# Vision과 NLP 강의 정리

Vision : CS231n 11강까지 수강

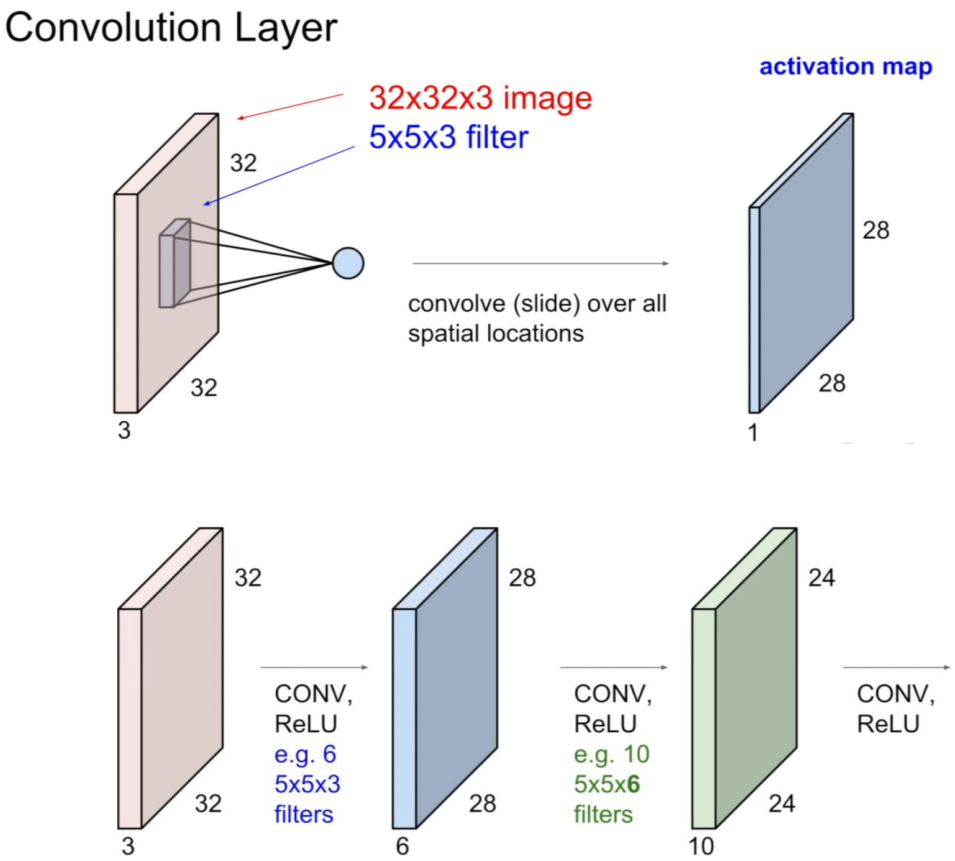
NLP: CS224n 5강까지 수강

정한주

# Vision - 이미지 인식

# CS231n

# #1. CNN

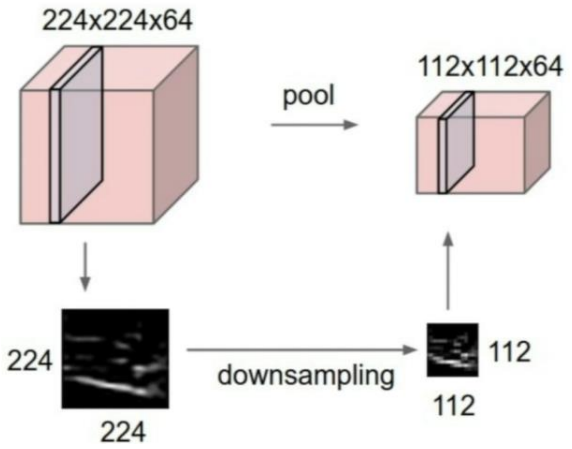



- 32*32*3 image의 기존 구조를 보존할 수 있음
- 여기에 5*5*3 filter를 취함. 이 필터를 이미지에 특정한 공간에 겹쳐놓고 내적을 수행함.
- Convolution은 이미지의 좌상단부터 시작하여 쭉 연산해나아감.
- 보통 여러 개의 필터를 사용하여 필터마다 다른 특징을 추출하도록 함.
- Activation map: 입력 이미지에 필터를 적용하고 출력되는 특징맵
  특징맵은 필터 개수만큼 생성됨 ex)5*5*3 filter 6개: 총 6개의 activation map을 얻음.
- Activation map의 출력 사이즈: {(N-F)/S} + 1
- 중간중간 activation function을 삽입함 (ex: ReLU)
- zero padding: 이미지 가장자리에 0을 채워서 모서리 데이터가 손실되는 것을 방지함.
- 제로 패딩을 적용한 특징맵의 출력 사이즈: {(N+2P-F)/S} + 1

# #1. CNN

## Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



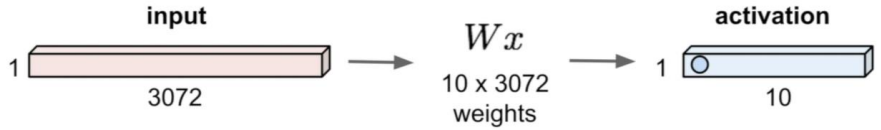

- 다운샘플링: representation들을 더 작고 쉽게 관리하도록 함.
- 파라미터 수가 줄어들고 공간적인 불변성을 가져감

➔ 공간적으로 크기를 downsample함

- Depth에는 아무짓도 하지 않고, max pooling이 주로 쓰임

## Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

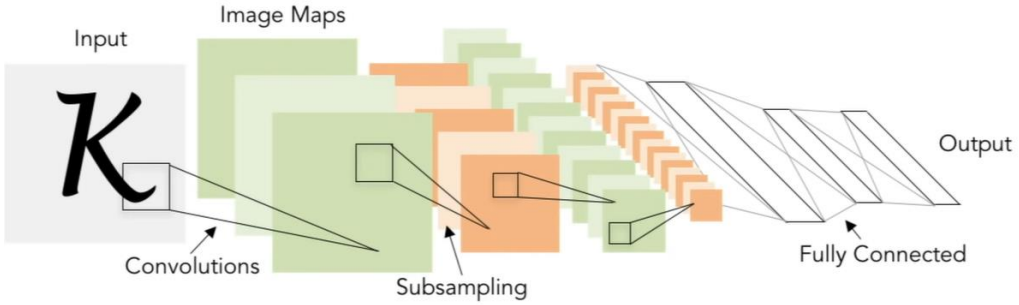



- 32*32*3 image 를 3072*1의 구조로 쫙 핌

# #2. CNN Architectures

## Review: LeNet-5

[LeCun et al., 1998]




Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

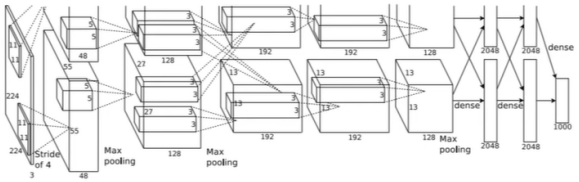
- 산업에 성공적으로 적용된 최초의 ConvNet
- 이미지를 입력으로 받고, 보폭이 1인 5*5 필터와 pooling layer를 거친 후 끝단에 FC layer가 있는 간단한 구조

➔ 매우 기본적인 모델이나 숫자 인식에서 엄청난 성공을 거둠

## Case Study: AlexNet

*[Krizhevsky et al. 2012]*

**Architecture:**
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8



- 최초의 Large Scale CNN
- 기존의 LeNet과 비슷하지만 레이어의 수가 더 많아짐

➔ 5개의 ConV layer와 2개의 FC layer

# #2. CNN Architectures

## Case Study: VGGNet

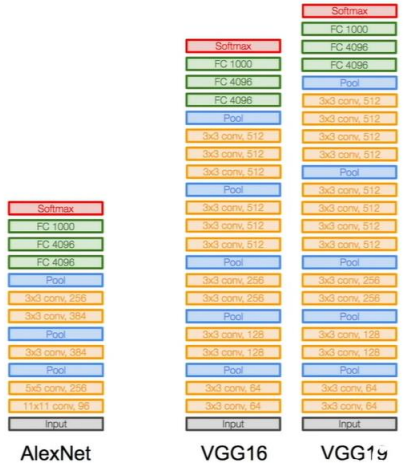
*[Simonyan and Zisserman, 2014]*

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14




- 훨씬 더 깊어지고, 더 작은 필터들을 사용함
- 3*3 ConV 필터: 이웃 픽셀을 포함할 수 있는 가장 작은 필터 사이즈
- 작은 필터를 사용하는 이유: 필터 크기가 작으면 파라미터 수도 더 작음

➔ 더 큰 필터에 비해 더 깊은 레이어를 쌓을 수 있음(Depth)

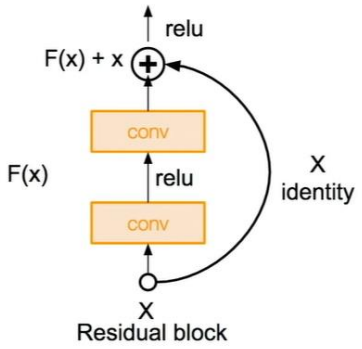
- VGG16과 VGG19: 각각 16개, 19개의 ConV layer를 가짐

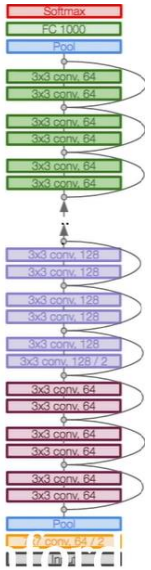
## Case Study: ResNet

*[He et al., 2015]*

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!

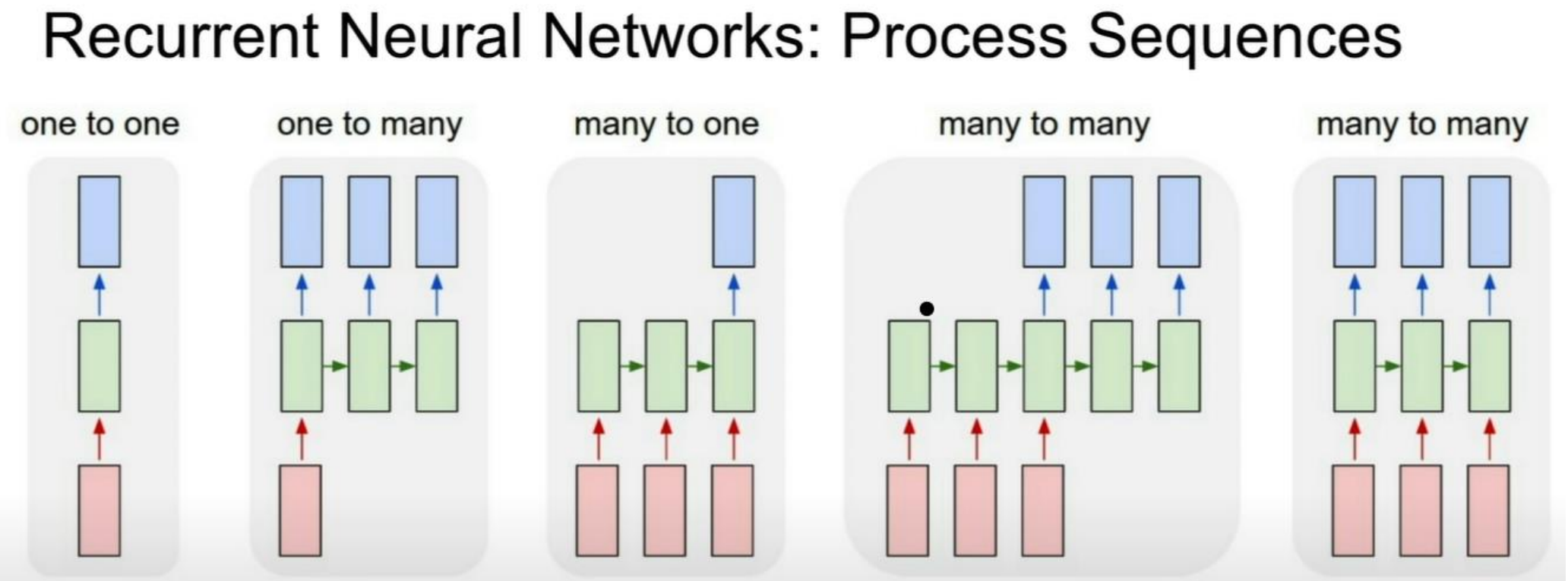


- H(X) = F(X) + X로 설정하여 계산량을 줄이기 위해
  residual인 F(x)만 학습시켜보자는 아이디어
- FC-layer를 상당히 걷어내고 gap으로 대체 시켜 파라미터 수를 많이 줄임
- Direct mapping 대신 residual mapping을 하는 방식으로 블록을 쌓음

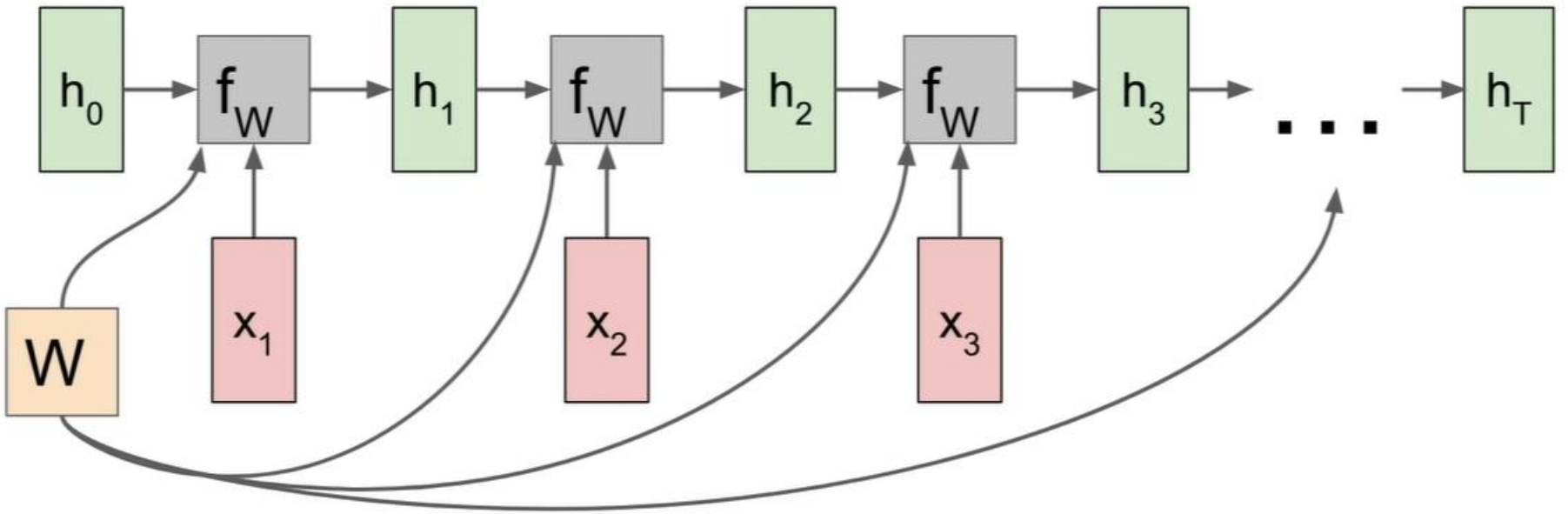
➔ 즉 ResNet = residual block을 쌓아 올리는 구조

# #2. RNN




- One to one: 입력 하나가 히든 레이어를 거쳐서 하나의 출력을 내보냄 (ex: Vanila Neural Networks)
- One to many: image captioning(이미지를 설명하는 자연어 문장을 생성하는 작업)
- Many to one: Sentiment Classification(감성 분석)
- Many to many: 기계 번역, video classification

# #2. RNN

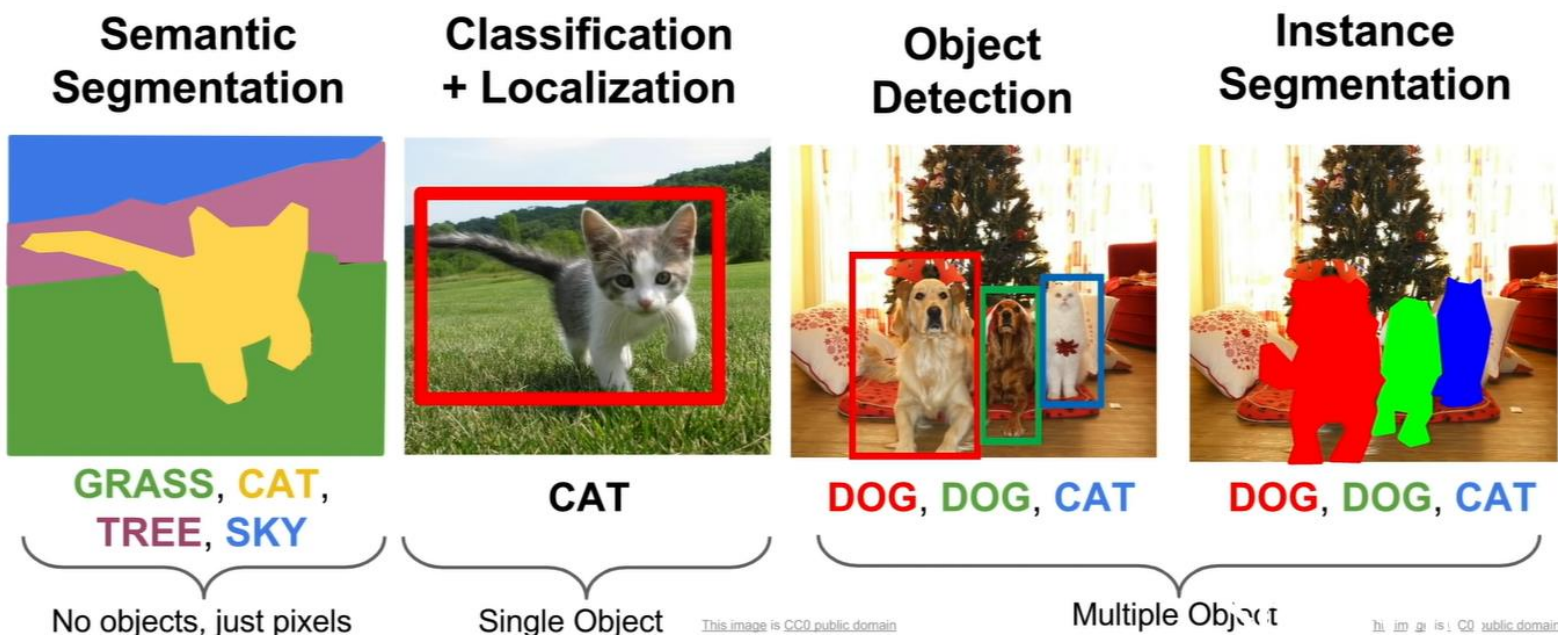



- RNN이 hidden state를 가지고 이를 재귀적으로 feedback함

1) Initial hidden state인 h_0, 보통 0으로 초기화
2) 입력 x_t
3) h_0과 x_1이 함수 f(w)의 입력으로 들어가서 출력 h_1으로 나옴
4) 이 과정을 반복하면서 가변 형태의 입력값인 x_t를 받음

- 매 step마다 h와 x는 달라지지만 가중치 w는 동일함
- RNN의 최종 loss; 각 개별 step의 loss들의 합

# #3. Computer Vision Tasks



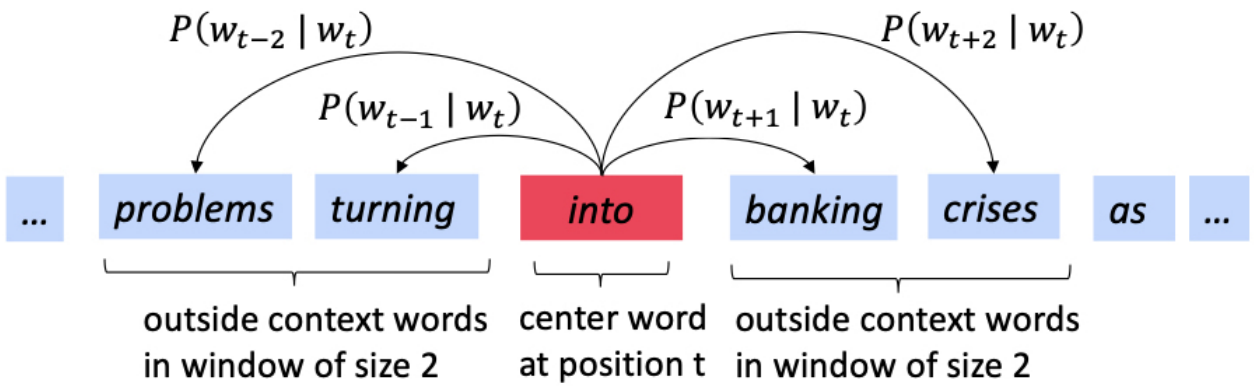

- Semantic Segmentation: 이미지를 픽셀 수준에서 객체의 영역으로 분할하여 객체의 영역을 정확하게 분리함
- Classification + Localization: 이미지를 분류하는 것뿐만 아니라 이미지 내의 객체 위치를 박스로 표시함
- Object Detection:  이미지 내에서 특정 객체의 존재와 위치를 탐지하고 여러 객체를 한 이미지에서 감지할 수 있음
- Instance Segmentation: object detection + segmentation의 작업으로
  이미지에서 각 객체의 위치를 표시하고 동일한 클래스에 속하는 서로 다른 객체를 개별적으로 식별하고 픽셀 수준으로 분할하는 작업을 수행함

# NLP - 자연어 처리

# CS224n

# #1. Neural Classifiers - 1. word2vec

- Example windows and process for computing $P(w_{t+j} \mid w_t)$




- 주변 단어를 예측하는 확률을 최대화하는 것이 목표로 대표적인 skip-gram 모델
- 주어진 중심 단어로부터 주변 단어들을 정확히 예측하는 모델을 학습하여 단어 간의 의미적 유사성을 잘 캡처하는 벡터 표현을 얻는 것
- Stochastic Gradient Descent: 모든 windows에 대한 함수 J(세타)를 사용하여 비용이 증가하는 특징을 완화하기 위해 확률론적 경사 하강법 사용
- Negative  Sampling: sparse한 단어를 고려해 값이 0이 아닌 행에 대해서만 gradient를 계산하여 update함

# #1. Neural Classifiers - 2. Co-occurrence matrix

| counts | I | like | enjoy | deep | learning | NLP | flying | . |
|---|---|---|---|---|---|---|---|---|
| I | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| like | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| enjoy | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| deep | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| learning | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| NLP | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| flying | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| . | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

Stanford

Window based co-occurrence matrix(단어-문맥 행렬)

| - | 과일이 | 길고 | 노란 | 먹고 |
|---|---|---|---|---|
| 문서1 | 0 | 0 | 0 | 1 |
| 문서2 | 0 | 0 | 0 | 1 |
| 문서3 | 0 | 1 | 1 | 0 |
| 문서4 | 1 | 0 | 0 | 0 |

Word-Document matrix (단어-문서 행렬)

- 주어진 중심 단어로부터 주변 단어들을 정확히 예측하는 Skip-gram 모델은 윈도우 개수를 늘린다 하더라도 동시 발생 확률을 알 수 없음

➔ 이를 보완하기 위해 단어가 서로 어떻게 나타나는지에 대한 개수 행렬, 즉 동시발생행렬을 생성함.

- 동시발생행렬은 규모가 크지만 매우 희박하다는 단점이 있어서 저차원으로 차원을 축소해야함

➔ 특이값 분해(SVD)를 도입: 동시발생행렬을 단어나 문서 기준으로 차원을 축소해준 뒤, 각 기준별로 잠재 의미 분석하는 방법(LSA) 가능

# #1. Neural Classifiers - 3. Glove

Crucial insight: Ratios of co-occurrence probabilities can encode meaning components

| | $x$ = solid | $x$ = gas | $x$ = water | $x$ = fashion |
|---|---|---|---|---|
| $P(x\|\text{ice})$ | $1.9 \times 10^{-4}$ | $6.6 \times 10^{-5}$ | $3.0 \times 10^{-3}$ | $1.7 \times 10^{-5}$ |
| $P(x\|\text{steam})$ | $2.2 \times 10^{-5}$ | $7.8 \times 10^{-4}$ | $2.2 \times 10^{-3}$ | $1.8 \times 10^{-5}$ |
| $\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$ | 8.9 | $8.5 \times 10^{-2}$ | 1.36 | 0.96 |

Stanfo

**GloVe results**

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

litoria

leptodactylidae

rana

eleutherodactylus

Stanford

- Word2vec은 효율적으로 통계정보를 사용하지 못하고, 동시발생행렬은 의미없는 큰 빈도수에 중요도를 크게 부여하는 단점이 있었음
- 이를 해결하기 위해 Glove 등장: 임베딩된 단어벡터 간의 유사도를 쉽게 측정하면서도, 전체의 통계정보를 반영할 수 있도록 함

➔ 임베딩 된 단어벡터들의 내적(input)이 말뭉치 전체에서의 동시발생확률 로그값(output)이 되도록 목적함수를 정의함

# #2. Parsing

- **파싱: 문장의 문법적인 구성 또는 구문을 분석하는 과정**
- Consitituency parsing: 문장의 구성요소를 파악하여 구조를 분석하는 방법(phrase-structure grammer) - 어순이 비교적 고정적인 언어(ex:영어)
- Dependency parsing: 단어 간 의존 관계를 파악하여 구조를 분석하는 방법(head-dependent) - 어순이 비교적 자유로운 언어(ex:한국어)

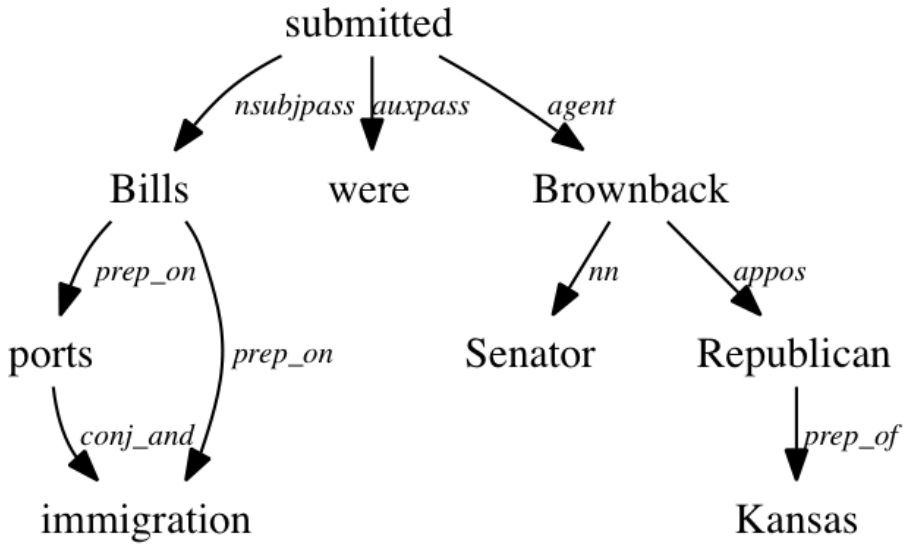
- **Pendency grammar: 모호성**
- 1)  Phrase Attachment Ambiguity
- 구(동사구, 형용사구, 전치사구)가 어떤 단어를 수식하는지에 따라 발생하는 모호성
- 2) Coordination Scope Ambiguity
- 특정 단어가 수식하는 대상의 범위에 따라 발생하는 모호성
- 3) Adjectival/Adverbial Modifier Ambiguity
- 형용사나 부사가 수식하는 단어가 무엇이냐에 따라 발생하는 모호성
- 4) Verb Phrase(VP) attachment ambiguity
- 동사구문이 설명할 부분에 발생하는 모호성

**➔ 단어 간 관계를 트리 구조로 표현할 수 있고 문장 구조에 크게 구애받지 않는 Dependency parsing에 대한 관심이 증가하고 있음**

# #2. Parsing

- **Dependency parsing의 표현방법**


**Sequence**




**Tree(트리)**

- Sequence: 연결관계를 제한없이 표시할 수 있음.
- Tree: 재사용이 쉬움(reusability)
- 루트라는 가상의 노드를 문장의 맨 처음에 추가하고, 최종 head를 root로 설정한 후, 모든 단어가 1개의 의존관계를 가지도록 설정함
- 수식을 받는 dependent와의 문법적 관계를 화살표로 표시

# #2. Parsing

- **Dependency parsing의 구현방법**
  - Dynamic programming
  - Graph Algorithms
  - Constraint Satisfaction
  - Transition-based parsing = deterministic dependency parsing

- **Greedy transition-based parsing**
  - 종속성 구문 분석을 위해 shift reduce 구문 분석과 같은
    일련의 전환을 사용하여 문장 위에 구문 분석 구조를 적용할 수 있게 함.

**Basic transition-based dependency parser**

Start: $\sigma = [ROOT], \beta = w_1, \ldots, w_n, A = \emptyset$

1. Shift $\quad \sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$
2. Left-Arc$_r$ $\quad \sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_j, \beta, A \cup \{r(w_j, w_i)\}$
3. Right-Arc$_r$ $\quad \sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_i, \beta, A \cup \{r(w_i, w_j)\}$

Finish: $\sigma = [w], \beta = \emptyset$

- 세 가지 action 중 하나를 선택
- Shift: 다음 단어가 스택으로 이동된 다음 축소 작업을 수행할 수 있음
- 왼쪽 arc(감소) or 오른쪽 arc(감소)를 선택하고 w_j를 w_i의 종속물로 만듦

➔ 종속 항목이 스택에서 사라지게 됨

- 이 과정에서 목적어, 명사 수식어와 같이 둘을 연결하는 문법적 관계도 지정할 수 있음
- 버퍼가 모두 비어있고 스택에 하나의 단어(루트)만 남아 있는 상태가 최종 시점

➔ 스택과 버퍼가 주어지면 다음 동작(Arc)를 예측할 수 있는 classifier를 작성하면 좋지 않을까? = MaltParser

# #3. Language Modeling

- **자연어처리에서의 언어 모델링**
  - The task of predicting what word comes next
  - 이 문맥에서 다른 단어가 나타날 확률을 제공하는 것으로 이전 문맥이 주어졌을 때 다음 단어에 대한 확률 분포

More formally: given a sequence of words $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$, compute the probability distribution of the next word $x^{(t+1)}$:

$$P(x^{(t+1)} | \, x^{(t)}, \ldots, x^{(1)})$$

where $x^{(t+1)}$ can be any word in the vocabulary $V = \{w_1, \ldots, w_{|V|}\}$

# #3. Language Modeling

- **N-gram Language Models**
  - N-gram: 연속된 N개의 단어 덩어리
  - '다음 단어는 오직 직전의 n-1개의 단어에만 영향을 받는다 ' 는 마르코브 전제를 사용하여, 현재 문장이 주어졌을 때 다음 단어가 올 확률을 계산함

➔ N 단어의 확률을 이전 n-1의 단어로 나눈 값

  - 희소성 문제, 비용 문제, 문맥을 충분히 반영하지 못하는 단점이 있음.

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\overbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}}^{n\text{-1 words}})$$

(assumption)

prob of a n-gram

prob of a (n-1)-gram

$$= \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$

(definition of conditional prob)

- **Question:** How do we get these *n*-gram and (*n*-1)-gram probabilities?
- **Answer:** By counting them in some large corpus of text!

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$

(statistical approximation)

Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the~~ students opened their _____

discard

condition on this

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

For example, suppose that in the corpus:

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - → P(exams | students opened their) = 0.1

Should we have discarded the "proctor" context?

# #3. Language Modeling

- **Neural Language Model**
  - 다음 단어를 예측하기 위해 특정 개수의 단어를 입력으로 받음
  - 입력값을 적절히 embedding 해준 뒤 Neural Network에 넣음
  - Softmax를 거쳐 output distribution을 출력함

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

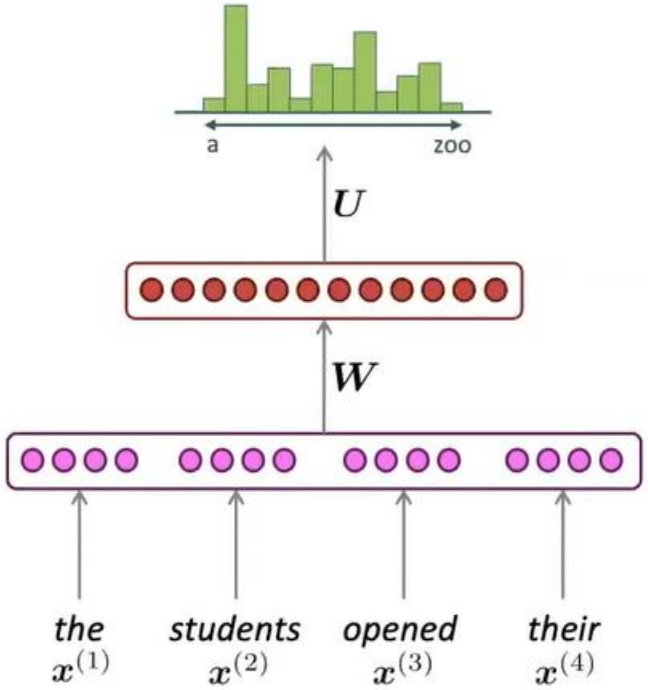
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$




- Softmax를 사용하여 희소성 문제를 해결했으나, 여전히 window를 파라미터로 사용하여 문맥을 반영하지 못하는 문제가 있음

➔ 임의의 양의 context를 처리하면서 파라미터를 더 많이 사용하면서도 근접성에 민감할 수 있는 모델이 필요함

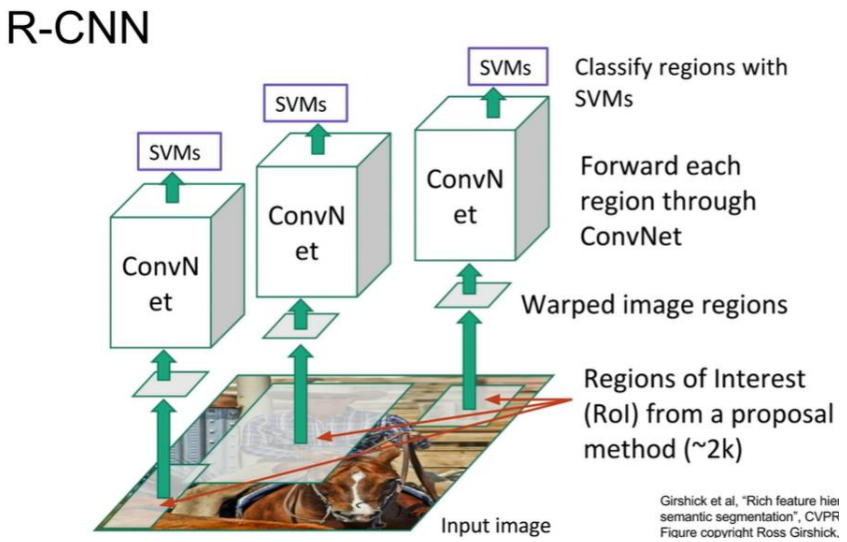
➔ RNN의 도입

# 희망하는 연구 세부분야

- **컴퓨터비전 - 객체탐지, 인스턴스 분할**

- **R-CNN: Object detection 분야에 특화된 방식으로 설계된 CNN모델**
- 1) Region Proposal: 영역 제안, 이미지 내에서 가능성 있는 객체의 위치를 추천함
- 2) CNN Feature Extraction: 영역 제안에 대해 CNN을 사용하여 이미지 특성 추출
- 3) 분류와 바운딩 박스 회귀
  - 추출된 CNN 특성을 FC의 입력층으로 사용하여 여러 클래스에 대한 분류와 바운딩 박스 회귀를 수행함
  - 각 영역 제안에 대한 객체의 클래스와 경계 박스를 예측함
- 단점: 처리속도가 느려서 계산비용과 훈련시간 소요가 큼

➔ 이를 개선하기 위해 Fast R-CNN이 도입됨




- **Mask R-CNN: Instance Segmentation 분야에 특화된 방식으로 설계됨**
- B-box를 통해 개체를 찾은 다음 픽셀별로 예측하는 방법
- Feedforward 한 번으로 이미지 내에 객체 수, 위치, 해당하는 픽셀 등을 알아낼 수 있음
- 객체의 신체 구조 추정(skeleton estimation)도 가능
- 많은 객체들이 겹쳐있는 이미지에도 잘 작동함