# 4p pytorch 모델 클래스화 하기

*- Class 클래스명(상속받는 클래스)*

NN이라는 class명을 정의한다.

NN클래스는 torch.nn.Module을 상속받는다. 모든 함수와 변수를 활용할 수 있다.

- super(NN,self).\_\_init\_\_()

nn.Module 클래스의 모든 속성을 그대로 받아서 나 자체도 초기화 하겠다는 의미

*- init과 forward함수*

def: 함수화

*- def\_\_init\_\_(self):*

나 자신에 대한 linear1변수를 정의.

클래스 내부에서의 *self*는 나 자신을 의미한다. =NN

클래스 안에서 정의하고 있기 때문에 계속 self를 붙여야 한다.

*- def Forward (self, x):*

: 명시적으로 입출력을 정의해줌으로써 모델의 방향성을 설정.

NN모델클래스가 호출되면 자동으로 함수가 수행되어 모델을 정의한다.

FClayer는 초기화에 따라 학습에 영향을 미치기 때문에 모델안에서 초기화하는 것이 좋다.

🡪 모델 정의 시, 그냥 NN 클래스 호출하기!

(<->기존에는 Sequential을 통해 모델을 정의했다.)

# CNN 모델 구현방법

데이터 로드 🡪 모델 정의 🡪 모델 학습 🡪 모델 평가

모델 정의가 계속 바뀌고 있다. 모델: NN🡪 DNN 🡪 CNN

View를 통해 2D 데이터를 1D로 바꾸고 SoftMax와 NLL loss를 통해 결과를 도출한다.

실험적으로 파라미터를 변경하며 모델을 재설계하다 보면 가장 optimal한 모델을 발견하게 된다.

Kernel size와 padding의 관계, stride와의 관계 등 변수들이 입출력에 미치는 영향을 알아야 한다.

**CNN모델 설계 배우기!** (모델 학습과 평가 부분은 이전 그대로 사용한다.)

1. 1ver: 순서대로 CONV, ReLU, MaxPOOL을 정의하는 방법

- Conv1= 입력1, 출력32,

커널 size=3이지만 padding=1 이기 때문에 28\*28 🡪 28\*28 줄어들지 않고 그대로 유지된다.

- Maxpoo1= kernel\_size=2, stride=2이므로 28\*28 🡪 14\*14 로 줄어든다.

- Fc= FC layer에 대한 initialize

\* Forward: 정의한 모델을 연결

Layer1, layer2에서는 2D데이터를 받아서 처리하고 출력했다.

하지만 FC layer는 1D데이터를 받아서 1D데이터를 출력한다.

그래서 **View를 통해 2D데이터(MaxPool에서 받은)를 1D로 바꿔준다.**

Out=out.view(out.size(0),-1)

- out.size(0), -1:

입력된 out= maxpool2의 결과물 [배치크기x7x7x64] (64: conv2 32가 64채널로 나오도록 설계되어 있어서)

Out.size(0)은 batchsize를 의미. ex)NN=32이면 batchsize는 32.

-1은 나머지 전부를 의미. 7\*7\*64

🡪 out.view(): batchsize와 7\*7\*64로 데이터를 변형시킨다.

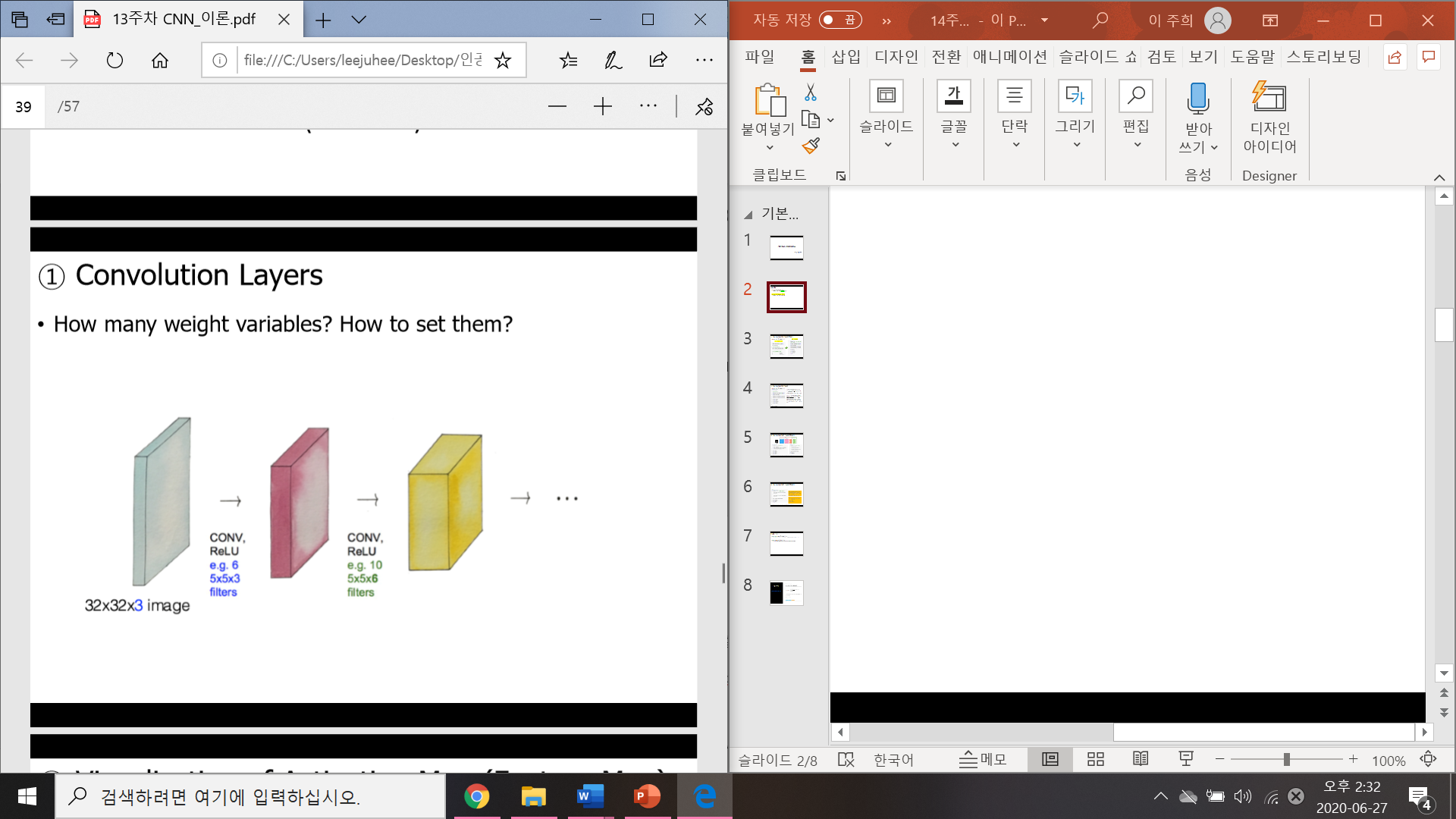
1D데이터(7\*7\*64)가 batchSize만큼 존재하게 된다.

🡪 FC layer는 7\*7\*64 1D를 받아서 10을 출력한다. (0~9,10개)

세로: 7\*7\*64=3136

가로: 1

1D데이터가 batchsize만큼 존재.



W=7, H=7, depth=64의

2D데이터가 batchsize만큼 존재.

🡪 이를 1D데이터로 변경.

1. Ver2: 모델의 모듈화