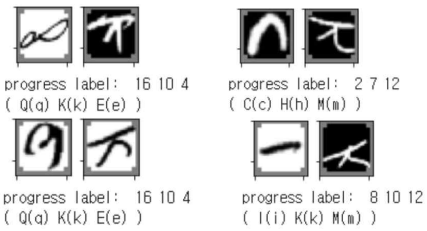
[Easy task]



[Normal task]

최종적으로 easy와 normal task 에서 validation accuracy를 99.75 %, 95.25%로 확인 할 수 있었다. gru1_accuracy(시력)와 gru2_accuracy(사고력)가 높아질수록 train과 valid accuracy가 높아졌다 gru1과 gru2중 하나의 accuracy만 높다면 valid accuracy는 결국 낮아지는 현상을 관찰했다. 이는 곧 시력은 좋지 않지만 사고력이 좋다거나 사고력은 좋지만 시력이 좋지않아서 발생하는 문제인 것 이다. 이를 잘 활용해서 learning rate을 설정할 때도 두개의 accuracy가 모두 비등하게 증가하도록 시행착오를 거쳐 설정하였다. Normal task를 진행할때는 특히 overfitting의 issue가 많이 관찰되었는데 이를 해결하기위해 soft label가 도입된 것이다. 결과적으로 noise에도 generalize를 더 잘하도록 모델을 재구성했다.

7. 발전 방향 제시



Normal task의 validation set에서 Model이 올바르지 않게 예측한 데이터들 중 4개를 가져왔다. 틀리게 예측한 대부분의 데이터는 시퀀스의 길이가 적다는 특징이 있었고 그와 더불어 K가 위와 같이 회전되어 있거나 다른 문자(여기에서는 H)와 헷갈리게 그려져있음을 확인했다. 따라서 더욱 validation accuracy를 높이기 위해선 시각 능력에 해당하는 gru1의 개선이 필요하다. 실제로 test set과 validation set의 가장 다른점은 sequence의 규칙이아니라 sequence image들이 모호하고 조금씩 회전된 데이터들이라는 것이다. 따라서 data

augmentation을 통해서 data의 일부를 random하게 회전시킨 뒤 몇차례 훈련을 하고 나면 더욱 성능이 향상되어있을 것으로 기대한다. 또한 label smoothing을 할때 noise value를 좀 더 크게 assign하여 generalization 능력에 더 큰 비중을 두는 것도 하나의 방법이라고 할 수 있겠다.

8. 참고 문헌

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. Deep Residual Learning for Image Recognition.

[2] Rafael Müller, Simon Kornblith and Geoffrey Hinton. When Does Label Smoothing Help? NeurIPS 2019.