<https://gnoej671.tistory.com/10?category=1034944>

**From RNNs to CNNs**

RNN 대신 CNN을 쓰는 이유

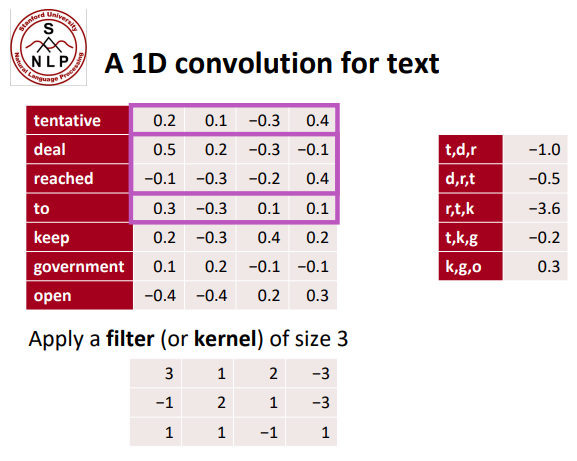
1. RNN은 중간의 단어, 구문에 대해서 별개로 인식 불가 (항상 이전에 나온 문맥이 포함되어야 한다, BiLSTM도 마찬가지) = 이전 prefix가 항상 있다는 전제하에 중간 phrase를 캡처해야 한다.
2. 따라서 final vector에 대한 정보를 가져오기 위해서 이전의 너무 많은 구문을 거쳐야 한다.

Ex) Softmax는 주로 마지막 hidden state의 output에 대하여 진행하곤 한다. = 그전까지의 정보를 다 거쳐야 한다. // sentence classification, sentiment analysis 같은 경우 몇 개의 key가 되는 단어로 분류 가능 => 쓸데 없이 많은 단어 포함 안해도 된다.

따라서 CNN의 경우 모든 n-gram에 대하여 독립적으로 vector를 만드는 것이 메인 아이디어!

Ex) “tentative deal reached to keep government open” ->  
(tentative deal reached), (deal reached to), … , (keep government open)

Convolution 이란 계산 방법은 잘 알고 있을 것이다.(filter로 slide 해가면서 weighted sum)!!



Padding을 사용하면 뭐가 좋을까?

Input의 dimension을 유지시킬 수 있다. = fixed input dimension! 보통 zero pad를 사용한다.

Padding을 심지어 2로하면 output 결과는 input dimension 보다 커질 것이다.(wide convolution)

또한 channel 수를 늘리면 그 수만큼 필터가 필요하다. (채널 수를 늘리면 좋은 이유?? 다양한 정보를 다양한 필터에 대해 추출할 수 있나..? 다시보기)

그리고 N-gram에서 N의 값에 따라 filter 크기가 정해진다.(행의 개수)

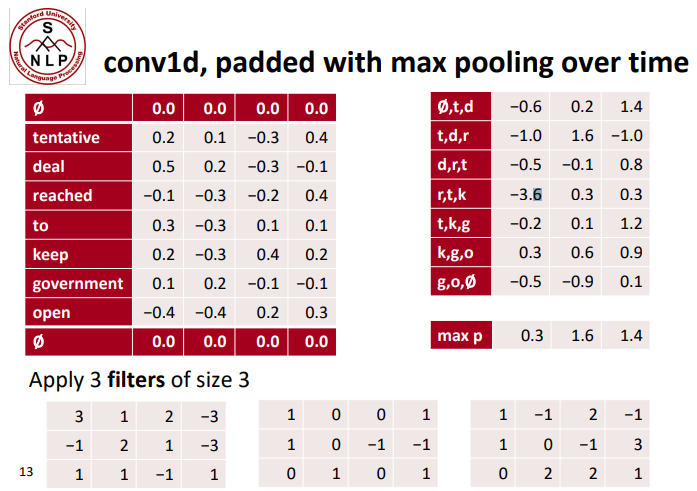
Pooling layer는 가장 두드러지는 특징을 뽑아낸다. = 모든 n-gram이 하나의 값의 벡터로 표현됨. Max pooling은 그렇고 avg pooling은 또 다른 특징을 가진다. (avg pooling의 장점?? 데이터들의 정보를 전체적으로 반영할 수 있나..? 다시보기)

그리고 이 벡터를 fc layer를 거쳐 원하는 label을 얻게 되면 끝

참고로 여기서 파라미터는 filters 이다. 또한 conv나 pool 은 이전 결과를 점차 줄여가면서 정보를 추출해간다 라는 느낌이다.

또한 이전 정보를 더 많이 포함하고 싶다면(더 높은 n-gram = 더 큰 filter)를 이용한다. CNN을 이용하면 sentence classification엔 강할 수 있다.(하지만 semantic 적인 부분에서는 조금 약할 수 있다.)

추가적으로 필터 한 개는 하나의 pattern을 잡는다고 한다.(a 필터는 기쁨, b 필터는 슬픔 …) 따라서 어떤 벡터가 a 필터를 거치고나서 큰 값을 도출한다면 그 어떤 벡터는 기쁨을 나타내는 단어일 확률이 크다. (cosine similarity, inner product 가 크다.) 따라서 이것은 채널을 늘리면 필터 수가 증가한다. 그리고 max pool을 하면 결과에 대해 최댓값만 반환하니 “이 부분이 중요해” 라고 뒷 레이어에 signal을 보내게 된다.



Less useful 하지만 이런 기법도 있다~

1. Stride = 2
2. Stride = 1 & local pooling(pooling stride = 2)

Less useful 이라고는 안했지만 이런 기법도 있다.

1. K-max pooling(stride = 1, k개 만큼 그냥 찾은 순서대로 추출)

Somewhat useful 한 기법도 있다.

1. Dilation (pooling 대신에 filters 도입 한번 더해서 소규모 conv 한번 더 하기)

하지만 모두 굳이 사용할 필요는 없습니다.

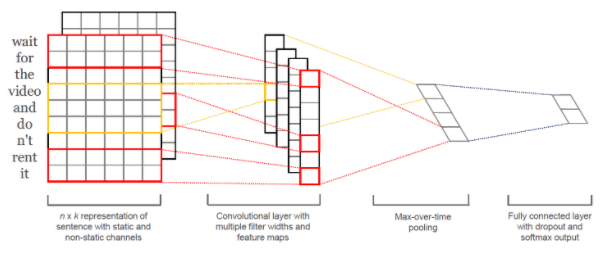
Yoon Kim의 CNN for Sentence Classification 이 있습니다.  
sentiment classification, sub or objective classification, question 종류 classification.

<https://reniew.github.io/26/>

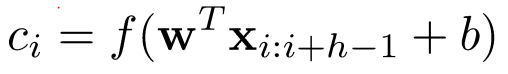
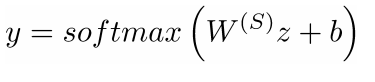
**Yoon Kim의 CNN for NLP (2014)**

문장에서 Sequential 을 중요하게 여기면서 처리하는 것이 중요 = RNN은 이것에 강함.  
그렇다고 CNN이 그 특성을 살리지 못하는 것은 아님.

앞서 말했듯이 RNN은 단어 입력 값을 순서대로 처리하고  
CNN은 n-gram 처럼 n에 따라 그 지역마다의 정보는 반영하면서 (마치 word2vec window 마냥) 어느정도 sequential을 반영할 수 있음. (문장 지역정보 보존, 단어의 등장 순서를 학습에 반영한다.)



이 그림에서 flow 설명(오로지 간단히 one conv and one pooling use)

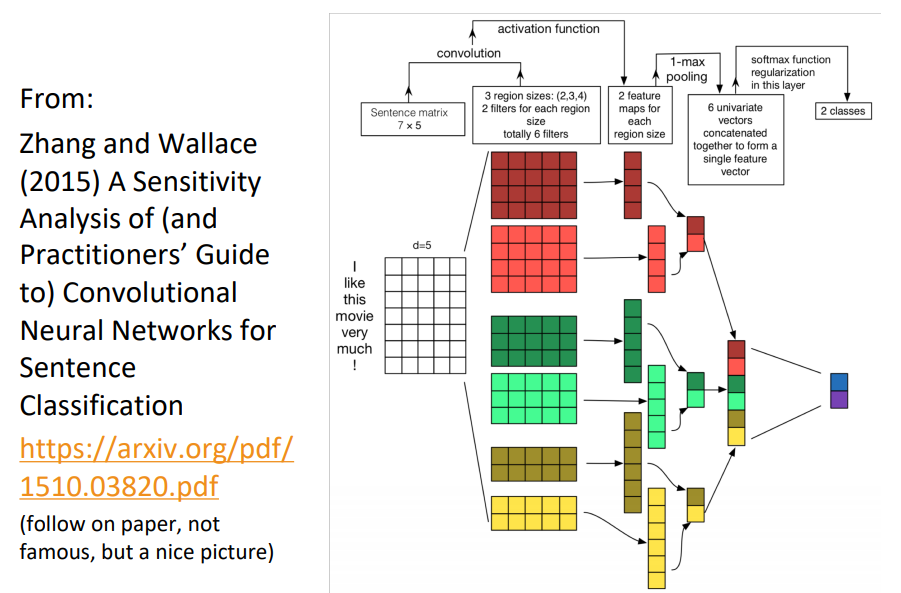
1. Word embeddig 후 concatenation (필요 시 padding)   
     
   
2. 필터는 4개(각 사이즈가 다르다, bi-gram이냐 tri-gram이냐에 따라  
   
3. Feature map이 생성되고 maxpooling 을 거쳐 중요한 feature만을 선택(penultimate layer)  
   
4. Fc layer를 거쳐 긍정 or 부정의 2 classification 적용  
   

**Multi-channel input idea**

Word vectorization을 할 때 2번을 한다. (처음에는 그러면 똑같은 거 2개겠지)  
그 후 한 개는 고정하고(static), 나머지 한 개는 fine-tune 시킴.

이렇게 되면 전체 계산을 2번하게 되지만 정확도는 올라간다고 한다.

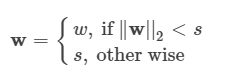
그 값을 종합할 때는 maxpooling 들어가기전에 c값에서 합치고 들어간다.(not concat, add ok)

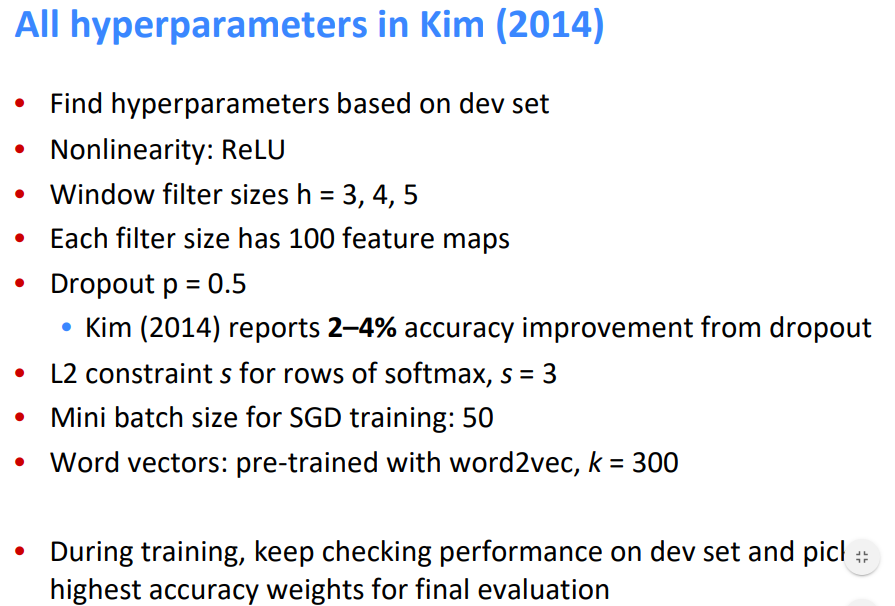


이라는 사람의 버전도 있습니다.

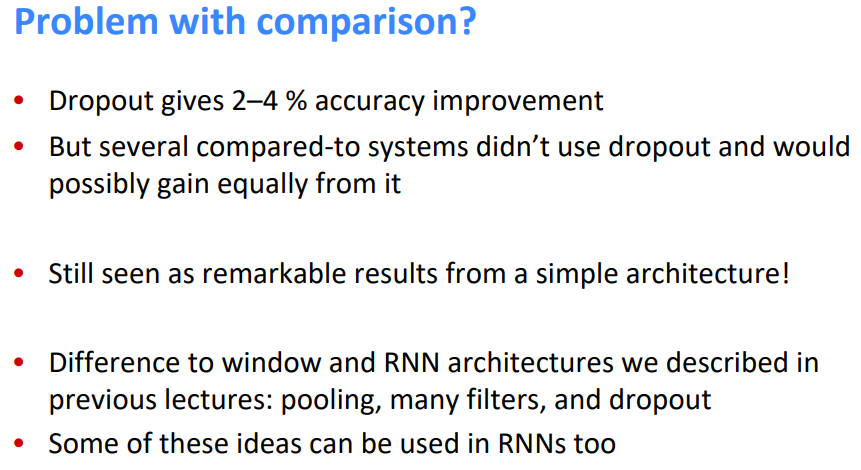
Regularization(일반화 향상, 웨이트 값 낮추기<l2-norm> or 아예 업데이트 안 시키기<dropout>)

(r = 베르누이 분포, 0 or 1, masking 작업, 연산은 아다마르 곱을 의미), training 시에만 적용, 테스트에는 사용 안함. => 일정 웨이트는 업데이트 안되어 정규화 기능 작용 => 단 penultimate layer -> last layer 에만 적용

l2-norm w가 너무 크다면 s(어떤 상수 값, 하이퍼 파라미터)으로 제한한다. (아마 마지막 loss function 시 모든 파라미터에 대해 일것임..?)

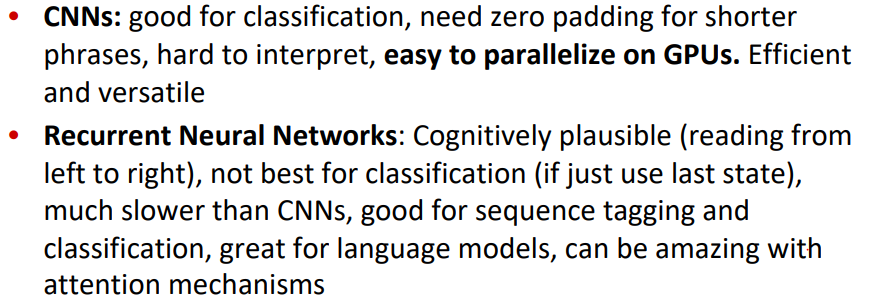


학습 방법임.

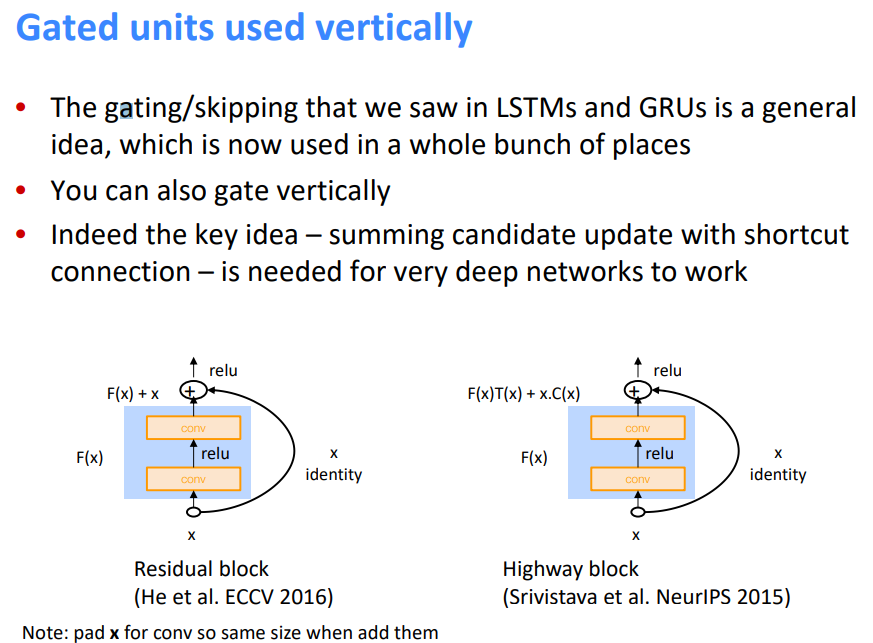


Dropout이 다른 모델에 쓰일 수 있듯이 여러 방법은 서로에게 도움을 줄 수 있음(논문을 작성하는데 필요한 방법,,, 이거는 약간 팁 같이 설명했음)

그러니 여러 실험과 같은 과정이 필요한 것이다. (우리는 아직 여러 부분을 공부하는 단계)



RNNs: cognitively plausible but not best for classification, slower than cnn, 전체적인 문맥 정보를 원한다.  
CNN: good for classification(only last state need), fast, easy to parallelize on GPUs, 지역적인 문맥 정보를 catch한다.



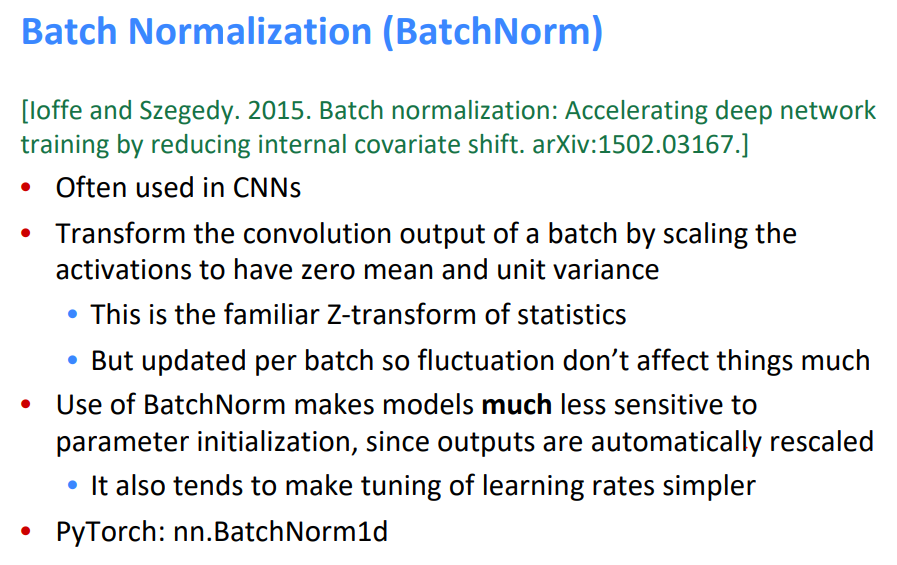
LSTM or GRU의 Gating(마치 Skip-connection, 필요 없다고 생각될 때는 이전 정보를 잊은 채로 진행한다.)

한 layer(conv-relu-conv) 과정을 무시한 채 다음 layer로 그대로 identity 하게 넘겨준다. 그리고 나서 relu를 거치게 된다. 이 과정이 하나의 residual block이라 하며 resnet에서 쓴 것으로 유명하다.

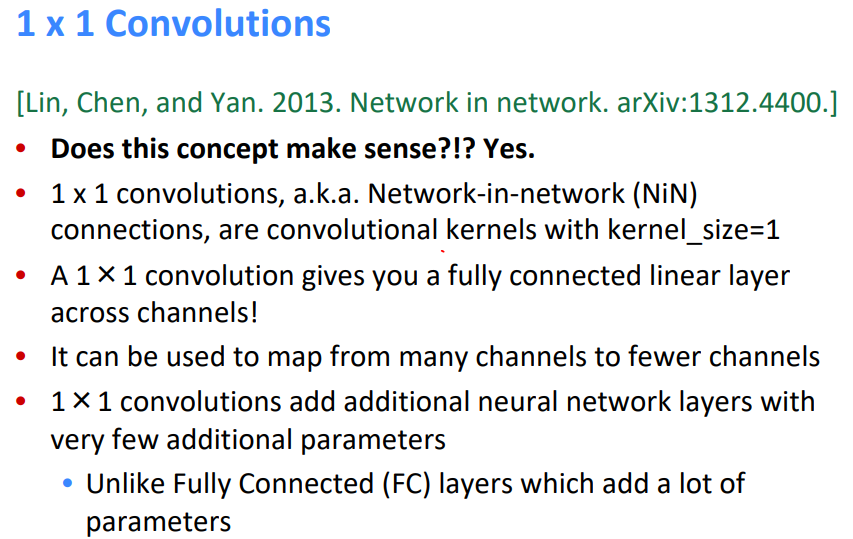
오른쪽 것이 좀 더 명시적인데 F(x)에는 T(x)의 forget gate, C(x)의 input gate가 있다.  
좀 더 LSTMish 하다.

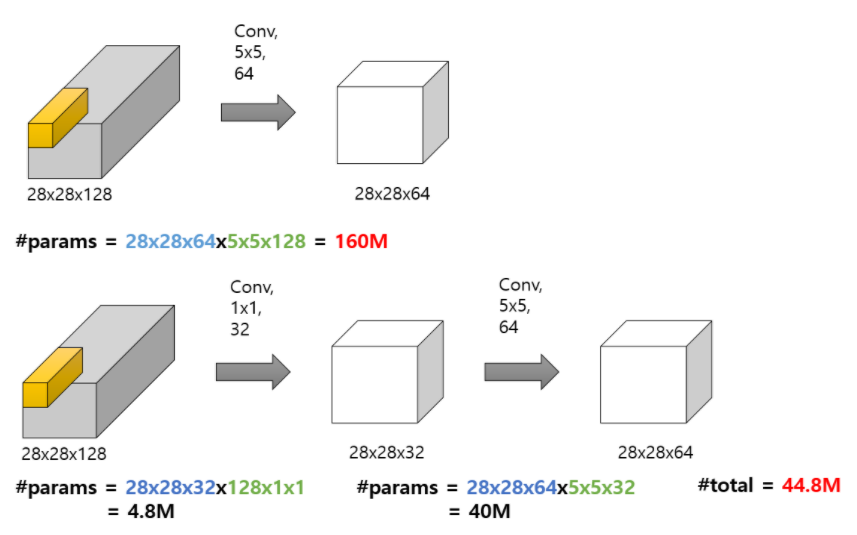
ResNet의 경우 skip or apply(0 or 1) 한 개이고, 오른쪽의 경우 둘 다 어느정도의 비율로 반영한다.

이 방법은 network가 깊을수록 좋은 영향을 끼친다.



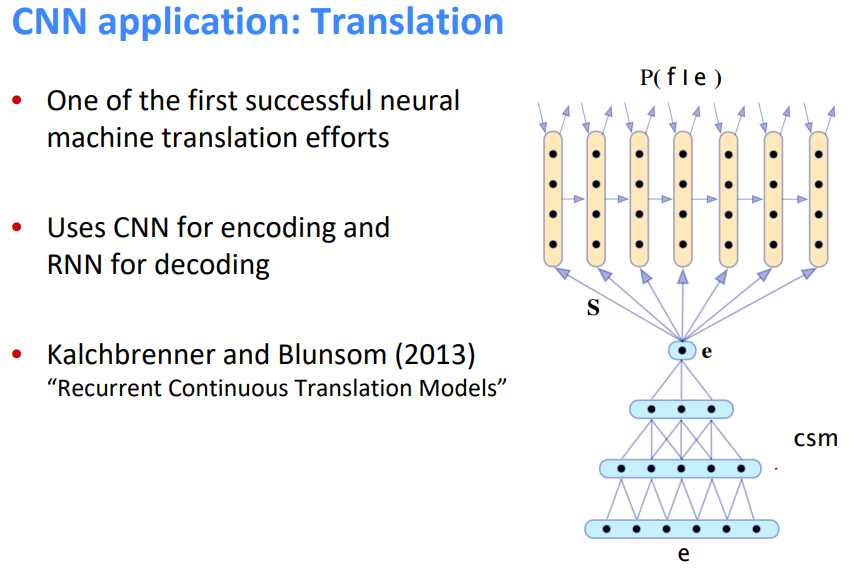
레이어를 거친 후 다시 한번 정규화(z-transform 마냥) => data scale을 재조정 => 거의 요즘 deep network에는 사용한다.



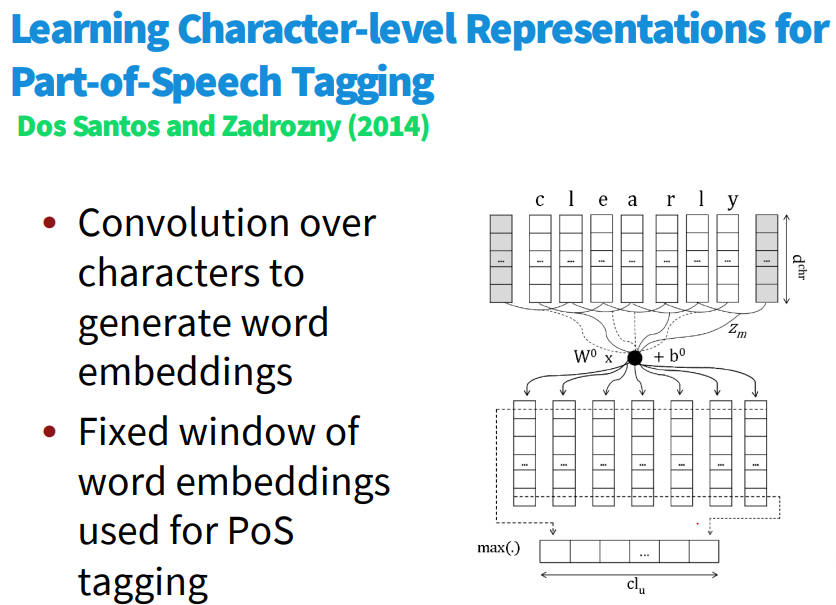


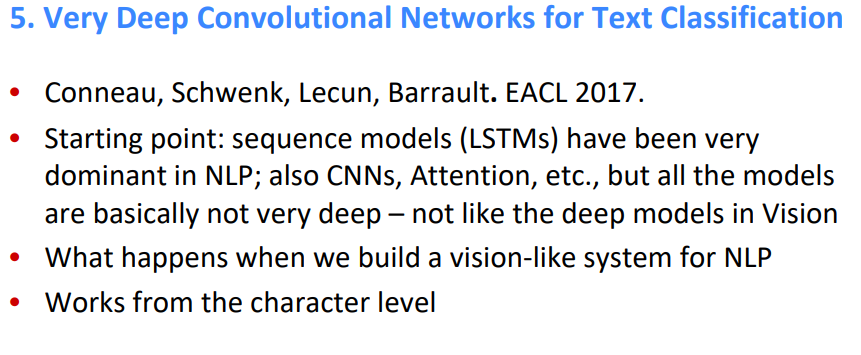
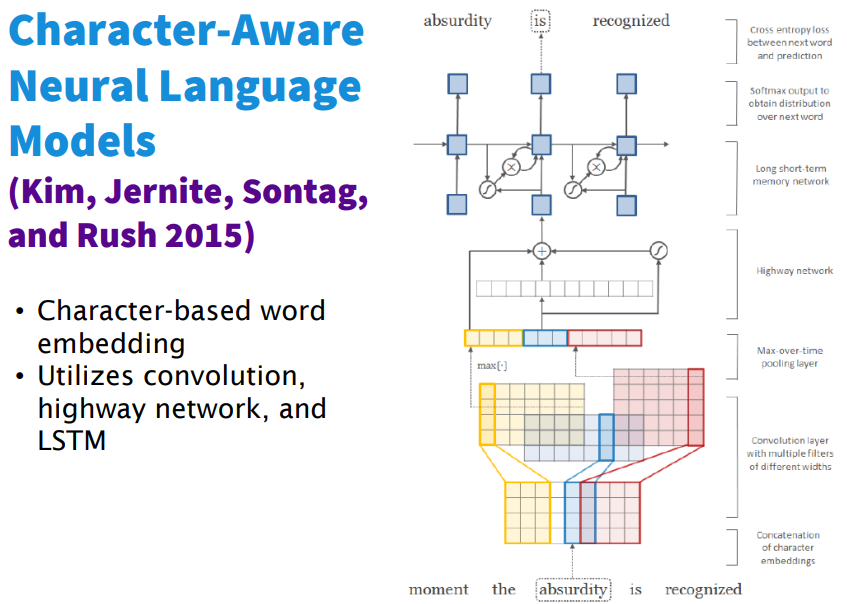
<https://hwiyong.tistory.com/45>

1. 채널 수 조절( size를 유지하면서 channel 수를 낮춰 후에 계산 파라미터 수를 낮출 수 있다.
2. 따라서 fc layer 대신에 1x1 conv로 output을 낸다면 파라미터 수를 낮춰 계산 속도를 빠르게 하면서 같은 성능을 낼 수 있다..?! (마지막 1x10 classification layer 대신에 1x1x10을 도출하도록 한다.)



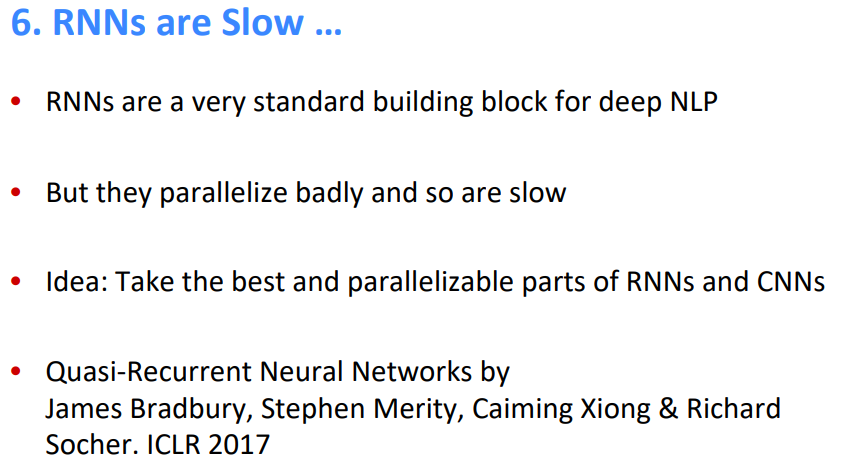
또한 인코더를 cnn으로 디코더를 rnn으로 사용하는 경우도 있다.(디코더를 cnn으로 사용할 수 있지만 거의 사용 안한다. In translation task(seq2seq2))

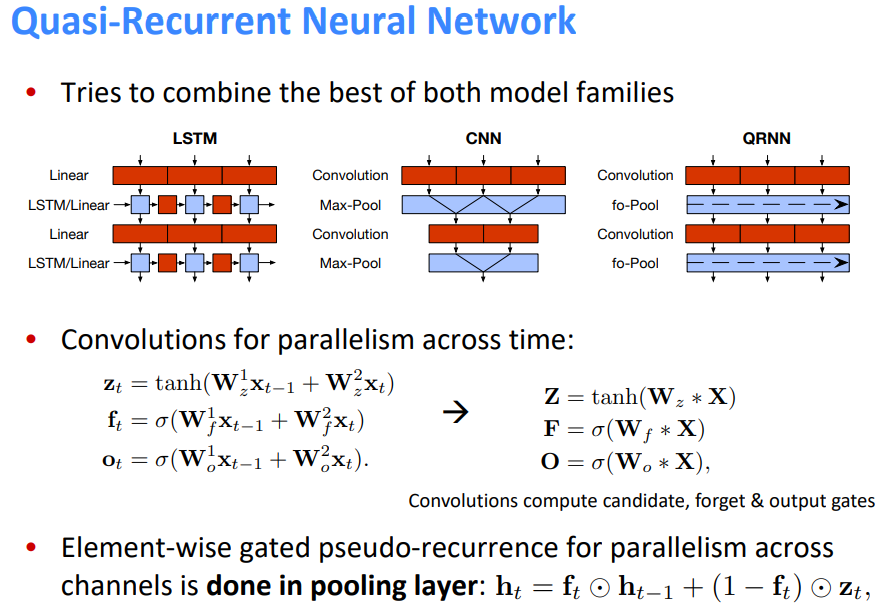




또한 cnn + characters 기반의 여러 task에 대한 모델이 있다.

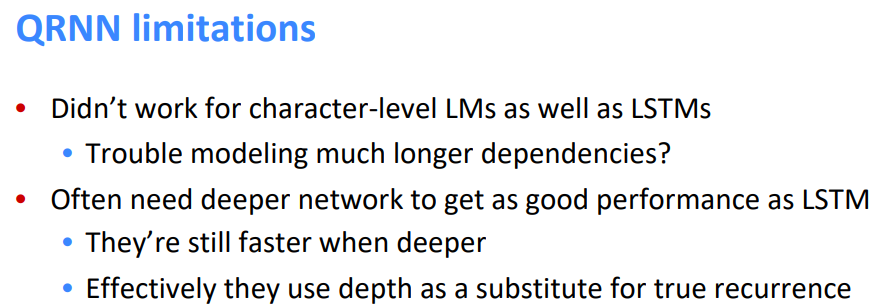
마지막인 VD-CNN에 대한 설명이 있지만 패스하겠다.





처럼 느리다. 병렬 불가..? 왜 느린지 다시 보기 그거 해결한게 Quasi-RNN(2017)??

계산 과정을 한번에?! (기존 LSTM(게이트 별 계산) 대신 Convolution 계산?? 행렬 계산?? 을 이용해 한번에??) => parallelization 가능 (더 알아보기) (인코더 부분이든 디코더 부분이든)



하지만 parallelization 하는데 있어서 모델 구조가 깊어질수록 한계가 있다.(RNN은 주로 어텐션을 이용하고 long-dependency를 항상 중요하게 여기기 때문이다.) = QRNN으로 parallelization in RNN을 했음에도 문제가 발생(아직은 그래도 RNN 기반에서 작동하기 때문)

그러면 attention을 할 때 꼭 RNN을 사용할 필요가 있을까..?(디코더는 제외하더라도 인코더 부분에서 꼭..?) => 트랜스포머의 동기