Gradient Based Learning Applied to Document Recognition

(Y. LeCun, 1998)

https://www.researchgate.net/publication/2985446_Gradient-Based_Learning_Applied_to_Document_Recognition

💡 CNN을 사용하면 문자 인식 분야에서 다른 머신러닝 모델에 비해 월등한 결과를 낼 수 있다

- 중요한 내용은 Gradient based learning과 LeNET-5까지, 그 아래는 이를 사용하여 문서인식을 하는 framework에 대한 소개

기존의 pattern recognition은 feature extracter → trainable classifier 2계층 구조로 되어있음

⇒ 이 때 feature extractor은 사전에 정의된, 제한된 정보만을 추출할 수 있고 classifier은 fully connected layer으로 이루어져있기 때문에 과도한 양의 연산이 필요한 경우가 많음

Gradient based learning

- Input pattern Z와 이에 해당하는 ground truth D에 대해 gradient based learning machine은 아래와 같은 함수 F를 계산함

$$Y^p = F(Z^p, W)$$

(Z^p : p 번째 input pattern, W : system 내부에 조정가능한 변수들의 집합)

- 이 때 함수 F의 결과값 Y가 최대한 비슷해야하므로 참값과 예측값의 차이(=error)을 최소화하는 것이 training의 목표임

$$E^p = D(D^p, F(Z^p, W)), E_{train} = \bar{E^p}$$

(E^p : p 번째 값에 대한 error, D : loss function)

- 모델의 최종목표는 어떠한 subset이 아닌 모든 data에 대해 정확한 예측을 하는 것이기 때문에 E_{train}은 엄밀하게 보면 꼭 최소화해야되는 값은 아님
  - train set이 아닌 원소들을 뽑아 test set을 꾸리고 해당 집합에서의 error을 계산하고 이를 전체집합에서의 error이라고 근사함
  - 즉, 최종적으로 최소화하고자하는 값은 E_{test}값임

- E_{test}의 기대값은 대략 아래 수식과 같다는 것이 알려져있음 [1]

$$E_{test} - E_{train} = k(h/P)^{\alpha}$$

(k : 상수, α : 0.5~1사이의 값, h : model 복잡도와 관련된 값, P : training sample 수)

  - 이 때 training error을 줄이기 위해 더 복잡한 모델을 사용하면 h가 증가하여 test와 train error 사이의 간극이 벌어지는 trade-off가 발생함
  - 이를 해결하기 위해 Structural Risk Minimization을 수행하여 아래와 같은 값을 최소화함

$$E_{train} + \beta H(W)$$

(β : 상수, H(W) : 모델 복잡도에 비례하는 값)

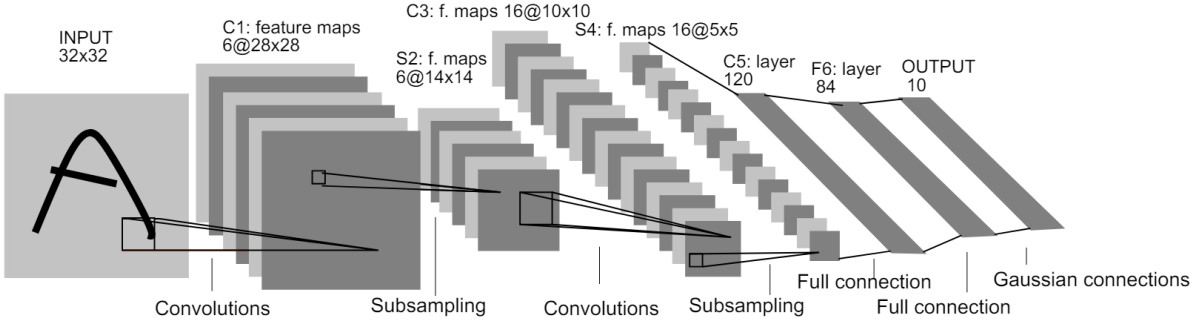
※ 이후부터 학습과정에서 최소화하고자하는 error값은 $E(W)$로 표기함

- E를 최소화시키는 W를 찾기 위해 stochastic gradient algorithm을 자주 사용함

$$W_k = W_{k-1} - \epsilon \frac{\partial E^{p_k}(W)}{\partial W}$$

(newtonian method처럼 전체 E값을 사용하는 것이 아니라 각 sample에 대해 E값을 계산하여 이를 하나하나 updating해주면 수렴 속도가 더 빠름)

Convolution Neural Network (단일 글자 인식)



LeNET-5 : 32*32 크기 흑백 이미지를 받아 크기 10의 vector 출력하는 모델

- C1 : 5*5 kernel 사용하는 convolution layer, 6*28*28 크기의 data 출력
- S2 : $y = \alpha \bar{x} + \beta$꼴로 2*2 영역을 pooling하는 layer (α, β : 학습가능한 parameter)
- C3 : 5*5 kernel 사용하는 convolution layer, 6 channel → 16 channel로 확장
  - 모델의 복잡도를 줄이고 대칭성을 없애기 위해 C3의 각각의 channel을 만들기 위해 S2의 일부 channel만 사용함
  - 연결구조는 아래와 같음

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

- S4 : S2와 같은 구조의 pooling layer
- C5 : 5*5 kernel 사용하는 convolution layer, 16개의 channel이 120개로 증가함

    - 32*32 크기의 input에 대해서는 fully connected layer와 사실상 동일하게 작동함
- F6 : 120 → 84 로 가는 fully connected layer (scaled tanh함수로 activation)
- Output : Gaussian connection, $y_i = |x - w_i|^2$ 반환 (Radial Base Function 결과)
    - w_i : 사전에 정의된 input와 크키가 같은 vector
    - 7*12 = 84 크기 공간에서의 숫자 형태를 띔(='normalize된 정답')

      ⇒ F6에서의 출력값과 이러한 w_i 벡터 사이의 거리를 최종적으로 출력함

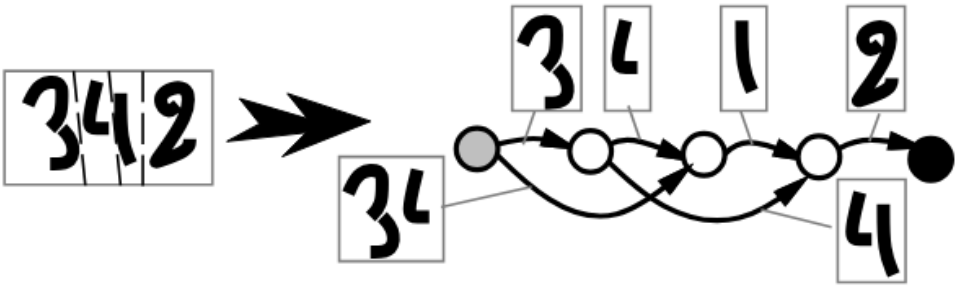
※ 실제 문자열 인식에서 헷갈리기 쉬운 문자들 (1, l, [,] 등등)을 나중에 보정하기 위해서는 이러한 방식이 더 적합함 (fc layer을 softmax하여 출력하면 이러한 오류를 나중에 고칠 수 없음)

- Loss function : 아래와 같은 function 사용함
    - y_{D^p} : p번째 input의 정답에 해당하는 rbf 결과값 (output layer 출력값)
    - y_i : i번째 label의 rbf 결과값
    - j : 훈련 말기에 log 함수 내의 값이 너무 작아지는 것을 막는 상수

$$E(W) = \frac{1}{P} \sum_i \left(y_{D^p}(Z^p, W) + \log \left(e^{-j} + \sum_i e^{-y_i(Z^p, W)} \right) \right)$$

Graph Transformer Network : Heuristic oversegmentation

- 고정된 길이의 vector을 입력으로 받는 신경망은 구조적인 한계가 존재함
    - variable-length의 입력 처리가 불가능 e.g. speech, text 등
- 가변 길이의 데이터를 표현하기 위해서는 directed acyclic graph (DAG)가 유용함
    - graph의 각 edge가 vector 하나를 의미
    - graph의 start node에서 end node까지의 complete path 하나하나가 전체 data의 가능한 segmentation 중 하나를 의미

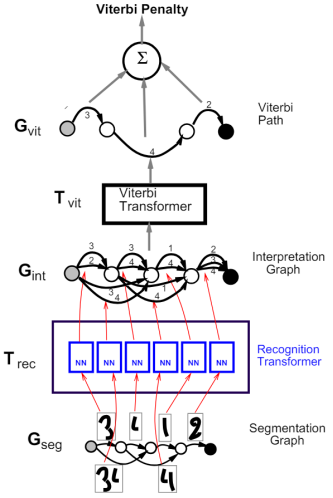


e.g.) 임의로 segmentation을 한 후 이를 DAG로 나타낼 수 있음

이때 node 0 → 1 → 3 → 4로 가는 path가 'best' segmentation임

- 두 개의 network를 같이 사용하면 문자열을 인식하는 network를 만들 수 있음
  - Recognition Transformer과 Viterbi Transformer 2개로 구성
  - Recognition Transformer : Segmentation graph의 각 edge 대해 출력과 penalty 쌍 (Y,E)를 반환함 (여러개의 candidate를 반환함)
  - Viterbi Transformer : Recognition Transformer이 반환한 Interpretation graph에서 total penalty가 최소가 되는 complete path(viterbi path)를 반환함

    (dynamic programming으로 계산 가능)

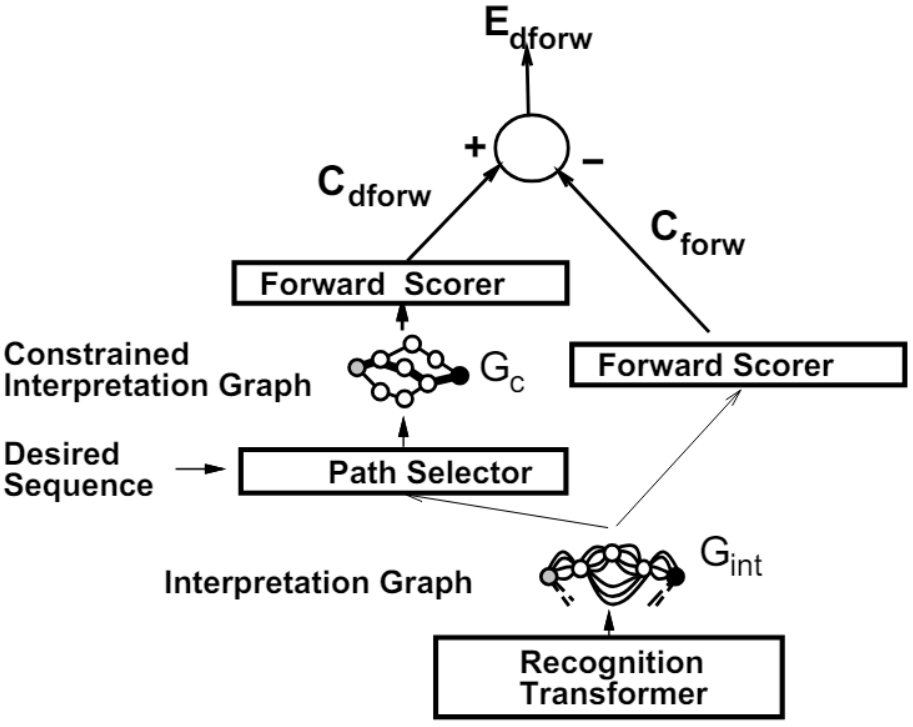


Discrimminitave Forward Training GTN

- 단순한 viterbi GTN의 경우 아래와 같은 문제를 가짐
  - Viterbi path가 실제 답이라는 보장이 없음
  - Error 항이 개별 loss의 합이므로 gradient가 0또는 1임 ⇒ 상관없는 edge는 갱신되지 않음
  - Error을 더하는 것은 통계적으로 의미가 없음
  - Total penalty가 의미를 가지지 않음 (⇒ 예측의 reliability에 대한 척도로 쓸 수 없음)

  ⇒ 이를 해결하기 위해 아래와 같은 구조의 network를 구상함

- Interpretation Graph 생성까지는 Viterbi GTN과 동일함



- Interpretation Graph 생성까지는 Viterbi GTN과 동일한 구조
- Path Selector을 통해 G_{int}에서 실제 정답과 같은 문자순서를 가진 path만을 골라내어 constrained Interpretation Graph (G_c)를 얻음
- G_c와 G_{int}에 대해 forward scorer을 통해 최단 경로를 계산, cumulative penalty 도출
  - 두 edge의 penalty값을 더할 때 단순 합산하지 않고 logadd() 연산을 통해 합침

$$\text{logadd}(x_1, x_2, ..., x_n) = -\log\left(\sum^{n} e^{-x_i}\right)$$

- 각 node에 대해 score는 exp (-penalty)로 계산함
- Model의 최종 error E_{dforw}은 Forward scorer을 통해 구해진 2개의 score의 차

이를 통한 back propagation은 아래와 같음

※ 아래 유도과정에서 E는 scorer을 통해 나온 score값에 대한 기울기값임

⇒ E_{dforw}에 대한 기울기는 각각 score에 대한 기울기의 차이로 계산하면 됨

D_n : node n에서 나오는 edge의 집합, s_i : edge i의 시작점, d_i : edge i의 끝점

$$\frac{\partial E}{\partial f_n} = \sum_{i \in D_n} \frac{\partial E}{\partial f_{d_i}} \frac{\partial f_{d_i}}{\partial f_n}$$

$$f_{d_i} = -\log(C + \exp(-(c_i + f_n)))$$

$$\frac{\partial f_{d_i}}{\partial f_n} = \frac{1}{C + \exp(-(c_i + f_n))} e^{-(c_i + f_n)} = e^{-f_n} e^{f_{d_i} - c_i}$$

$$\Rightarrow \frac{\partial E}{\partial f_n} = e^{-f_n} \sum_{i \in D_n} \frac{\partial E}{\partial f_{d_i}} e^{f_{d_i} - c_i}$$

위 식을 통해 재귀적으로 node마다 loss에 대한 기울기를 구할 수 있고 이를 사용하면 penalty에 대한 loss의 기울기를 계산할 수 있음

$$\frac{\partial E}{\partial c_i} = \frac{\partial E}{\partial f_{d_i}} \frac{\partial f_{d_i}}{\partial c_i} = \frac{\partial E}{\partial f_{d_i}} e^{-c_i - f_{s_i} + f_{d_i}}$$

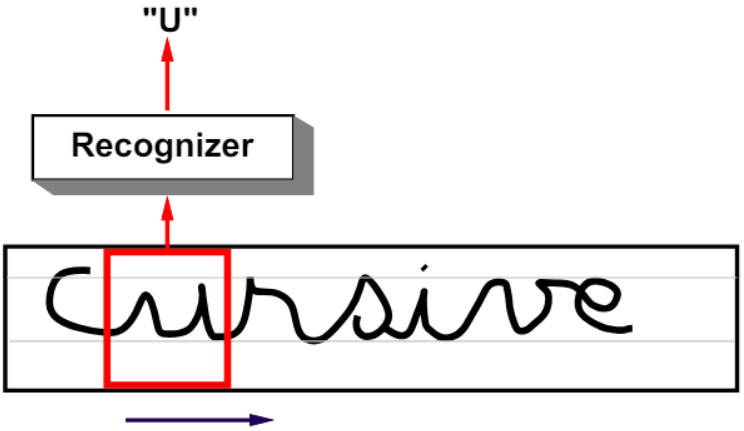
해당 gradient값을 recognition transformer의 신경망에 입력하여 오류역전파를 하면 모델 갱신 가능

Space Displacement Neural Net(SDNN)

- 단어를 인식하는 또 다른 방식
  - Segmentation을 하는 대신 모든 가능한 위치에서 recognizer을 적용함
  - Sliding window algorithm과 유사한 방식

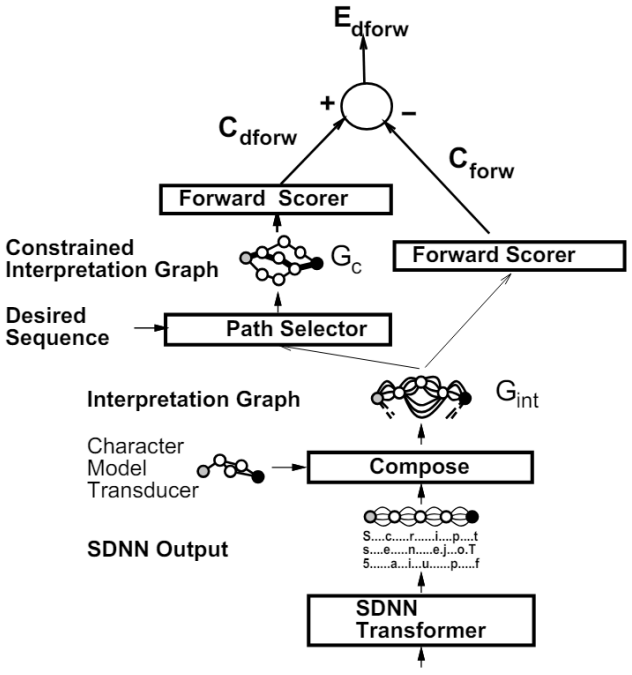
- CNN의 경우 kernel을 옮기면서 계산하기 때문에 SDNN과 원리가 유사함

⇒ CNN을 사용하면 SDNN을 더 적은 계산량을 통해 구현할 수 있음



- Discrimminitive Forward Training GTN과 구조는 거의 유사함
  - SDNN transformer을 통해 각 위치마다 예측값을 담은 그래프 출력
  - Character model transducer 을 통해 'legel'한 예측값들만 남겨 interpretation graph를 형성

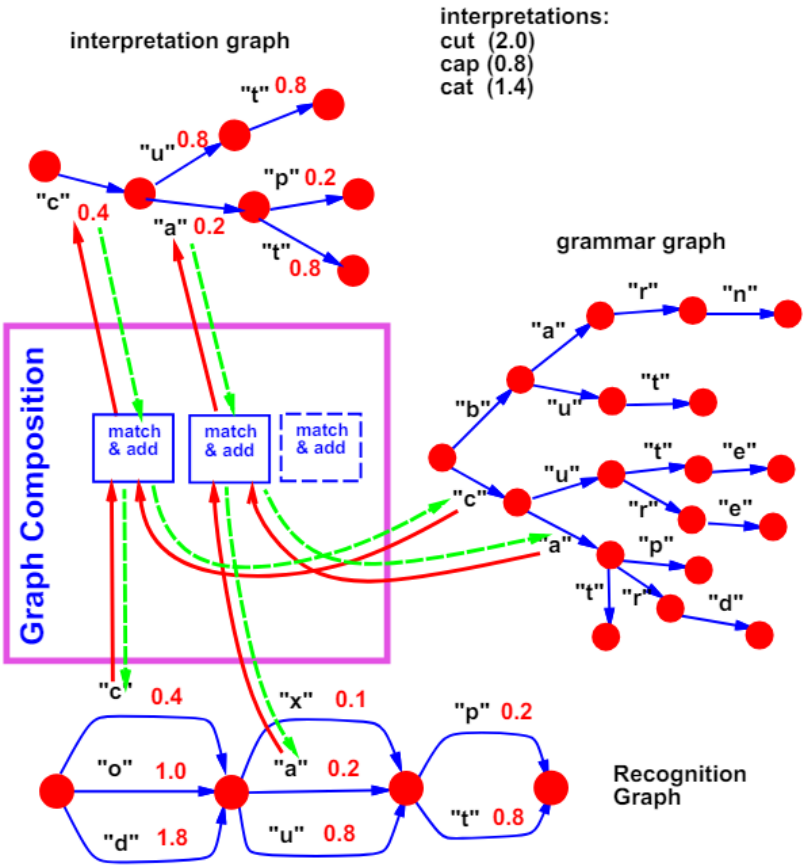
    ⇒ 예측한 문자열들 중에서 말이 되는 문자열만 남김
  - 이후 과정은 Discrimminitive Forward Training GTN과 동일함



- 해당 메커니즘을 응용하면 단어인식뿐만 아니라 사진에서 object detection을 하는데 사용할 수 있음

Graph Transformer Network/Transducer

- 위 절에서 interpretation graph를 만드는 과정은 transduction operation이라고 함
  - 해당 방법은 그래프의 edge가 이산적인 data (문자열 등)을 담고 있을때 사용 가능



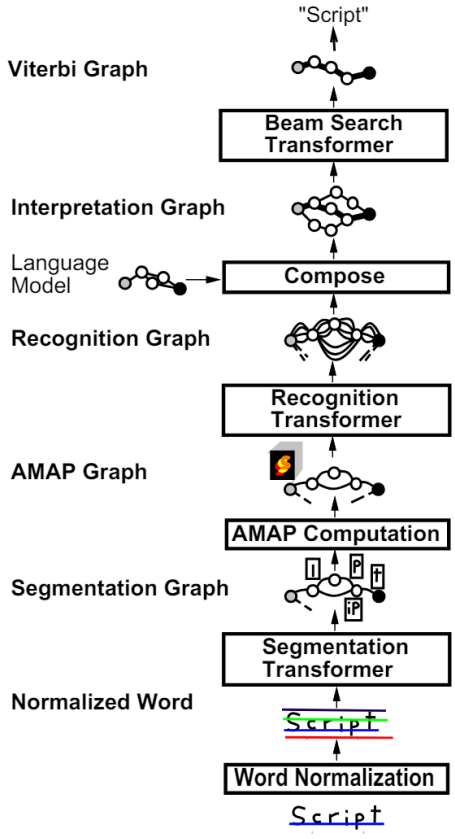
⇒ edge에 더 복잡한 data 구조가 들어있을 때 사용가능한 알고리즘을 만들고자함

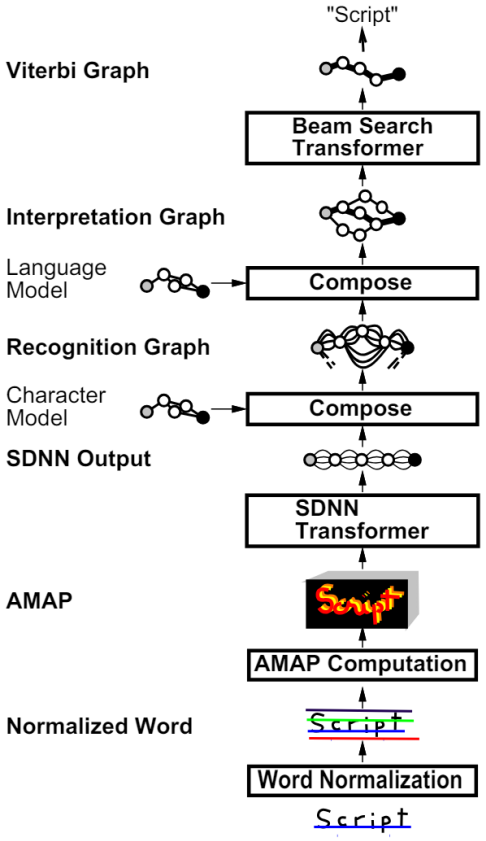
- Composition Transformer
  - edge에 복잡한 data (이미지, 벡터 등)을 포함한 그래프 두개를 합치는 알고리즘
  - check, fprop, bprop 3개의 method를 가져야함

        - check : 두 개 그래프의 edge를 입력으로 받고 두 edge에 대한 새로운 edge를 만들어야하는지 확인
        - fprop : check == true인 경우 두 개의 edge들로부터 새로운 node와 edge를 만듬
        - bprop : fprop 과정의 도함수를 포함, 역전파에 사용
    - e.g.) 위 상황에서 recognition graph와 grammar graph의 두 edge에 같은 값이 들어있으면 check = true, 이때 fprop에 의해 recognition graph의 edge와 동일한 새로운 edge 생성

Online Handwriting Recognition

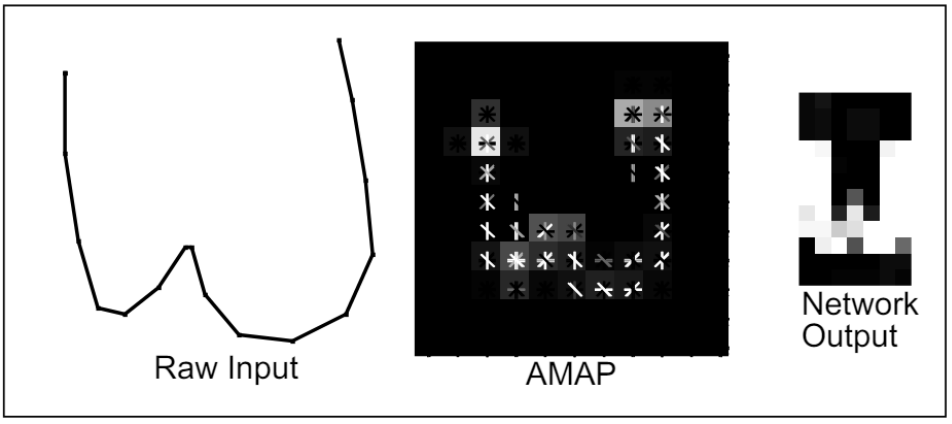
- 온라인상에서 실시간으로 글씨를 쓰고 있을 때 이를 인식하는 system





- Heuristic Oversegmentation
    - word normalization : 휘거나 기울어진 단어를 신경망이 인식하기 쉽게 똑바로 펴줌
    - Segmentation : 임의의 위치에서 잘라줌

    - AMAP : Input data는 시간에 따른 펜의 (x,y) 좌표로 나타낼 수 있음 ⇒ 해당 데이터를 사용하여 글씨를 사진으로 바꾸어줌 (raw input → network output)

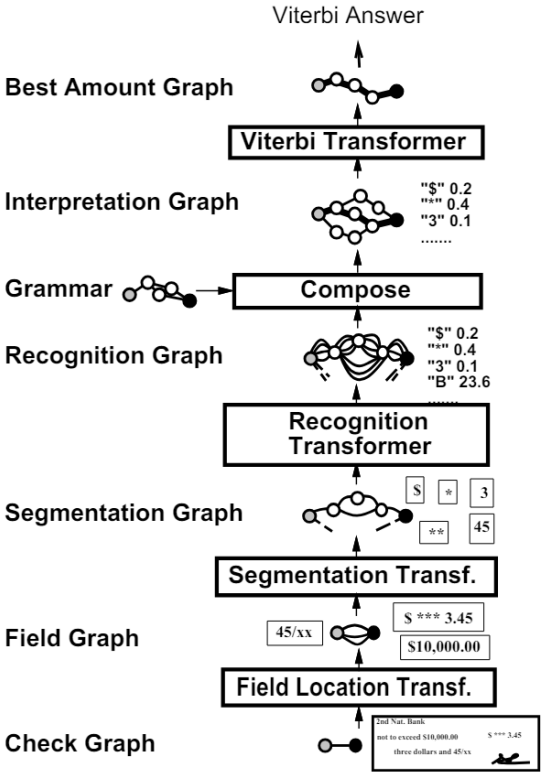


    - 위 절에서 설명한 recognition graph, interpretation graph 생성
    - Beam Search Algorithm을 사용하여 최적의 path(=경로) 찾음
- SDNN
    - Word Normalization
    - AMAP
    - 이후 생성된 사진에 위에서 설명한 SDNN 모델을 적용

- Recognition에 사용된 network은 LeNET과 유사한 구조
    1. 8개의 3*3 kernel로 convolution
    2. 2*2 pooling
    3. 25개의 5*5 kernel로 convolution
    4. 84개의 4*4 kernel로 convolution
    5. 2*1 pooling
    6. 95개의 rbf unit을 통과시켜 최종 예측

Check Reading System

- 은행 수표의 금액을 읽어내는 모델



- Field Location Transformer
  - 수표 사진을 인식하여 field graph를 출력함
  - connected component analysis, ink density histogram, layout analysis 등 다양한 기법을 사용하여 수표의 금액이 적혀있을법한 위치들을 뽑아냄
  - Field graph는 2개의 node로 구성, 각각의 edge에는 전체사진에서 추출된 region과 해당 지역이 실제 정보를 담고 있는지 여부와 관련된 loss값이 있음
- Segmentation Transformer
  - 추출된 이미지를 자름
- Recognition Transformer
  - 잘라진 이미지를 인식, 글자와 loss값을 반환
- Compose
  - 추출된 글자들의 조합 중 'legal'한 = 문법적으로 말이 되는 조합만을 추출함
- Viterbi Transformer
  - 추출된 글자들의 조합 = 문자열 중 loss값이 최소인 것을 얻음

Conclusion

1. CNN을 사용하면 사진들에서 feature을 추출할 수 있음
2. Segmentation과 Recognition은 완전히 분리할 수 없음 : 대충 자르고 글자를 인식한 후 이를 다시 조합하는 과정이 필요함
3. 단어를 직접 segmentation하는 대신 제안된 graph transformer network를 사용하면 적은 수고를 들이고 모델을 훈련할 수 있음
4. Segmentation, Recogition, 그리고 이를 조합하는 과정은 task-specific할 필요가 없음
5. SDNN을 사용하면 Segmentation을 할 필요 자체가 없음

Seung, H. & Sompolinsky, Haim & Tishby, Naftali. (1992). Statistical mechanics of learning from examples. Physical review. A. 45. 6056-6091. 10.1103/PhysRevA.45.6056.