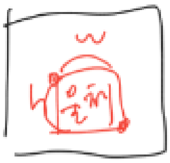
Object localization.

Softmax를 통해 나오는 레이블 $y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$

(c_1, c_2, c_3) → 물체의 분류에 대한 클래스값

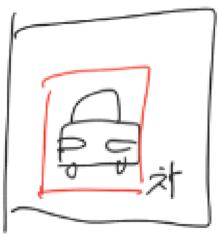
(p_c는 물체가 있는지에 대한 확률
$p_c = 0$ 이면 물체가 없는 배경임을 의미.

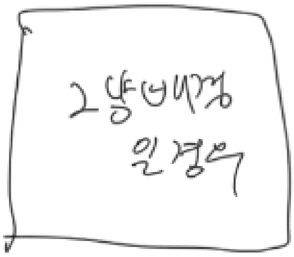
b_x, b_y, b_w, b_h는 경계상자에 대한 변수값이다.



자율주행 차라면, ex) C_1 = 보행자
C_2 = 차
C_3 = 오토바이

∴ 이 과정은 이미지가 단 한개의 물체만 포함한다는 가정

예제) img x = 



분류레이블 y

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_w \\ b_h \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_w \\ b_h \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

p_c가 0 이므로 경계상자 $b_{x,y,w,h}$ 와 물체 클래스 분류를 할 필요가 없어진다.

'don't care'

이렇게 소프트맥스층을 통해 얻은 분류레이블 y를 가지고

Mean Squared Error로 하면 $L(\hat{y}, y)$

실제값 예측값

$E = \Sigma (\hat{y}_i - y_i)^2$ → y의 경우

실제 $\hat{y}$ = 1이고

Pc역시 1이면 0

⋮

Pc 즉, 물체가 있는지 인식한 확률에 따라 손실함수 값이 다르다.

i) $P_c = 0$ (물체가 없다는 뜻)

그러면 $L(\hat{y}, y)$는

∴ $(\hat{y}_1 - y_1)^2$이된다.

왜냐면 물체가 없으니까

뒤에 출력값 신경쓸필요X

ii) $P_c = 1$ (물체가 있음)

$\sum (\hat{y}_i - y_i)^2$ 모든 출력값에 대한 오차제곱의 합

이렇게 나눠서 손실함수 계산.

실제로는 자세히 하면

- Pc ⇒ 로지스틱 회귀 손실함수 사용
- bx … bn 는 MSE 사용
- c1 … cn 은 소프트맥스 함수를 사용.

∴ Pc 즉 y_1의 값을 얼마나 정확히 계산하냐에 따라 손실함수 값이 좌우된다.

⇒ 물체 로컬리제이션 → 물체를 인지함을 넘어

물체의 위치가 어디있는지 파악해서

(경계 상자)를 이용한다.

예를 들어 소프트맥스층을 이용한 분류레이블을 클래스별로

p_c (물체가 있는 지 없는지)

b_x
b_y
b_w
b_h ← 물체의 위치를 파악하는 경계상자

c_1
⋮
c_n ⇒ 소프트맥스층이 물체가 무엇인지 분류하는 클래스 확률

이를 출력하고 이값을 p_c $\begin{cases} =0 \\ =1 \end{cases}$ 에 따라

손실함수를 계산 → p_c가 0이면 나머지 개체 항은 don't care 한다. (물체가 없으니까)

✕ 특징점 탐지
= landmark detection

→ ex) 얼굴사진에서 눈을 탐지하고자 한다면

각각 눈꼬리 양끝점 총 4개의 $\begin{matrix} l_{1x}, l_{1y} \\ l_{2x}, l_{2y} \\ l_{3x}, l_{3y} \\ l_{4x}, l_{4y} \end{matrix}$

을 출력 레이블로 나타내면된다.

CONV Network

softmax

l_{1x} l_{1y} l_{2x} ... l_{ny}

* 이것이 얼굴에서 감정을 인식하는 것의 기본적 설계단위이다.

⇒ 특징점 탐지 ⇒ 웃음을 통해 인식하고자 하는 특징점들의 각 좌표를 출력한다.

(특징점은 다른 이미지에 대해서도 항상 동일해야한다.)

↓

즉, 레이블의 순서는 다른 이미지에 대해서도 항상 동일해야한다.

* 물체 인식 (Object detection) 알고리즘 설계방법.

가장 먼저 label training set을 만든다.

Training set :

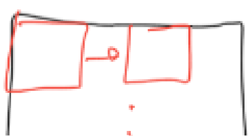
x	y
차	1
차	1
차	1
배경	0

이 레이블 훈련셋으로 CNN을 훈련시킬수있다.

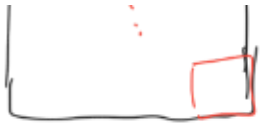
← Crop된 이미지를 사용해 확실하게 train

[차] → CONV Net → y

* sliding windows detection



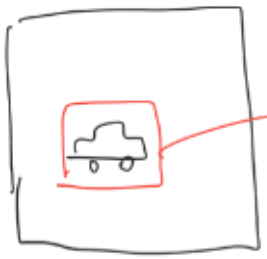
합성곱 신경망에 이 작은 사각형영역을 입력



□ 크기 window같은 이미지의 작은영역을 잘라서
합성곱 신경망에 통과시켜 예측값을 구한다.

이미지의 잘려진 부분에 대한 합성곱 신경망을 통과한
예측값은 각 위치에서 0 or 1의 값으로 분류된다.

* 한번의 sliding window과 오두 끝나면
그다음은 더 큰 □ 을 CONV Net에 통과시킨다



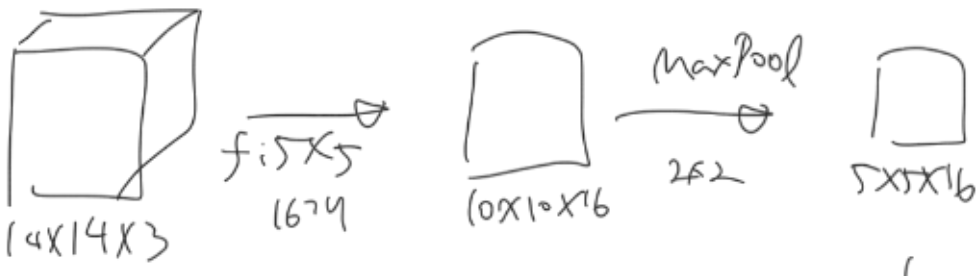
이렇게 차가 잘리면 1을 출력할

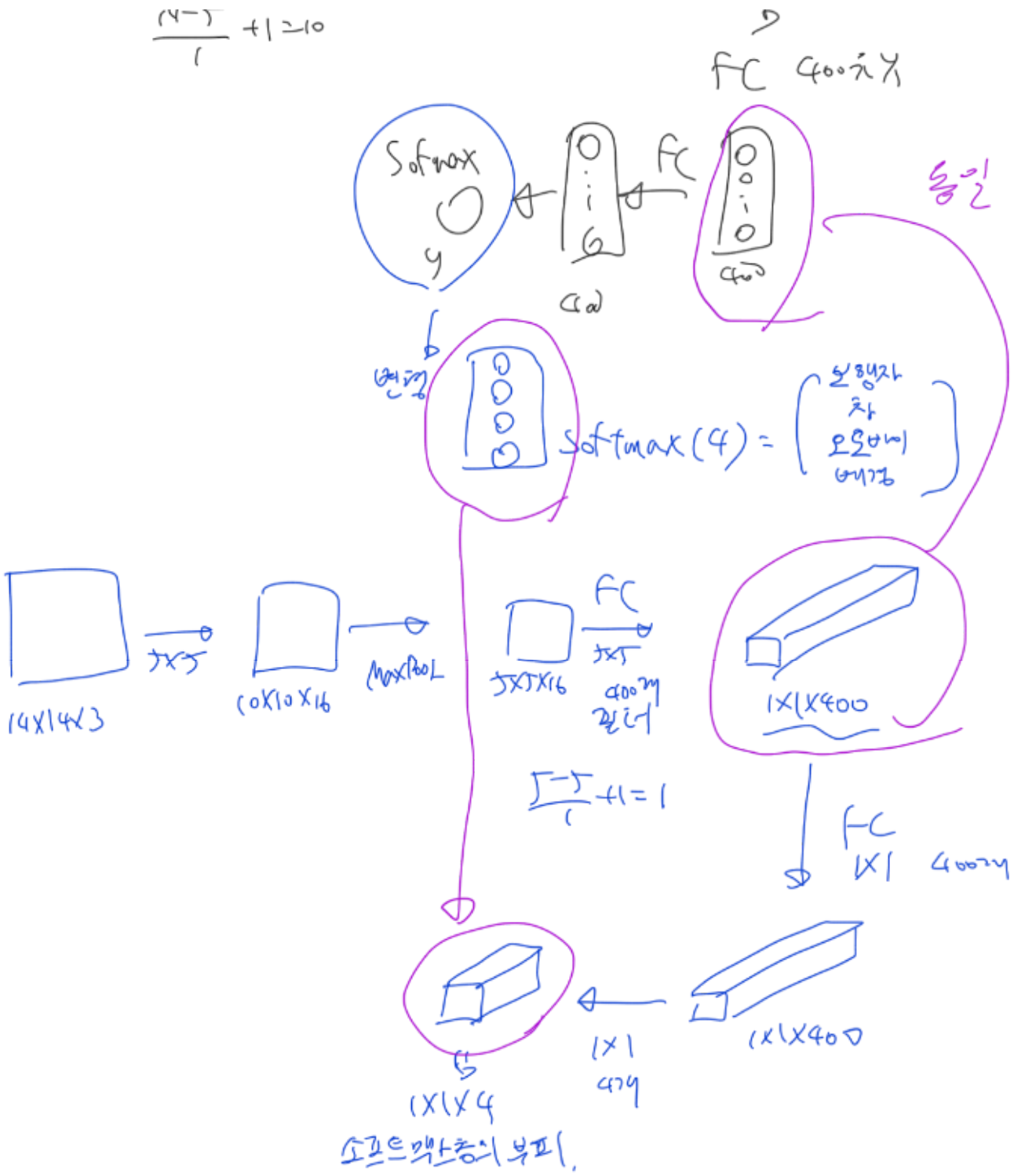
이를 sliding window detection 이라한다.

* sliding window 검출의
단점은 → 계산 비용이다.
↳ 두방은 작은영역 계산에 대한

* 해결 : 슬라이딩 윈도 알고리즘을 합성곱을 통해
구현해낼수 있다.

* Turning FC Layer into Convolutional Layers

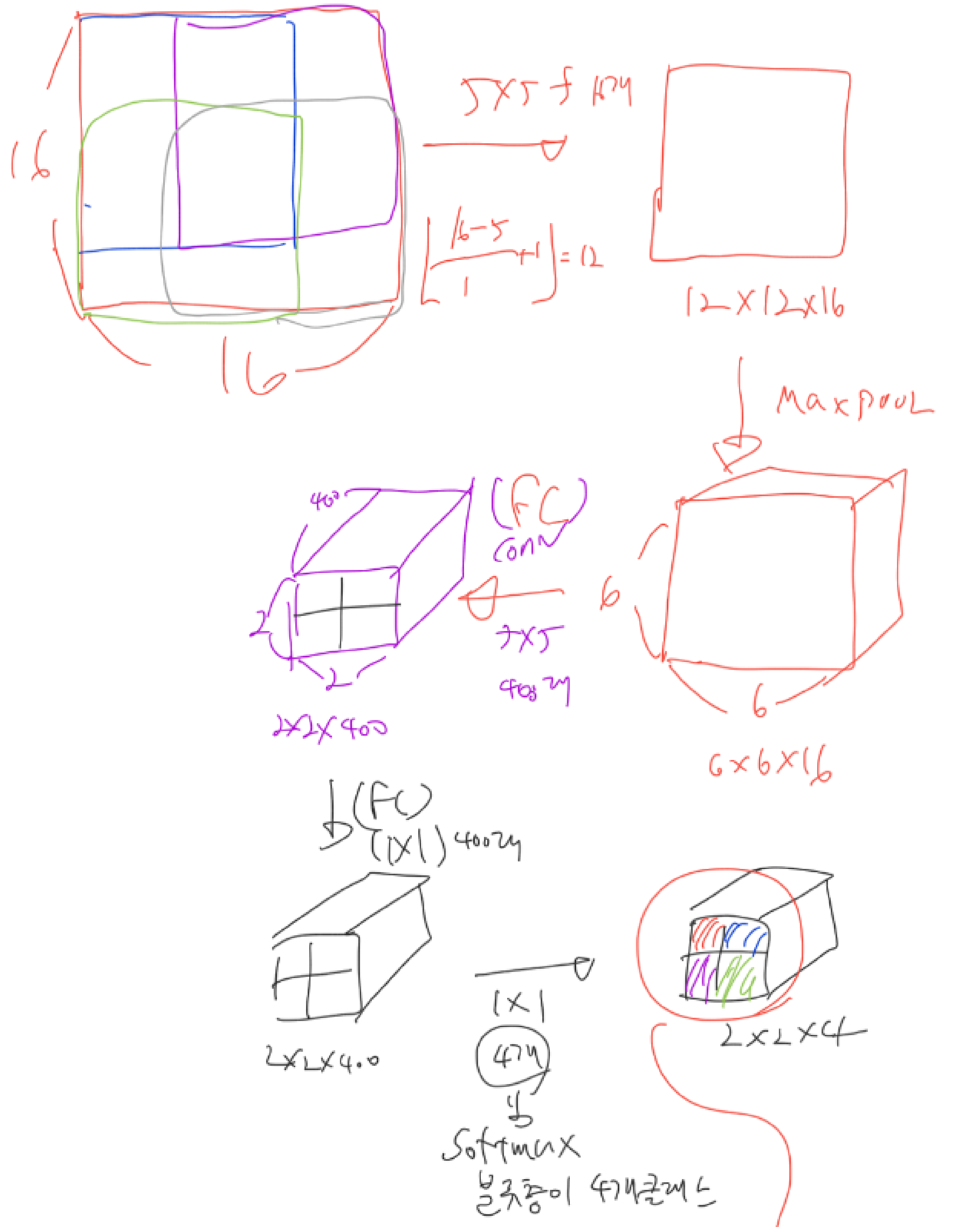




FC층을 이용해 합성곱 층을 구현해 낸 것.

슬라이딩 윈도를 합성곱 NN을 통해 구현함과 장점은
계산 변수를 공유한다는 것.

ex) 4개의 윈도를 슬라이딩하여 물체 검출 예시



이 4개 각 영역이
슬라이딩 윈도 4번 하여 나온 값이다.
즉 4번 계산한 것을 합성곱을 통해
1번 계산으로 다 할 수 있다.

마지막 수정: 오후 5:08