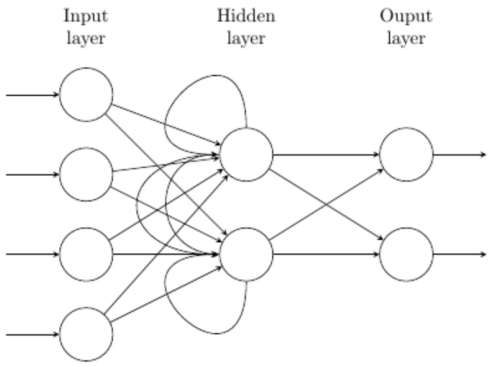
1주차

4-1. 순환신경망(RNN)



- 자연어 처리 문제에 주로 사용되는 인공신경망 구조
- 시계열 데이터를 다루기에 최적화된 인공신경망
- ANN구조에 이전시간(t-1)의 은닉층의 출력값을 다음시간(t)에 은닉층의 입력값으로 다지 집어넣는 경로 추가한 것
- 현재 시간 t의 결과가 다음 시간 t+1에 영향을 미치고, 다시 다음시간 t+2에 영향을 미치는 과정의 반복
- x_i^t (시간이 t일 때 i 번째 입력데이터) 에 가중치를 곱하고, 이전시간의 히든유닛 활성화값과 순환가중치의 곱의 합으로 a_h^t(히든레이어의 출력값)을 나타냄 → a_h^t 에 활성함수를 씌워 히든유닛 활성화 값 b_h^t 계산.

$$a_h^t = \sum_{i=1}^{I} w_{ih} x_i^t + \sum_{h\prime=1}^{H} w_{h\prime h} b_{h\prime}^{t-1}$$

- 장점 : 이전상태에 대한 정보를 일종의 메모리 형태로 저장할 수 있다(시계열 데이터에 유리)

4-2. 경사도 사라짐 문제(Vanishing Gradient Problem)

- 시간축에 따라 시계열 데이터를 지속적으로 받기 때문에 time step 이 진행됨에 따라 과거 데이터의 영향력이 점진적으로 사라짐
- 문제점 : 장기 기억력을 가지지 못함 → LSTM 으로 해결
- Long-Term-Memory-Networks (장/단기 기억 네트워크)
- LSTM 은 은닉층의 각각의 노드를 인풋게이트, 포겟게이트, 아웃풋게이트로 구성된 메모리블럭으로 구성
  - 인풋게이트 : 인풋노드, 이전시간메모리블럭, 셀스테이트 의 합
  - 포겟게이트 : 인풋노드, 이전시간메모리블럭, 셀스테이트 의 합
  - 블럭인풋게이트 : 메모리 블럭의 입력값에 활성화 함수 씌워주는 게이트
  - cell-state : 이전시간 셀 출력값과 현재시간 포겟 게이터의 출력값을 곱한 값에 현재시간 t의 블럭 인풋게이트의 출력값에 인풋게이트 출력값을 곱한 후 더함
- 셀에서 일어나는 연산이 인풋게이트와 포겟게이트를 열고 닫아서 현재 시간과 이전 시간의 데이터의 영향력을 가져갈지 말지를 결정함
- 단점 : 기존 RNN보다 연산량이 많다 → 간략화 버전인 GRU(리셋게이트, 업데이트게이트)는 게이트를 3개에서 2개로 축소

- GRU는 LSTM에서 출력 게이트를 제거, 인풋게이트와 포겟게이트의 역할을 리셋/업데이트 게이트로 나눔. 컴퓨팅 리소스가 충분하지 않은 경우 GRU 사용하는 것 좋을 수 있음

4-3. 임베딩

- 머신러닝 알고리즘을 사용할 때, 특히 자연어 처리 문제 다룰 때 사용되는 기법
- one-hot-encoding 은 표현 형태가 sparse(희박) 한 문제가 있음
- 임베딩 : 원핫인코딩 데이터 표현을 dense 한 형태로 변환하는 기법, 임베딩 행렬을 곱해서 데이터의 표현함, 차원을 줄여줄 수 있음
- language modeling : 하나의 글자를 RNN의 입력값으로 받고, RNN은 다음에 올 글자를 예측하는 문제
- Charr-RNN : 출력값 형태는 학습에 사용하는 전체 문자 집합에 대한 소프트맥스 출력값이 됨

4-4. TensorFlow 이용한 Char-RNN 구현

deep-learning-tensorflow-book-code/train_and_sampling_v2_keras.py at master · solaris33/deep-learning-tensorflow-book-code
[『텐서플로로 배우는 딥러닝』, 솔라리스, 영진닷컴, 2018] 도서의 소스코드입니다. Contribute to solaris33/deep-learning-tensorflow-book-code development by creating an account on GitHub.

https://github.com/solaris33/deep-learning-tensorflow-book-code/blob/master/Ch08-RNN/Char-RNN/train_and_sampling_v2_keras.py

4-5. Gradient Clapping

- 계산과정에서 vanishing gradient problem 과 반대로 gradient 가 무한히 커지는 exploding gradient problem 발생할 수 있음 → gradient clapping 으로 해결
- threshold 지정해서 특정 threshold 넘으면 gradient 다시 작아지게 바꿔줌

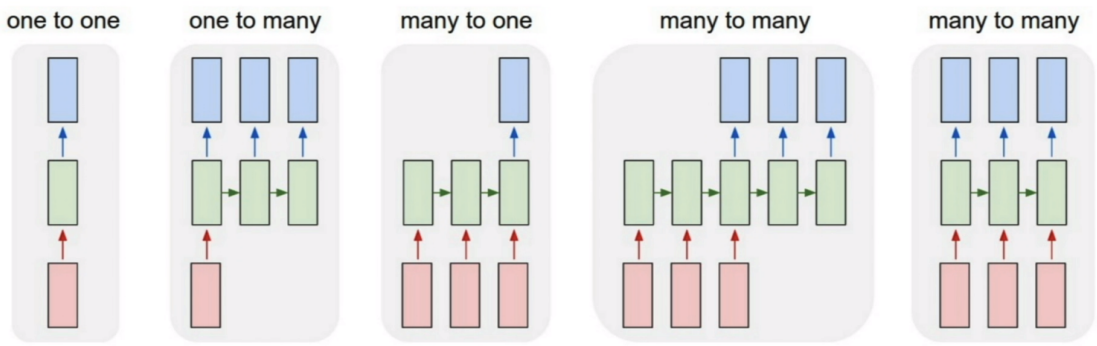
```
def train_step(model, x, y):
  with tf.GradientTape() as tape:
    y_pred = model(x)
    loss = mse_loss(y_pred, y)
  gradients = tape.gradient(loss, model.trainable_variables)
  # Gradient Clipping을 적용
  clipped_grads = []
  for grad in gradients:
    clipped_grads.append(tf.clip_by_norm(grad, grad_clip))
  optimizer.apply_gradients(zip(clipped_grads, model.trainable_variables))
```

4-6. Bidirectional Recurrent Neural Networks(RNNs)

- 이전상태 뿐만아니라 이후 정보를 모두 사용
- “푸른 하늘에 00 이 떠있다” ← 떠있다 라는 단어까지 고려하면 00 예측에 더 좋음

```
model = tf.keras.Sequential([
        tf.keras.layers.Embedding(encoder.vocab_size, 64),
        tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dense(1)
])
```

4-7. RNN 의 다양한 구성형태



- one to one : 1개의 인풋을 받아서 1개의 아웃풋을 출력
  - Char-RNN 등
- one to many : 1개의 인풋을 받아서 여러개의 타임스탭 아웃풋 출력
  - Image Captioning 등
- many to one : 여러개를 인풋으로 받아서 1개의 타임스탭의 아웃풋 출력
  - Sentiment Classification
- many to many : 여러개 인풋 → 여러개 아웃풋
  - Machine Translation (1) : seq2seq 형태로 인풋 데이터의 특징을 인코딩 한 뒤 이를 순차적으로 디코딩 할 수 있음, 시간축 엇갈림 있음
  - Video Classification (2) : 시간축 내에서 수행

4-8. Sentiment Classification, IMDB movie review dataset

- input → text vectorization → embedding → bidirectional → dense → classification
- https://github.com/daheeda/2023-Tobigs-CV-seminar/blob/main/week1/imdb_movie_review_sentiment_classification_using_rnn_py의_사본.ipynb

5-1. 일반적인 딥러닝 논문 구현 프로젝트의 파일 구조

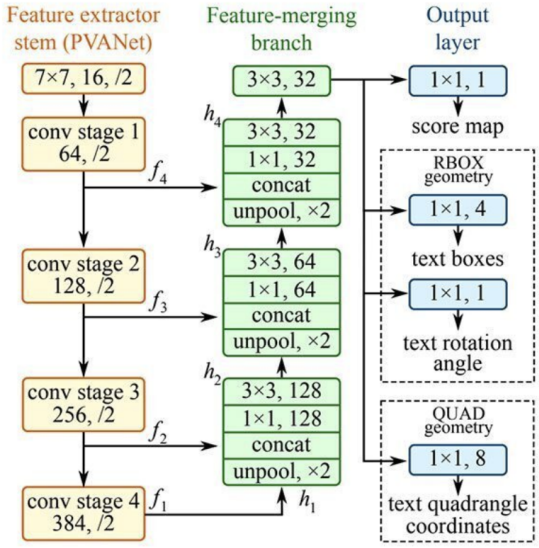
- train.py
- evaluate.py / test.py
- model.py
- dataset.py
- utils.py
- loss.py

6-1. Text Detection 문제영역 소개

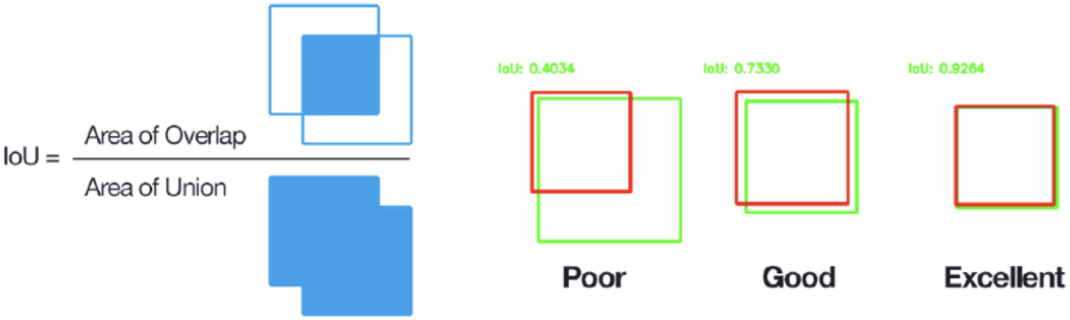
- 이미지 내에 텍스트가 존재하는 영역의 위치정보를 bounding box로 찾는 문제 영역
- object detection과의 차이는 앵글까지 고려한다는 점

6-2. EAST(Efficient and Accuracy Scene Text detector)

- https://github.com/argman/EAST

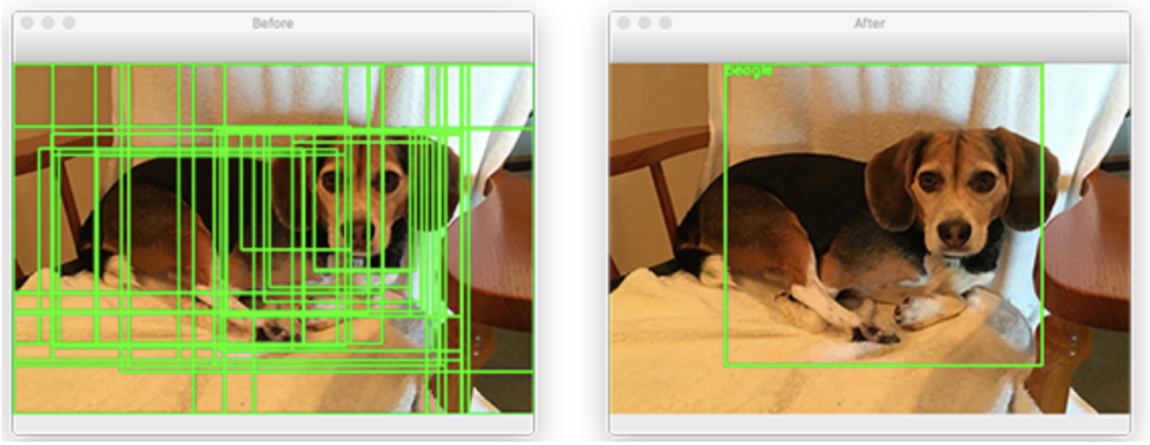


- 특징추출레이어 - 피처머징레이어 - 아웃풋레이어
  - stem layer : 원본에서 특징 추출 만들어주는 레이어, conv 레이어 탈수록 피처맵 사이즈 작아지게 만들어 줌
  - feature-merging : 작은글자/큰글자 다 잘 찾을 수 있도록, 작은 피처맵 크기 키워줌
  - output : (1)rbox, (2)quad
- loss ← score map loss func + geometry loss func
- intersection over union metric
  - target과 prediction 간 percent overlap 을 정량화 하는 방법
  - 바운딩 박스 겹치는 정도를 0~1 사이로 표현



$$IoU = \frac{target \cap prediction}{target \cup prediction}$$

- Non-Maximum Suppression



6-3. CRAFT(Character Region Awareness For Text detection)

https://clova.ai/ocr/

- 곡선형태로 잡기 위해 character 단위로 한글자씩 예측 + affinity 로 단어 합칠지 말지 결정

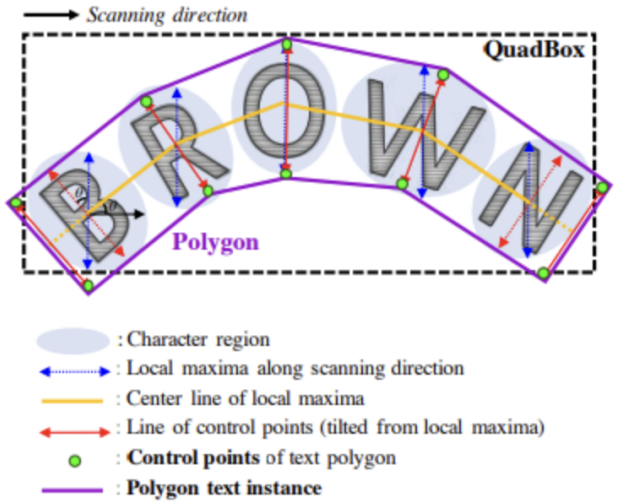


Figure 7. Polygon generation for arbitrarily-shaped texts.

1) scanning direction을 따라 local maxima line (파란색 선)을 찾음

2) 최종 polygon 결과가 울퉁불퉁해지는 것을 막기 위해 local maxima의 길이는 local maxima중에 제일 긴 길이로 동일하게 설정

3) local maxima의 중심을 이어 center line(노란색 선)을 찾음

4) local maxima line 이 중심선에 수직이 되도록 회전하여 문자의 기울기 각도를 반영 (빨간색 선)

5) local maxima line의 endpoint는 text polygon control points의 후보

6) 두 개의 가장 바깥쪽으로 기울어진 Local maxima line을 Center line을 따라 바깥쪽으로 이동하여 최종 Control point를 결정한다.(텍스트 영역을 완전히 커버하기 위함)

2023.04.09 최다희