# Al Lab Newton 으로 배우는 인공지능

The Deep Learning Solutions for Students



Last Update : 2024/2/10
www.ailabnewton.com

차례

## I. AI Lab Newton 요약 Ⅱ. 인공지능 아이디어 Ⅲ. 데이터 변환(버튼과 매개변수) Ⅳ. 레이어 패널 Ⅴ. Open & Save 패널 Ⅵ. Training 패널

**APPENDICES** 

## I. Al Lab Newton 요약

## 1. 로그인 화면

## Al Lab Newton



#### 프로그램을 시작하면 좌측과 같은 로그인 화면이 보여진다(인터넷 연결되어 있어야 한다).

(1) 선생님으로 받은 학교 비밀번호
(2) 이멜 ID (<u>myID@gmail.com</u>에서 myID)
(3) 개인 비밀번호를 입력하고

입장(5) 버튼을 클릭하면 입장 된다. 만약 오류가 있으면 메시지(4) 가 출력된다. 2. 시작 화면

## Al Lab Newton 3.1 Start (1) Finish (2) Info (3)

입장이 되면 좌측과 같은 시작 화면이 보여진다.

(1) "Strat" 버튼은 프로그램 시작 버튼이다.
(2) "Finish" 버튼을 클릭하면 프로그램이 종료된다.
(3) "Info" 버튼을 클릭하면 새로운 버전이 있으니 다운로드 받으라는 등 본 프로그램과 관련된 정 보가 아래 처럼 출력된다. 주기적으로 "Info" 버 튼을 클릭하여 새로운 정보가 있는지 확인하는 것이 바람직하다.



## 3. 메인 화면



앞 시작 화면의 "Start" 버튼을 클릭하면 좌측과 같은 메인 화면이 보여진다. 여기에서 인공지능 을 실습한다.

 좌측 하단의 "Scenes" 버튼으로 화면의 Scene 을 변경할 수 있다. 현재 8개의 Scene 들이 주어져 있는데 각자 마음에 드는 것을 선택할 수 있다.
 좌측 상단에 "Key Idea" 버튼이 있다. 인공지능의 원리 및 메인 화면의 각 버튼들이 무엇을 의미하 는지 3개의 화면에 직관적으로 보여진다.

## 4. Key Idea 버튼



#### [사람의 인식]



#### We distinguish between cats and dogs. How? Decision Input information making Information (1)processing in the brain 단순화한 구조 뉴런의 신경구조 (3)(2)

"Key Idea"을 클릭하면 좌측에 0, 1, 2 세개의 번호가 보여진다. 여기서 "0" 버튼을 클릭하면 좌측의 화면 이 보여진다.

사람은 강아지와 고양이를 어떻게 구분하나?

- 1) 눈으로 입력된 정보를 뇌에서 정보 처리를 하여 강아지인지 고양이 인지 판단을 한다.
- 정보 처리 하는 뉴런의 신경 구조를 현미경
   으로 촬영한 이미지이다.
- 3) 뉴런의 신경 구조를 단순화한 이미지이다.

#### [사람의 인식 과정을 모방하는 인공지능 아이디어 ]



사람의 뇌는 입력되는 정 보(데이터)를 받아 정보 를 처리(신경 구조)하여 판단한다

인공지능은 입력되는 정 보(데이터)를 받아 정보 를 처리(변환 구조)하여 판단한다

#### "1" 버튼을 클릭하면 아래의 화면이 보여진다.



보통 이미지가 칼라 이미지(2c) 이면 R, G, B 가 합쳐진 것이다.

학습은 CPU 보다 GPU (3) 에서 해야 빠르게 한다. 학습하는 방법은 Optimizer (4)라 부른다. 학습하는 속도와 관계되는 learning rate (5)를 미세 조절하여 학습이 잘되게 조절할 수 있다. 학습되어 잘 다듬어진 변수들(1b)을 모아 놓은 것을 Trained data (6) 이라 부른다.



Idea

Class Num	10				
Epoch Num	500				
Batch Num 64					
Learning Rate	0.002				
Training Data F	older				
None					
Trained Data Folder					
None					
Optimizer					
Adam 🗸					
Prediction Data	Folder				
	Contract of the local division of the local				









InputData	Activation	Dense	Layer
	AddSC	Padding	
	BN	Poolng	
UDM	CONV	Reshape	

Confirm



Init Draw	Closer	Wider	5	C
	the second se			

Replace



#### "2" 버튼을 클릭하면 아래의 화면이 보여진다.



학습된 Trained data 를 이용하여 실제적 예측할 데이터(Prediction Data) 에 대하여 예측을 한다.

II. 인공지능 아이디어

## 1. 우리가 고양이와 강아지를 구분하는 방법은?

1-1. 고양이와 강아지를 어떻게 구분하는가?



질문 : 사람은 고양이와 강아지를 딱 보면 구분을 할 수 있는데 어떻게 구분할까?

인지과정: 우리 눈을 통하여 입력된 고양이와 강아지 이미지는 시신 경을 통해 뇌로 전달되고, 뇌에서는 어릴 적 부터 경험하고 보아 왔 던 기억들을 참고해서, 강아지와 고양이의 특징들에 대한 정보들이 처리되어 판단한다.

결론 : 뇌로 입력된 이미지 정보는 뇌에서 어릴 때 부터 경험했던 기 억들을 참고하여 정보 처리를 한 후 판단을 하게 된다고 요약을 할 수 있다.



#### 1-2. 우리 뇌의 판단 과정과 비슷하게 컴퓨터 프로그램을 만들면...?

(1) 그렇다~! 그렇게 만든 것이 최근의 인공지능 프로그램이다

(2) 본 AI Lab Newton 소프트웨어는 아주 쉬운 방법으로 이를 구현한 것이다

#### 1-3. 그러면 좀 더 구체적으로 인공지능의 아이디어는 ?



## 2. 인공지능에서 정보 입력 방법은?

#### 2-1. 정보 입력은 어떻게 하나?

사진이란 0~255 값들이 각 픽셀에 카메라의 이미지 센서 각 점(소위 픽셀)에 고양이로 부터 나온 빛이 쌓여 밝기에 따라 0~255 까지의 값들로 표시된 이미지이다. 이 숫자들 을 입력



2-2. 아래 사진의 고양이 사진 정보는 몇개의 숫자로 표현될까?



즉, 17x15의 3 배 즉 17 x 15 x 3 = 765 개의 숫자로 표시된다~!

### **3. 인공지능에서 정보처리는?** 3-1. 인공지능에서 "정보처리 "는 구체적으로 무엇인가?



#### 3-2. "변환 " 의 대표적 예를 하나 들면?

예를 들어, 765 개의 픽셀의 각 값 x[i] 들에 어떤 값 w[i, j] 들을 곱하여 죽 더하면 새로운 값 y[j] 를 하나 얻을 수 있다.(여기서 j =0, ..., 999라 하자)



```
y[0] = x[0]w[0,0] + x[1]w[1,0] + ... + x[674]w[764,0]
y[1] = x[0]w[0,1] + x[1]w[1,1] + ... + x[674]w[764,1]
```

y[999] = x[0]w[0,999] + x[1]w[1,999] + ... + x[764]w[764,999]

#### [간단한 계산 예]

<계산 편의상 가정>

20

131

34

126

15

23

고양이

강아지

- (1) 고양이와 강아지가 계산 편의상 3 픽셀로 되어 있다.
- (2) 앞의 고양이 픽셀값(노드값)은 20,131,34 이고 강아지 픽셀값은 126, 15, 23 이다

y[0] = 20x2 + 131x1.4 + 34x0.7 = 247.2

y[1] = 20x0.12 + 131x2.5 + 34x5.6 = 520.3

y[0] = 126x2 + 15x1.4 + 23x0.7 = 289.1

v[1] = 126x0.12 + 15x2.5 + 23x5.6 = 181.42

- (3) 에지값(weights)은 2, 1.4, 0.7 과 0.12, 2.5, 5.6 이다

이러한 가정하에 계산을 해보면 다음과 같다

0.12

2.5

5.6

0.12

2.5

5.6

- (4) 두번째 레이어의 노드 갯수를 2개(y[0], y[1])이다

2

1.4

0.7

2

1.4

0.7

## 4. 인공지능에서 판단은?

#### 4-1. 인공지능에서 "판단 " 은 어떻게 할까?







31

201832131148172344728

고양이와 강아지의 픽셀 갯수는 여기서도 계산 편의상 픽셀 3개로 표시된다고 가정하 고, 각각 픽셀값을 좌의 그림 처럼 가정하자.

또한 에지값(weights)은 2, 1.4, 0.7 과 0.12, 2.5, 5.6 로 가정하고 y[0], y[1] 을 계산해 보 자.



126

15

23



41





[계산 결과]



20

131

34







324.4 590.6 28

weights 를 다음 처럼 놓고

w[0,0] = 2, w[1,0] = 1.4, w[2,0] = 0.7w[1,0] = 0.12, w[1,1] = 2.5, w[2,1] = 5.6

다음 값을 계산하면, y[0] = x[0]\*w[0,0] + x[1]\*w[1,0] + x[2]\*w[2,0] $y[1] = x[0]^*w[0,1] + x[1]^*w[1,1] + x[2]^*w[2,1]$ 

289.1

181.4

126

15

23

247.2

520.3





484.5

375.3

좌측의 빨간색 값이다

관찰: 계산된 결과를 자세히 들여다 보면, 강아지들은 모두 y[0] 값이 크고, 고양이들은 y[1] 값이 크다.

결론 : <u>따라서, y[0] 값이 크면 강아지 이고 y[1]</u> 가 크면 고양이 라고 판단한다.



160 386.5 27 316.3 41

181 72 31

## 5. 학습한다는 뜻은?

5-1. 그러면 인공지능에서 weights 는 어떻게 구할까?



(1) 앞에서 w[0,0], w[1,0], ..., w[2,1] 등을 주어진 것으로 가정하고 y[0], y[1] 을 계산하였고, y[0] 이 더 크면 강아지이고 y[1]이 더 크면 고양이라고 판단하였다.

(2) 그러면, weights w[0,0], w[1,0], ..., w[2,1] 등을 어떻게 구할까?

답 : 다수의 고양이 사진과 강아지 사진들(예: 각 1 만장)에 대하여 강아지 사진들은 y[0] 값이 크게 하고 고양이 사진들에 대해서 는 y[1] 값이 크게 하는 w[0,0], w[1,0], ..., w[2,1] 를 구하는 것이다.

즉, 다음 식에서 여태까지는 weights들을 안다고 하고 y[0], y[1] 을 구했지만 이제는 y[0], y[1]을 안다고 하고 weights를 구하는 것이다.

 $y[0] = x[0]^*w[0,0] + x[1]^*w[1,0] + x[2]^*w[2,0]$  $y[1] = x[0]^*w[0,1] + x[1]^*w[1,1] + x[2]^*w[2,1]$ 

참고: 보통 weights 의 갯수가 많아 solutions 이 여러 개 가능하다. 이 중 가능한 임의의 이미지에 대해 가장 정확도가 큰 것을 solution으로 택한다

#### 5-2. 그러면 인공지능에서 "학습한다" 는 용어의 정체는?





(a) 아무렇게나 변환하면 보통은 weights 가 구하기 어렵다.
(b) 주어진 공통의 이미지 데이터들을 이용하여 weights를 잘 구하는 변환에 대해 최근 2010년 부터 세계인들이 모여 경쟁적으로 대회를 하였다. 이를 통하여 중요한 변환들에 대해 알게 되었고 오늘날 인공지능이 꽃을 피기 시작하였다.

\*대회 이름 : ImageNet Large Scale Visual Recognition Challenge(ILSRC)

#### 5-3. Dense 말고 또 다른 중요한 변환의 예가 무엇이 있나?

- (1) CONV (Convolution) 등등이 있는데 본 소프트웨어의 인터페이스를 보면 쉽게 알 수 있다
- (2) 이제 본 소프트웨어의 인터페이스를 사용하는 방법에 대하여 설명하며 변환들에 대하여 알아보 자(AI Lab Newton 으로 배우는 인공지능 III. 데이터 변환 참고)





중간 변환들이 2개 이상 많으면 deep 하다 고 말한다(deep learning)

# III. 데이터 변환 (버튼과 매개변수)

데이터 변환 = 버튼 및 매개변수를 이용하여 변환 노드들 = 데이터 변환 결과 얻은 값들 레이어 = 노드들 모아 놓은 것

## [버튼 패널 및 매개변수 패널]

InputData	Activation	Dense	Layer
	AddSC	Padding	
	BN	Poolng	
	CONV	Reshape	
			Confirm

버튼 패널에 InputData 버튼을 포함하여 9개의 버튼이 있다.

- InputData 버튼은 처음 입력한 데이터들을 나타내고 나머 지 버튼들은 앞 데이터를 변환하여 새로운 데이터를 만드 는 수단이다.
- 처음 입력한 데이터 및 변환된 데이터들을 **레이어**라고 부 른다.
- 따라서 변환 구조는 레이어 구조이다.

버튼 및 매개변수 패널에서 Confirm 된 변환을 "Add" 버튼을 이용하여 레 이어 패널에 순서대로 쌓는다. "Delete", "Replace" 등의 버튼을 이용 하여 수정도 가능하다. 추후 더 자세 히 설명한다.



[레이어 패널]

## 1. InputData





(1)

InputData	Activation	Dense	■ InputData ny nz ch 30 30 3 (2)				
	AddSC	Padding					
	BN	Poolng	ny : number of nodes in y-direction nz : number of nodes in z-direction (3				
	CONV	Reshape	ny, nz, ch > 0				

Confirm

모든 변환은 입력되는 데이터를 변환하 는 것이므로 InputData 부터 시작한다. 즉 위의 고양이 이미지를 넣어주는 버 튼이다.

(1) Input Data 입력 버튼
(2) 매개변수들 : ny, nz, ch
(디폴트값 : 30,30,3)
(3) 매개변수들 설명
(4) Confirm 버튼을 클릭하면
(5) 의 위치에 다음과 같이 버튼 이 름과 매개변수 값들이 기록된다 InputData 30 30 3



(4)

앞의 InputData 에 대하여 Confirm 버튼을 클릭한 후 우측 패널(레이어 패널이라 부 른다) 에서

- (1) "Add" 버튼을 클릭하면
- (2) Confirm 된 정보가 레이어 패널에 들어 간다. 레이어 패널의 첫 레이어는 항상 InputData 레이어이어야 한다.
- (3) 레이어 패널에 쌓인 레이어들은 3D 로 보여진다.

#### 3D 공간에 보여지는 3D 레이어 C Ⴢ Closer Wider Init Draw (2) InputData 30 30 3 레이어 패널에 쌓인 첫 레이어 Delete All Add Delete Replace Insert (1)

(3)

- (1) Alt + ML (Mouse Left button) 을 누른 상 태에서 움직이면 회전 시킬 수 있다
- (2) 마우스를 3D 이미지에 두고 Ctrl + ML 클 릭하면레이어를 이루고 있는 노드들을 볼 수 있다. ny : 세로 방향 노드 수, nz : 가로 방향 노드 수, ch : 이런 사각형 노드 모은 것 3개 를 나타낸다. 여기서 (ny, nz, ch)를 Shape(형상)라 부른다.
- (3) InputData 레이어는 입력되는 칼라 데이 터 (앞의 예 고양이 이미지)의 R, G, B 를 나타낸다.

[참고] 흑백 이미지는 채널 ch = 1



## [ 참고: 단축키]

(1) 마우스 휠(MW) 을 클릭하면 단축키 요약을 볼 수 있다다 시 마우스 휠 을 클릭하면 단축키 요약이 사라진다
(2) Alt + ML 하고 마우스를 움직이면 카메라가 회전한다
(3) Shift + MW 하고 W(휠)을 돌리면 카메라 줌(빠르게)이 된다
(4) Shift + ML 하고 마우스 움직이면 카메라가 이동, 선택 레이 어 취소(2 기능)
(5) Ctrl + ML 하면 3D 레이어가 선택된다

- \* 선택된 레이어의 글씨는 초록색으로 바뀌며
- \* 화면 상단에 (ny, nz, ch) 이 출력된다.

(6) Ctrl + Up/DownArrow 하면 사운드 조절된다

MW : 단축키 보여주기 (토글) Alt + ML : 카메라 회전 Shift + MW : 카메라 줌 (빠르게) Shift + ML : 카메라 이동, 선택 레이어 취소 Ctrl + ML : 레이어 선택 Ctrl + Up/DownArrow : 사운드 조절

## [ 참고: 단축키 보충 설명]

(4) Shift + ML 하고 마우스 움직이면 카메라가 이동, 선택 레이 어 취소(2 기능) (5) Ctrl + ML 하면 3D 레이어가 선택된다

\* 선택된 레이어의 글씨는 초록색으로 바뀌며

\* 화면 상단에 (ny, nz, ch) 이 출력된다.



## 2. Dense 변환

(1)



Dense 변환은 뒤 레이어의 노드 값을 앞 레이어의 각 노드 값에 웨이트를 곱하여 더하는 변환이다. Dense 레이어의 shape 는 (ny, 1, 1) 이다.

(1) Dense 버튼을 클릭하면

(2)

(3)

(2) 매개변수가 3개가 보여진다. 디폴트 는 10, 1, 0 이다.

(3) Confirm 버튼을 클릭하고

(4) 레이어 패널에 "Add" 버튼을 클릭하 여 넣어주면

(5) 3D 이미지가 생성된다.

10개 노드 ny 는 뒤 레이어의 세로(y) 방향 노 드 개수를 나타낸다.

> 이제 다른 매개변수는 무엇을 의 미하는지 살펴보자



## 2-1. Dense 변환에서 bias 란?

Delete All

Insert



Replace

Add

Delete



불필요하게 복잡한 경우를 피하고, 좀 더 간단 한 경우 살펴보자.

- (1) 다시 InputData 버튼을 클릭한 후 매개 변수 를 5, 1, 1 로 변경하고 Confirm 한다.
- (2) 레이어 패널에서 InputData 30 30 3 인 위치 를 마우스로 클릭한 후
- (3) "Replace" 버튼을 클릭하면 Input 5 1 1 로 바 뀌며

(4) 3D 이미지도 (4) 처럼 변경된다.

(5) 이때 두번째 3D 이미지에 마우스를 놓고 Ctrl + ML 을 클릭하면

(6) 처럼 두번째 이미지의 노드 10개가 보이고, 첫번째 노드와 연결된 흰색의 웨이트(에지) 들이 보인다.





그림에서 흰색의 웨이트를 위에서부터 각각 w[0,0], w[1,0], ..., w[4,0] 이고 파란 점(값은 1)과 연결된 웨이트를 w[0] 이라 하면

y[0] = x[0]\*w[0,0] + x[1]\*w[1,0] + ... + x[4]\*w[0,4] + w[0] 로 계산하면 된다. y[1] = x[0]\*w[0,1] + x[1]\*w[1,1] + ... + x[4]\*w[0,1] + w[1] 이 된다.

y[2], y[3], ..., y[9] 모두 같은 방법으로 구한다. 여기서 파란 점을 bias 라 부른다.



만약, bias 는 디폴트로 1이지만 만약 이것을 0 으로 두면 값이 1 인 파란 점은 사라진다. 즉, y[0], y[1], ...은 y[0] = x[0]\*w[0,0] + x[1]\*w[1,0] + ... + x[4]\*w[0,4] y[1] = x[0]\*w[0,1] + x[1]\*w[1,1] + ... + x[4]\*w[0,1]

등이 된다.

[참고] Appendix I. Dense 변환의 수학적 표현 (굳이 알 필요 없습니다. 개념이 중요합니다.)

## 2-1. Dense 변환에서 dropout 란?



dropout 이 0.2 이면 앞 레이어의 노드 5개 중 20% 를 없는 것 처럼 취급한다. 즉, 앞 레이어 노드 5개 중 랜덤하게 20%인 1개를 선택되어 그 노드 값을 0 으로 가정한다.

즉 해당 weight 값이 아니라 **노드 값을 0**으로 본다. 디폴트 0 인 경우는 0 인 노드가 한 개도 없다는 뜻이다. 1 Epoch 동안 유지되면 **1 Epoch가 지나면 랜덤으로 다시** 50% 만큼을 0 으로 놓는다(why? 그래야 학습이 잘된다고 알려져 있다)

[예] 앞 레이어의 shape가 (10,10,1) 인 경우 dropout = 0, 0.5, 0.8 인 경우



## 3. CONV(Convolution) 변환

모든 레이어들의 shape 와 그 레이어를 구성하는 노드 값들을 구하는 것이 데이터 변환의 핵심이다. CONV 변환은 인공지능의 발전을 가능하게 했던 가장 중요한 변환 중 하나이다.

## 3-1. CONV 레이어 shape 계산 방법은?

[예] CONV 레이어의 앞 레이어를 (1) InputData 레이어 shape = (5, 5, 3) 으로 하고 (2) CONV 레이어 CONV 2 2 10 0 1 1 0 으로 하자. 그러면 (3) CONV 레이어의 shape = (4,4,10) 이다.

InputData	Activation	Dense		<b>h</b>	W 2	<b>ch</b>	padding
	AddSC	Padding		sy 1	SZ 1	bias	(2)
	BN	Poolng	h : filter size in y-direction (2: default) w : filter size in z-direction (2: default) ch : number of output channels (3)				
	CONV	Reshape					
		sy : stride in the y-direction (1:default) sz : stride in the z-direction (1:default)					
InputData 5	553	1)	Dias . 0, 1	(1.ueiau			
CONV 2 2	100110 (	2)	Confirm	CONV 2	2 10 0 1	11	(2)

(3)

CONV 레이어의 채널 ch 은

3번째 매개 변수 값~!


#### 편의상 앞 레이어 노드들을 작은 사각형으로 그렸다



sy=1, sz=1 (stride 라고 부른다)로 부터 세로 방향(y 방향)으로 1씩 가로 방향(z 방향)으로 1씩 움직이며 y[j, k, s] 를 계산한다.

따라서, shape 는 (ny, nz, ch) = (4, 4, 10) 이 된다.

관찰: 채널ch = 10은 내가 선택하지만 (ny, nz) = (4,4) 는 앞 레이어와 CONV 매개 변수에 의해 결정된다.



CONV 레이어



### 3-2. padding 이란?

[예] CONV 레이어의 앞 레이어를 (1) InputData 레이어 shape = (5, 5, 3) 으로 앞과 동일하게 하고 (2) CONV 레이어 CONV 3 2 10 1 2 1 0 으로 하자. 그러면 (3) shape = (3,5,10) 인 CONV 레이어가 그려진다.

CONV 레이어의 shape = (3,5,10) 은 우측의 매개 변수로 부터 구하는 방법을 살펴보자.

ch = 10 은 주어진 것. (ny, nz) 구하자.

이 문제의 풀이는 padding = 1일 때, 노드들을 계산하는 방법과 관련 있다.

(가) padding = 0 이면, 가로(z) 방향으로 w = 2, sz = 1 이면 앞 예와 동일하 게 nz = 4가 될 것이다.

(나) padding = 1 이면, sz = 1 이니까 4각형 가로 방향으로 시작 위치가(화 살표)가 놓일 수 있는 위치는 5개 있다. 그런데 마지막은 한 줄 부족. 따라 서 한 줄 덧 붙인다



덧 붙인 것을 pad 라 한다. padding = 1이라는 것은 pad 를 덧 붙일 수 있다는 뜻이다. 즉 노 드가 있는 것 처럼 덧 붙이는데 노드 값은 0 으로 둔다. 위 그림 (1) 에서 우측 노란색 노드 들로 표시되었다.

h = 3	
w =2	
ch = 10	
padding = 1	
sy = 2	
sz = 1	
bias = 0	
	_



#### (다) padding = 0 이면, 세로(y) 방향으로 h = 3, sy = 2 이면 2번 이동하면 끝이다. 이 경우 nz = 2 가 된다. 만약 h = 4 이면 1번 밖에 이동 못하고 나머지들은 모두 버린다. 이 경우는 nz = 1 이 된다





h = 4, sy = 2

 $\square$ 



h = 3, sy = 2

(라) padding = 1 이면, sy = 2 이니까 4각형 세로 방향으로 시작 위치가(화살표)가 놓일 수 있는 위치는 3 개 있다. 그런데 마지막은 2줄 부족(A). 따라서 2줄 덧 붙인다(B). 덧 붙이는 규칙은 줄의 개수를 똑같이 나누어 아래 위에. 만약 하나 남으면 아래 쪽에 덧붙인다. 가로 방향도 동일하다. 한 줄 남으면 우측에 덧붙인다.

padding = 1 이면, 만약 h=4 이면 sy = 2 이니까 4각형 세로 방향으로 시작 위치가(화살표)가 놓일 수 있는 위치는 3개 있다. 그런데 마지막은 3줄 부족(C). 따라서 위 1줄 아래 2줄 덧 붙인다(D).





#### 3-3. bias 란?

CONV 레이어에서 bias 는 Dense 레이어의 bias 개념과 동일하다.





Dense 레이어와 마찬가지로 CONV 레이어에서도 bias = 1 인 경우 노드 값이 1인 추가 노드가 더해진다.

[참고] Appendix I. CONV 변환의 수학적 표현 (굳이 알 필요 없습니다. 개념이 중요합니다.)

## 4. Pooling 변환

일반적으로 Pooling 변환은 앞 레이어의 크기 (ny, nz)를 줄이는 변환 이다. 따라서 앞 레이어의 크기를 줄이는 목적으로 사용한다.

### 4-1. Pooling 레이어 shape 계산 방법은?

[예] Pooling 레이어의 앞 레이어를 (1) InputData 레이어 shape = (5, 5, 3) 으로 하고 (2) Pooling 레이어 Pooling 2 2 0 0 2 2 로 하자. 그러면 (3) Pooling 레이어의 shape = (2,2,3) 이다

1.	Pooling 레이어의 채널 ch 는
	앞 레이어와 동일
2.	매개 변수 : h, w, padding, sy, sz
	는 CONV 변환과 동일한 개념
	는 CONV 변환과 동일한 개념





h = 2 w =2 N = 0padding = 0sy = 2 sz = 2

Pooling 레이어의 shape = (2,2,3) 은 좌측의 매개 변수로 부터 어떻게 구하나? (N = 0 경우)

N = 0 (Max Pooling) 는 앞 레이어의 (h, w) 에 대해 가장 큰 값을 선택하는 것



N=0은 세로 h=2, 가로 w=2 박스에서 가장 큰 값을 택한 것이 Pooling 레이어의 노드 값들이다. 다음은 Pooling 레이어 (1) 을 구하는 방법을 보여준다

0.1	0.9	0.2	0.5	0.3
0.8	0.5	1.2	1.1	0.7
1.3	1.6	0.1	2.1	1.3
2.3	0.9	1.2	3.2	3.2
0.2	1.4	0.3	1.3	2.2

0.1	0.9	0.2	0.5	0.3
0.8	0.5	1.2	1.1	0.7
1.3	1.6	0.1	2.1	1.3
2.3	0.9	1.2	3.2	3.2
0.2	1.4	0.3	1.3	2.2

0.1	0.9	0.2	0.5	0.3
0.8	0.5	1.2	1.1	0.7
1.3	1.6	0.1	2.1	1.3
2.3	0.9	1.2	3.2	3.2
0.2	1.4	0.3	1.3	2.2

0.1	1 0.9	0.2	0.5	0.3
0.8	3 0.5	1.2	1.1	0.7
1.3	3 1.6	0.1	2.1	1.3
2.3	3 0.9	1.2	3.2	3.2
0.2	2 1.4	0.3	1.3	2.2

(1) 0.9 1.2 2.3 3.2

N=0

h = 2 w =2 N = 1 padding = 0 sy = 2 sz = 2

Pooling 레이어의 shape = (2,2,3) 은 좌측의 매개 변수로 부터 어떻게 구하나? (N = 1 경우)

N = 1 (Average Pooling) 는 앞 레이어의 (h, w) 에 대 해 평균 값을 선택하는 것



N=1은 세로 h=2, 가로 w=2 박스에서 평균 값을 택한 것이 Pooling 레이어의 노드 값들이다. 다음은 Pooling 레이어 (1) 을 구하는 방법을 보여준다

0.1	0.9	0.2	0.5	0.3
0.8	0.5	1.2	1.1	0.7
1.3	1.6	0.1	2.1	1.3
2.3	0.9	1.2	3.2	3.2
0.2	1.4	0.3	1.3	2.2

0.1	0.9	0.2	0.5	0.3
0.8	0.5	1.2	1.1	0.7
1.3	1.6	0.1	2.1	1.3
2.3	0.9	1.2	3.2	3.2
0.2	1.4	0.3	1.3	2.2

(0.1+0.8+0.9+0.5)/4 =0.575 (0.2+1.2+0.5+1.1)/4 =0.75 (1.3+2.3+1.6+0.9)/4 =1.525 (0.1+1.2+2.1+3.2)/4 =1.65

(1)

0.575 0.75

1.525 1.65

 0.1
 0.9
 0.2
 0.5
 0.3

 0.8
 0.5
 1.2
 1.1
 0.7

 1.3
 1.6
 0.1
 2.1
 1.3

 2.3
 0.9
 1.2
 3.2
 3.2

 0.2
 1.4
 0.3
 1.3
 2.2

	0.1	0.9	0.2	0.5	0.3
	0.8	0.5	1.2	1.1	0.7
	1.3	1.6	0.1	2.1	1.3
	2.3	0.9	1.2	3.2	3.2
	0.2	1.4	0.3	1.3	2.2

N=1

#### 4-2. padding = 1 인 Pooling 레이어 예





**예**: 앞레이어 shape = (5,5,3) 이고 h = w = 3, sy = sz = 2, N = 0, padding = 1 이다.

앞 레이어에서 가능한 시작 위치(화살표)가 3개인데 노드가 부 족하다(A). 가로 세로 모두 추가 2줄이 필요하다.

따라서 Pooling 레이어 shape = (3,3,3) 이다(1) 위아래좌우로 하나씩 pad 노드가 붙여져서 앞 레이어에 pad 가 덧 붙여지고(2) Pooling 레이어 노드 값들을 계산한다(B).



(2)

(B)

0	0	0	0	0	0	0
0	0.1	0.9	0.2	0.5	0.3	0
0	0.8	0.5	1.2	1.1	0.7	0
0	1.3	1.6	0.1	2.1	1.3	0
0	2.3	0.9	1.2	3.2	3.2	0
0	0.2	1.4	0.3	1.3	2.2	0
0	0	0	0	0	0	0

N	=	0	
_	_		

0.9	1.2	1.1
2.3	3.2	3.2
2.3	3.2	3.2

[**참고] Pooling 레이어도 에지 값을 학습시키는가?** 답: No~! 따라서 에지를 칼라로 그렸다. 학습하는 weights 들은 흰색으로 그렸다. [앞 Dense, CONV, Pooling 레이어들 참조]

## 5. Padding 변환

Padding 변환(1)은 앞 레이어(2)에 위(tpad)아래(bpad)좌(lpad)우(rpad)에 0 의 값을 갖는 pad들을 덧붙이는 것이다(3). That's it! (Padding 레이어는 따로 그리지 않고 앞 레이어에 pad 만 붙인다)

InputData	Activation	Dense	Padding	tpad	bpad 2	lpad	rpad		Inpu
	AddSC	Padding							Pade
	BN	Poolng	tpad : nu bpad : n	mber of to umber of	op padding bottom pa	g nodes dding no	(1: default odes	t)	
	CONV	Reshape	rpad : nu rpad : nu all >= 0	umber of i umber of i	eπ paddin right paddi	g noaes ng node			
			Values or value.	utside the	range are	set to th	ne nearnes	st	(3
			Confirm	Padding	1212	(1)			



212

(2)

[주의] Padding 버튼을 2번 사용하는 것은 1번으로 바꿀 수 있다. 2번 연속 사용하면 경고 메시지가 뜬다.

## 6. Activation 변환

노드 값들을 일정 범위 내로 줄이거나 음수인 노드 값들은 0 으로 만들거 나 하는 등의 변환을 할 수 있다. 이러한 변환이 Activation 변환이다. 역시 weights는 학습하는 대상이 아니다



#### 옵션: N은 0~5 (디폴트 1)

dropout은 앞 dense 때와 동일하다. 즉 dropout = 0.1 이라면 앞 레이어의 노 드 값들 중 10% 를 랜덤하게 0으로 만 든다.

(1) Activation 변환된 레이어의 shape는 앞 레이어 shape 와 동일하다

(2) 앞 레이어 노드와 연결되는 선의 칼라를 하얀색으로 하 지 않았다. 학습되는 대상이 아니라는 의미이다.

#### 6-1. Activation 함수들 모양은?











N = 3

N = 4

### 6-2. softmax 함수는 무엇인가? (N = 5 경우)

softmax 함수는 주어진 값들을 확률로 바꾸어 주는 변환이다. 이 함수는 이미지 분류에서 확률을 나타내는 유용한 변환이다.

#### <u>softmax 함수</u>

[참고] Appendix I-3 수학식

[예] II.4 에서 고양이와 강아지에 대한 변환 값에 대하여 위 변환을 다시 해 보면 다음과 같다. 여기서 n = 2.

고양이 원래 값	softmax 변환 값	강아지 원래 값	softmax 변환 값
(247.2, 520.3)	(0,1)	(289.1, 181.4)	(1, 0)
(276.1, 635.4)	(0,1)	(386.5, 316.3)	(1, 0)
(324.4, 590.6)	(0,1)	(484.5, 375.3)	(1, 0)

원래 값보다 확률로 변환된 값들이 훨씬 통일되고 이해하기 쉽다~

### 6.3 y[j] 를 다 더하면 정말 1 이 되는가? 즉 확률이라 볼 수 있는가?

예: (2.3, 1.1, 3.5, 1.4, 0.5) 라는 값들을 softmax 변환하면 순서대로 (0.19256025, 0.05799803, 0.63932256, 0.07828916, 0.03183) 을 얻는데 이 값들 다 더하면 1 이다.



## 7. BN(Batch Normalization) 변환

weights 계산을 빠르고 안정적으로 잘되게 도움을 주는 변환이다.



BN 변환은 다음처럼 한다. (1) 앞 노드 x[0,0,0]를 적절히 다듬어서 변환한다 (2) y[0,0,0] = (x[0,0,0] 를 변환한 값)\*w[0] + b[0] y[1,0,0] = (x[1,0,0] 를 변환한 값)\*w[0] + b[0] ... (3) y[4,4,2] = (x[4,4,2] 를 변환한 값)\*w[2] + b[2] N=0 이면 이것 없음 [옵션] N은 0, 1 (디폴트 1)]

학습 시킬 웨이트는 채널 개수 ch에 대하여 - w[0], w[1], ..., w[ch-1] : ch 개 - b[0], b[1],..., b[ch-1] (N=1 때만)



(x[0,0,0] 를 변환한 값) ⇒ (그림에서 작은 구)

좀 더 구체적으로 변환에 대하여 알아보자.

I. AI Lab Newton 요약에서 학습할 데이터들을 모아놓은 것들이 Training Data인데 학습 데이터 전체를 한번에 넣어 학습하는 것보다 일정 묶음(Batch) 씩 나누어 학 습 시킨다고 하였다.



보통 batch 개수를 50 개 이상으로 한다. 그러나

(1) batch 개수 : 5개

(2) shape = (2, 2, 2)

여기서는 간단한 예를 보이기 위하여 다음과 같이 가정하자.

채널 0 : 배치 5개의 전체 노드 개수 : 2\*2\*5 = 20 이 노드들을 설명의 편의상 20개 노드들을 의 1차원으로 놓아보자





이제 이 batch 데이터 노드들에 대하여 BN 변환을 어떻게 하는지 살펴보자. 채널1도 동일하다.

 $\hat{\nabla}$ 

batch 20 개 노드 값(original node values): 64, 87, 82, 36, 94, 8, 31, 5, 49, 81, 52, 12, 15, 61, 4, 65, 45, 26, 99, 88

평균값 (mean) = (64 + 87 + 82 + ...+99 + 88)/20 = 50.2

v[1,1,0] = (x[1,1,0]를 변환한 값)\*w[0] + b[0]

v[0,0,1] = (x[0,0,1]를 변환한 값)\*w[1] + b[1]

y[0,1,1] = (x[0,1,1]를 변환한 값)\*w[1] + b[1] y[1,0,1] = (x[1,0,1]를 변환한 값)\*w[1] + b[1]

v[1,1,1] = (x[1,1,1]를 변환한 값)\*w[1] + b[1]

분산 (variance) = ( (64- mean)\*(64-mean) + (87- mean)\*(87-mean) + ... + (88- mean)\*(88-mean) )/20 = 970.66



원래 같은 배치 내의 노드 값들(Original node values)은 크기 차이가 매우 크다. 반면 변 환한 노드 값(Normalized node values)들은 그 차이가 매우 작다. 물론 그럼에도 100배로 확대하여 보면 오르락 내리락 경향은 같다(주황색 그래프)

따라서 변환한 normalized node values 는 마치"날뛰는 값"들을 "진정 시킨 값"처럼 볼 수 있다. 이를 통해 학습을 안정적으로 빠르게 할 수 있다.

## 8. Reshape 변환

Reshape 변환은 노드 자체 값이나 노드 전체 개수는 변하지 않고 shape 만 변경하는 변환이다.

Reshape 레이어는 앞 레이어의 노드들을 1차원 또는 2,3차원으로 바꾸 어 변환된 레이어이다





#### [예] Reshape 변환하는 방법

편의상 InputData 레이어와 Dense 레이어가 있다 하자.

#### (1) Reshape 할 레이어 선택 방법 : Ctrl + ML 로 3D 레이어를 클릭하여 선택

- (2) Reshape 버튼을 클릭한다. 그러면 매개변수 패널에 가능한 경우들이 출력된다. 즉 앞 레이어 shape 가 (10,1,1) 인 경우 가능한 변환은 (10,1,1), (5,2,1), (2,5,1) 3종류가 있다. 이 중 (5,2,1)을 마우스로 클릭하고
- (3) Confirm 버튼을 클릭한다.
- (4) 레이어 패널에서 "Add" 버튼을 클릭하면 초록색 레이어 뒤에 Reshape 5 2 1
  - 이 위치하며
- (5) 앞레이어 바로 뒤에 3D Reshape 레이어가 그려진다.

InputData	Activation	Dense	Reshape	InputData 30 30 3 Dense 10 1 0	
	AddSC	Padding	521 (2)	Reshape 521 (4)	(5) (
	BN	Poolng	251		
	CONV	Reshape			
			ny(number of nodes in y-direction) nz(number of nodes in z-direction) ch(number of channels) Confirm Reshape 5 2 1 (3)		

#### [참고] (3,3,2) 레이어를 Reshape 18 1 1 로 변환하면 각 픽셀 값의 위치는?

(3,3,2)

편의상 노드들을 작은 사각형으로 그렸다 따라서 3x3 사각 이미지로 그렸다.

0	1	2
3	4	5
6	7	8

9	10	11
12	13	14
15	16	17





### 8-1. Reshape 레이어 Add/Insert 하는 방법

#### [예] Reshape 할 레이어가 끝에 있을 때는 Add 한다.

- (1) 예를 들어 레이어가 4개 있는데 끝 레이어 Activation 1 0 를 Reshape 할려면
- (2) 3D 레이어들 중 Ctrl + ML 를 클릭하여 선택한다. 그러면 글씨가 초록색으로 변한다.
- (3) 이때 Reshape 버튼을 클릭하면 자동으로 가능한 변환이 보여진다. 여기서 841 10 1 를 택하고 Confirm 버튼을 클 릭한다.
- (4) 그리고 레이어 패널에서 "Add" 버튼을 클릭하면
- (5) Reshape 841 10 1이 레이어 패널에 추가되며, 동시에
- (6) 3D Reshape 레이어가 Activation 레이어 뒤에 그려진다.











### 8-2. Reshape 레이어 Add/Insert 하는 방법

#### [예] Reshape 할 레이어가 중간에 있을 때는 Insert 한다.

- (1) 예를 들어 레이어가 4개 있는데 중간 레이어 CONV 2 2 10 0 1 1 1 를 Reshape 할려면
- (2) 3D 레이어들 중 Ctrl + ML 를 클릭하여 선택한다. 그러면 글씨가 초록색으로 변한다.
- (3) 이때 Reshape 버튼을 클릭하면 자동으로 가능한 변환이 보여진다. 여기서 290 29 1를 택하고 Confirm 버튼을 클릭 한다.
- (4) 그리고 레이어 패널에서 "Insert" 버튼을 클릭하면
- (5) Reshape 290 29 1 이 레이어 패널에 선택된 레이어 뒤에 추가되며, 동시에
- (6) 3D Reshape 레이어가 CONV 레이어 뒤에 그려진다.











### 8-3. Reshape 레이어가 있는 상황에서 앞 레이어가 변경 될 때





Ctrl + ML 로 3D 레이어를 선택한 후 Reshape 10 64 1변환을 했다. Shape = (10,64,1)

•	InputData 10 10 1	
	CONV 2 2 10 0 1 1 1	이때 드니페 케이이크
	BN 1 0	이때 누면째 레이어들
	Activation 1 0	ML 로 선택한 후
	CONV 2 2 10 0 1 1 1	
	Reshape 10 64 1	
	Ļ	
•	InputData 10 10 1	(0, 0, 10)
	BN 1 0	(9, 9, 10)
	Activation 1 0	
	CONV 2 2 10 0 1 1 1	
	Reshape 10 64 1	
		- 🚩 🔽 🦵 🗸 🗤

Delete 하면 CONV 레이어의 shape = (9,9,10) 이 되 었다. 따라서 Reshape 10 64 1 은 앞 레이어와 총 노 드 개수가 다르게 되었다(일단 Delete 가 실행됨).





따라서 이런 경우에 앞 문제의 Reshape 10 64 1을 Delete 한 후 다시 Reshape 레이어를 원하는 것으로 추가하면 된다.



이후 레이어 패널의 레이어들의 변화가 있으면(여기 서 Activation 레이어 추가) 레이어 패널을 Check 하여 문제가 있는 앞의 Reshape 레이어 색이 빨간색으로 변하여 잘못된 레이어 임을 알려준다.

## 9. AddSC (Skip Connection) 변환

AddSC 변환은 고급 기능이다. 그러나 앞의 변환들에 익숙하다면 쉽다. 따라서 앞의 다른 변환에 익숙한 다음에 사용할 것을 권장한다.

SC

layers



그림 A 와 B의 차이는 B는 A의 main layers 에 layers 를 추가한 변환이다. main layers 에 추가된 레이어를 SC(Skip Connection) 레이어라 부른다. SC 레이어는 인공지능 학습 성능을 획기적으로 높인 변환이다.

#### 9-1. AddSC 변환에서 노드 값 계산하기

AddSC 레이어 노드는 다음과 같이 계산한다



- 어떤 레이어의 노드 값을 메인 레이어 외에 추가로 레이어를 만들어 여러 갈래로 계산되 어 오다가 AddSC 레이어에서 더하는 것이다.

- 메인 레이어에서 AddSC 레이어(1) 의 노드 값 x 는 메인 레이어 중 바로 앞 레이어(2) 의 노드 값 x1 과 추가된 레 이어(3) 의 노드 값 x2 를 더한 값이다.

즉 x = x1 + x2 로 계산한다~

이제 이러한 추가 레이어를 만드는 방법을 살펴보자.

### 9-2. 레이어 2개를 선택하여 SC 레이어들을 추가

레이어를 추가하여 연결하려면

- 연결할 2 레이어를 먼저 선택해야 한다
- 2 레이어를 선택하는 방법은 2레이어에 같은 번 호(SC 번호라 부른다)를 붙인다. 같은 번호 쌍을 연결한다.

번호가 붙지 않은 모든 레이어들을 C 라하고, AddSC 레이어가 아닌 번호가 붙은 레이어들을 H(머리)라 하자. AddSC 레이어는 T(꼬리)라 하자.

이때 0H, 0T 가 한 쌍으로 서로 짝이다. 번호는 0 을 포함하여 1~9 의 10개 중 아무 번호나 사용할 수 있다.

[머리-꼬리 규칙] T(꼬리) 가 있으면 반드시 그 앞에 같은 번호를 갖는 짝인 H(머리)가 있어야 한다

#### [좋은 예1]

1H... 2H... 0H ... 2T ... 0T... 1T ...

[잘못된 예]

0H ... 1T ... (1T 가 있으면 반드시 그 앞에 1T의 짝 1H가 있어야 한다)

nputData 224 224 3		С
Padding 3 3 3 3		С
@CONV 7 7 64 0 2 2 1		ОH
3N 1 0		С
.@Activation 10		1H
Padding 1111		С
@Pooling 3 3 0 0 2 2		2H
CONV 1 1 256 0 1 1 1		С
3N 1 0	7	С
CONV 1 1 64 0 1 1 1		С
.@AddSC	,	1T
3N 1 0		С
)@AddSC		0T
CONV 3 3 64 1 1 1 1		С
@AddSC		2T
3N 1 0		С
Activation 10		С
)@BN 1 0		ОН
		7

• H(머리)는 혼자 있을 수 있다. 아직 뒤에 T(꼬리)가 나오지 않았다는 뜻

(0)

(1)

(2)

### 9-3. AddSC 레이어 외 레이어에 SC 번호 붙이기 : Add



입력하여 Confirm

#### 9-3. AddSC 레이어 외 레이어에 SC 번호 붙이기 : Replace









CONV 2 2 10 0 1 1 1

BN 1 0



해당 레이어(CONV) 버튼을 클 릭한 후 SC 번호 입력

### 9-4. AddSC 레이어 추가하는 방법



앞에서 L4 레이어에 SC 번호 0 을 붙였다. 이제 L4에서 시작 하는 SC 레이어를 추가해 보 자.



#### ↑ 추가 sc 레이어가 그려진다





3D 레이어 L7을 Ctrl + ML 하여 선택하면 레이어 패 널의 글자가 초록색으로 변한다.

추천 추가 SC레이어 클릭 하고 Confirm

AddSC 버튼을 클릭하 면 추가 레이어들을 추 천되어 보여준다. (여기 서는 1개). 또한 앞 SC 번호가 자동 입력된다.





Ľ

### 9-5. 앞에 SC 번호가 2개 이상일 때



SC 번호가 붙은 레이어가 2개 있을 때 AddSC 레이어 를 추가하기 위하여 맨 마 지막 레이어를 Ctrl + ML 로 택하면

₽



선택된 레이어 글씨가 초 록으로 변하고

Select one skip connection number in previous layers-! OK Select one skip connection number in previous layers-!

여러 개(여기서는 2)의 SC 번호 중 하나를 택하라고 팝업 창이 뜬다.

예를 들어 0을 입력(없는 번 호 입력하면 다시 입력 받기 위해 대기한다)하고 OK 클릭



### 9-5. AddSC 다음 숫자 n 은 무슨 뜻인가?

가 레이어 패널에 쌓이고



두번째 레이어 노드와 AddSC 레이어 노드 사이에 위로 선이 그려진 다. 이는 두번째 레이어 노드 값(x2)을 변환없이 직전 레이어 노드 값 (x1) 과 더한다는 뜻이다. 즉 x = x1 + x2

### 9-6. AddSC 레이어가 있는 상황에서 앞 레이어가 변경 될 때





(30, 30, 1)

이후 레이어 패널의 레이어들의 변화가 있으면(여기 서 Activation 레이어 추가) 레이어 패널을 Check 하여 문제가 있는 앞의 AddSC 레이어 색이 빨간색으로 변 하여 잘못된 레이어 임을 알려준다.

#### 9-7. Reshape/AddSC 레이어가 있는 상황에서 앞 레이어가 변경 될 때 (레이어들 특징)



# IV. 레이어 패널



### 1. Add : 레이어 패널 끝에 넣는다

Add Delete	Replace	Insert	Delete All
------------	---------	--------	------------



[예] InputData 버튼을 클릭하고 Confirm 한 다음 Add하면 레이어 패널 끝에 들어간다. 레이어 패널 에 쌓이는 레이어들은 자동으로 3D 로 그려진다



[예] CONV 버튼을 클릭하고 Confirm 한 다음 Add하면 InputData 레이어 뒤의 레이어 패널 끝에 들어간다

#### 첫 레이어는 중심에 있어 1자로 보인다. Alt + ML 로 회전 시켜 3D로 볼 수 있다





#### 2. Delete : 삭제하려는 레이어를 ML 로 선택 후 Delete 하면 삭제된다





[예] 삭제하려는 레이어를 ML 로 선택한 후 Delete 하면 선택 된 레이어가 삭제된다.

### 3. Replace : 레이어 패널의 레이어를 ML 로 선택 후 Replace 하면 대체된다


## 4. Insert: 레이어를 ML 로 선택 후 Insert하면 선택 레이어 앞에 삽입된다



ML 도 신택안 우 Insert 아 앞에 삽입된다.

#### DeleteAll 버튼은 레이어 패널 의 모든 레이어들을 삭제한다.

Replace

Add

Activation 10

Delete

#### 레이어들이 모두 삭제되었다



Delete All

Insert





# 5. DeleteAll : 레이어 패널의 모든 레이어를 삭제한다

## 6. InitDraw : 회전/이동/확대된 3D 레이어들을 초기 상태로 그린다 (MM 참고하세요)





## 7. Closer : 레이어들의 간격을 축소한다 8. Wider : 축소된 레이어들을 확장한다











#### 실행한 것을 취소하여 이전으로 되돌리거나 다음으로 가게 한다.

# V. Open & Save 패널



# 1. Open 버튼



#### Open : 저장해 놓은 파일을 연 다. 이때 다음 3개가 수행된다.

C:\0	Newton	Model\	sample	laver8

	InputData 30 30 1
•	CONV 2 2 10 0 1 1 1
	BN 1 0
	Activation 1 0
	CONV 3 3 10 0 1 1 1
	BN 1 0
	Activation 1 0
	CONV 2 2 10 0 1 1 1

 파일 경로와 파일 이름이 좌측상단에 표시된다

 레이어 패널에 레이어들이 표시된다

> • 레이어 패널에 표시된 레이 어들이 3D 로 그려진다

# 2. SaveAs 버튼

SaveAs : 현재 레이어 패널의 레이어 모델을 새로운 이름으로 저장한다 Save : Open 된 파일의 레이어 모델을 (변경하는 경우 등일 때) 다시 저장한다.

# 3. Save 버튼

Save : Open 된 파일의 레이어 모델을 (변경하는 경우 등일 때) 다시 저장한다.

# VI. Training 패널



# [Training? 무엇을 학습하나?]

(1) 입력데이터들 전체를 모델(model) 에 따라 변환한 후

(2) 변환 결과 원하는 정답이 나오도록 에지들 결정 : loss(손실) ~ 모든 데이터들에 대해 변환 결과와 정답과 차이 평균값 학습 목표 : 가능한 loss ~ 0 이 되도록 weights(흰색 에지들)를 구하기

(3) 학습이 잘되었는지 구체적인 평가는 accuracy 로 한다.









#### [미리 Setting 할 것을 그림으로 이해하기]





# [Start Training 하기 전 setting 할 것(constants)]



[Start Training 하면 setting 한 것은 고정된다. Setting 을 다시 바꾸어 도 영향이 없다. Setting 을 바꾸면 다시 Start Training 해야 한다~!] (1) Class Num : Output Data 개, 즉 분류할 개수 예를 들어 고양이와 강아지를 구분한다면 Class Num = 2.

(2) Epoch Num : Training Dataset 전체 학습 회수. 주어진 데이터가 예를 들어 10000개라면 학습할 때 10000개를 몇 번이나 학습할 것인가를 나타낸다. 예 를 들어 이 값을 500으로 두면 전체 학습 데이터 10000개를 500 번 학습하겠다 는 뜻이다. 디폴트는 500.

(3) Batch Num : 학습할 때 데이터 전체를 계속 입력 시키며 학습시키 는 것 보다 한 묶음 씩 학습 시키는 게 좋다. 즉 한 묶음 다 학습 시킨 후 그 다음 묶음 학습시키고... 등등. 예를 들어,학습할 때 10000개를 한번에 메모리로 불러 학습하는 것 은 메모리 부족 문제도 있을 수 있고 weights 구하기가 일반적으로 어 렵다. 전체 데이터를 한번에 학습하지 않고 나누어 학습시키면 훨씬 효과 적이다. 예를 들어, 10000 개 중 100개씩 나누어 학습할 수 있다. 이럴 때100을 Batch Num 이라 부른다. 디폴트는 64.

(4) Learning Rate : 학습하는 속도를 나타낸다. 이 값이 크면 빨리 학습하는데 정 확도가 떨어질 수 있다. 이 값이 작으면 학습 속도가 느릴 수 있다. 학습 데이터 에 따라 달라질 수 있다. 디폴트는 0.002.





#### (5) Training Data Folder : 학습할 데이터들이 저장된

**폴더.** 다음 그림은 daisy 부터 tulips 까지 5종류의 꽃들을 분류하기 위하여 학습할 꽃들을 폴더 별로 구분하여 저장해 놓은 폴더 구조를 보여준다. Training Data Folder 는 이 경우 flower\_photos 폴더를 나타낸다. 잘못 선택하면



(5a) Training Data Folder path : 상기 Training Data Folder 를 클릭하여 폴더를 선택하면 여기에 쓰여진다.

(6) Trained Data Folder : 학습한 weights들을 저장할 폴더를 선택한다. 그러면 학습이 끝난 후 weights 값들 관련(데이터 파일과 INDEX 파일로 되어 있음) 파일들이 그 폴더 아래에 저장한다.

(6a) Trained Data Folder path: 경로가 잘 선택했는지 보여진다.

(7) Optimizer(최적화): 인공지능에서 변환하는 weights 를 구하는 방법을 수학적으로 정교하게 만들어 놓은 것이다. 현재 현업에서 많이 사용하는 Adam, SGD, RMSprop 의 3가지 옵션 을 선택할 수 있다. 이외에도 사용자가 필요한 것을 요구하면 추가할 수 있다. (7a) 어떤 것을 선택할 것인가? 초보 사용자는 어느 것을 택해도 비슷하게

weights 를 구할 수 있다. 같은 모델이라도 이것 저것 택해 사용해 보기를 적극 권장한다.

(8, 8a) Prediction Data Folder : Trained Data Folder 에 저장된 학습한 weights들을 이용하여 Prediction Data Folder 에 있는 이미지들을 불러와 예측한다. 이것은 Start Prediction 버튼을 클릭하기 전 설정해야 한다.

# 1. Start Training 버튼

[다시 강조] "Start Training" 버튼 클릭 후에 모델 이나 상수 등이 변경되어 파일이 변경되면 "Start Training"을 다시 클릭해야한다.



#### Start Training 버튼을 클릭하면 우측의 그림과 같은 박스가 뜨며, 그 아래 4개의 버튼을 볼 수 있다.

#### **Open Colab**

사용자 계정의 Colab 과 Google Drive 웹사이트가 열린다.

#### **Copy Mount**

"Copy Mount" 버튼을 클릭하면 내용이 클립 보드에 저장된다. Colab 에서 "+코드 " 를 클릭 한 후 "Ctrl + V" 하면 내용이 코드 란에 붙여 넣 어 진다.

#### Default

Dropdown 버튼을 클릭하면 Default 아래 Mnist, Fashion\_mnist 등 4개의 데이터 셋이 보 인다. 이중 선택할 수 있다.

#### Copy Main

학습시킬 코드를 클립보드에 저장되게 하여 colab 에 "Ctrl + V" 하면 된다.



## 1-1. Open Colab

Google Gmail OIN & III OM

Open Colab 버튼을 클릭하면 사용자 계정의 구글의 Colaboratory(Colab)과 Drive 웹사이트가 동시에 열린다.

Colab 의 노트에서 CPU/GPU 계산한다

cc	▶ Colaboratory에 오신 것	일 환영합니다		
	파일 수정 보기 삽입 런 목차	노트 열기		
Q	시작하기	예	>	Q
{ <i>x</i> }	데이터 과학 머신러닝	최근 사용	>	
•	추가 리소스	Google Drive	>	4
	추천 예시 ■ 섹션	GitHub	>	co
		업로드	>	4

#### Colab에서 노트 열기하고 그곳에 코드를 넣고 계 산한다

#### Drive 는 Colab 과 연동되는 메모리이다.

본인 Google 계정

이어야 한다~!



Drive 에 학습할 Training Dataset을 저장하고 Colab 에서 연동 시키면 Colab노트에서 계산할 때 이곳의 데이터 셋을 이용하여 계산한다. Colab에서 실행하기 전에 런타임 > 런타임 유형 변경 > 하드웨어 가속기 에서 GPU 를 선택한다. 무료이지만 세계 많은 사람이 사용하기 때문에 간혹 GPU 자원이 부족할 수 있다. 그러면 CPU 로만 계산하여야 한다.

C	0	Colabora	tory <sup>0</sup>	네 오신	! 것을 힘	환영협	입다			
		파일 수정	보기	삽입	런타임	도구	도움말	<u>변경사</u>	항을 저장	할 =
=	목기	4			모두	실행			Ctrl+	-F9
_					이전	셀 실형	ł		Ctrl+	F8
ર	٨	직하기			초점	이 맞춰	진 셀 실형	뱅	Ctrl+En	ter
rl	G	이터 과학			선택	항목 실	행	Ct	rl+Shift+En	ter
[-*]	D	1신러당			이후	셀 실험	ł		Ctrl+F	10
	축	독가 리소스			실행	중단			Ctrl+I	MI
추천 예시			런타	임 다시	시작		Ctrl+	М.		
		구선 에지		다시	시작 및	L 모두 실험	행			
	H	섹션			런타	임 연결	해제 및	삭제		
					런타	임 유형	변경			
					세션	관리				
					리소	스 보기				

런타임 유형 변경						
런타임 유형						
Python 3	•					
하드웨어 가속기 ?						
О СРИ 💽 ТА	4 GPU	$\bigcirc$	A100 GPU	$\bigcirc$	V100 (	GPU
O TPU						
프리미엄 GPU를 이용하시겠	넷어요?	추가 컴	퓨팅 단위 구매			
				i	취소	저장

#### 유료로 사용할 수도 있다. 자세한 내용에 대해서는 Colab 웹사이트를 읽어보기를 권장한다.

# <Google Drive 와 USB 이용>

1. 개인 컴퓨터

- 데스크탑 Drive 설치(1)하고 동기화 시켜 사용하거나 C:\ 또는 D:\ 아래 또는 usb 이용 (2) 할 수 있다.

#### 2. 공용 컴퓨터 (예: 학교 컴퓨터)

학교 컴퓨터 라도 각자의 구글 드라이브를 다운받아 설치하고 사용할 수 있다. 컴퓨터 1개당 Drive 4개 까지 설치 가능하다. 그러나 이 방법은 보안을 유지하기 어렵고 자칫 오류가 발생할 수 있다. 따라서 AI Lab Newton 은 컴퓨터 1개에 한번만 설치해도 되지만 구글 Drive 데이터들은 각자가 관리하는 것이 편리하다.

주의: (디렉토리 이름):\한글로 "내 드라이브"

\* 학교에서 사용할 때 추천 방법 (보안 측면에서도 강력히 권장)\* - 자기 USB (2) 에 E:\내 드라이브\ (3) 아래 데이터들을 넣은 디렉토 리 구조(4)를 만들어 이 폴더를 합쳐 zip 파일(5)로 만든 후 구글 Drive 에서 업로드하여 unzip 하고 사용한다.

- 학습 폴더(training) (6)는 반드시 하위 CLASS\_NUM 만큼의 데이터 폴더(7) 가 포함되어야 한다.

- 학교 컴퓨터에서 사용할 때도 학교 컴퓨터에 USB 꽂아 사용하면 편리하다 (미리 학교에서 실습하기 전에 구글 Drive 에 데이터 업로 드하면 시간 절약될 수 있다)



이렇게 USB 에 폴더 만들어 저장하고 사용하는 이유? (1) 구글 Drive 에 업로드할 데이터셋 저장 (2) 구글 Drive 데이터 경로를 AI Lab Newton 에서 미리 정하여 CLASS\_NUM 과 하위 폴더 개수가 맞는지 검토하기 위하여

#### 힌트: E 아래 반드시 "내 드라이브" 그 다음 아래는 자기 마음대로 편하게. 구글 Drive 에도 이 구조를 똑같이.

#### [문제] 옳은 방법과 잘못된 방법 찾기



[답] x o x o x

#### **1-2. Copy Mount** 주어진 코드는 Colab 에 Google Drive 를 Mount 시키는 역할



#### 이 박스 안에 python으로 만들어 진 간단한 코드가 보여진다. 동시 에 클립보드에도 저장된다.

이미 클립보드에 저장되어 있기 때문에 Colab 노트에 "Ctrl + V" 로 붙여 넣기 하면 다음 처럼 보인다.

#### Colab의 노트는 주어진 계산 명령을 실행한다.

Colab 텍스트 등에 글을 쓰거나 그림을 넣을 수도 있다. 이러한 기 능들은 colab 을 읽어보기를 권장하나 본 AI Lab Newton 에서는 오직 계산하는 기능만 사용한다.



Colab의 노트에 데이터 셋을 이용한 계산 명령을 실 행하려면 사용자 계정의 Drive 에 연결해야 한다. 이때 Colab 과 Drive 를 연결하기 위해서는 다음 엑 세스 요청을 허락해야 한다.

(\* 코드는 계산 명령들 집합이라고 볼 수 있다)

노트북에서 Google Drive 파일에 액세스하도록 허용하시 겠습니까?

이 노트북에서 Google Drive 파일에 대한 액세스를 요청합니다. Google Drive에 대한 액세스 권한을 부여하면 노트북에서 실행되는 코드가 Google Drive의 파일을 수정 할 수 있게 됩니다. 이 액세스를 허용하기 전에 노트북 코드를 검토하시기 바랍니다.

아니요 Google Drive에 연결



### 1-3. Dropdown : Default & Non-defaults



Dropdown 버튼에 다음과 같은 옵션들이 있다 : default & non-defaults

Default : 사용자가 Drive 에 데이터 셋을 로드하고 그 데이터 셋을 학습시 키는 것은 디폴트이다.

그러나 Colab 무료 버전은 세계 많은 사람들이 사용하기 때문에 GPU를 못 쓸 수도 있고 속도가 느릴 수도 있다(물론 유료로 사 용하면 좀 더 빠르게 사용할 수도 있다)

이러한 문제점을 극복하기 위하여 <u>CPU 만 사용하더라도 특정</u> <u>주어진 데이터 셋들(non-defaults)에 대해서는 빠르게 계산할 수</u> <u>있다</u>.

Non-defaults : Drive 에 있는 사용자가 로드한 데이터 셋 대신 구 글에서 제공하는 데이터 셋을 이용하여 학습한다.

(추후 상기 4개의 데이터 셋 외에도 사용자들의 요구에 따라 확장할 예정이다)

## 1-4. Copy Main : default

Dataset Load Method default Ready to paste into Colab notebook~!
Open Colab Copy Mount Default 🗸 Copy Main

일반적으로 내용이 길어 자세한 코드는 colab에 붙여놓기한 후 볼 수 있다.

- 사용하는 데이터 셋(이 경우 default)과 - "Ready to paste Colab notebook~!" (colab 노트를 붙여 넣을 준비가 됨~!) 만 출력한다.

실제로 학습시킬 tensorflow 코드 가 클립보드에 저장된다. 이를 colab 에 "Ctrl + V" 로 붙여 넣기 하고 실행시키면 된다.

#### <학습하며 출력하는 내용 이해하기 : 데이터를 나누는 이유와 역할>

	Training data		Evaluation data	Test data
Batch	Batch 로 나누어 Batch 단위로 학습. 1 epoch 는 training data 전체 학습 한번 하는 것 Evaluation data 의 loss와 accurate 습 데이터에 대해 과적합(overfi 생하고 있는지, 모델이 검증 데 서도 일반화(generalizing)를 잘 를 판단할 수 있다. 이러한 평가 학습 진행 상태를 모니터링하고 변수들을 튜닝하는 데 중요한 2 한다.	1 epoch 학습 후 evaluation data 에 디 여 loss 와 accuracy 전체 학습 오/를 보고 학 오/를 보고 학 와 accuract 와 accuract 와 accuract 와 accuract 와 accuract	배하 예산 이 종료된 후 y 를 계산한	드에 Ioss 다

#### <batch 학습 중 출력문 설명>





#### < 전체 학습 종료 후 출력문 설명 >

image\_count: 60000 Ir : Learning rate ['0' '1' '2' '3' class names: '8' '9'1 Epoch 1/500 Epoch 2/500 750/750 [------] - 5s 7ms/step - loss: 0.0721 - accuracy: 0.9773 - val\_loss: 0.0555 - val\_accuracy: 0.9820 - lr: 0.0020 Epoch 3/500 750/750 [========================] - 4s 6ms/step - loss: 0.0512 - accuracy: 0.9838 - val\_loss: 0.0374 - val\_accuracy: 0.9872 - lr: 0.0020 Epoch 4/500 750/750 [==================] - 5s 6ms/step - loss: 0.0385 - accuracy: 0.9874 - val\_loss: 0.0884 - val\_accuracy: 0.9738 - lr: 0.0020 Epoch 5/500 Epoch 6/500 Epoch 7/500 Epoch 8/500 Epoch 9/500 93/93 [------] - 342s 3s/step - Joss: 0.0464 - accuracy: 0.9872 🔨 Trained Data Folder: /content/drive/MyDrive/trained Test data 에 대한 loss 와 accuracy 계산 Test Loss: 0.046380020678043365 Test accuracy: 0.9872311949729919 학습한 결과 weights 데이터 저 장한 위치 Test data 에 대한 loss 와 accuracy 계산 결과



#### [다른 예]

image\_count: 60000 class\_names: ['0'] '1' '2' '3' '4' '5' '6' '8' '9'] Epoch 1/500 750/750 [=========================] = 7276s 9s/step = loss: 0.1530 = accuracy: 0.9536 = val\_loss: 0.0741 = val\_accuracy: 0.9768 = lr: 0.0020 Epoch 2/500 750/750 [========================] - 5s 6ms/step - loss: 0.0707 - accuracy: 0.9780 - val\_loss: 0.0737 - val\_accuracy: 0.9763 - lr: 0.0020 Epoch 3/500 750/750 [=================] - 5s 7ms/step - loss: 0.0521 - accuracy: 0.9832 - val\_loss: 0.0448 - val\_accuracy: 0.9869 - lr: 0.0020 Epoch 4/500 750/750 [======] - 4s 5ms/step - loss: 0.0399 - accuracy: 0.9868 - val\_loss: 0.0653 - val\_accuracy: 0.9824 - lr: 0.0020 Epoch 5/500 93/93 [------] - 239s 2s/step - loss: 0.0388 - accuracy: 0.9879 Trained Data Folder: /content/drive/MyDrive/trained Test Loss: 0.038781456649303436 Test accuracy: 0.9879032373428345



#### [또 다른 예]

image_count: 3670
class_names: ['daisy' 'dandelion' 'roses' 'sunflowers' 'tulips']
Epoch 1/500
45/45 [===========================] - 71s 1s/step - loss: 33.2833 - accuracy: 0.3736 - val_loss: 6.5903 - val_accuracy: 0.3125
Epoch 2/500
45/45 [=========================] - 75s 2s/step - loss: 1.9538 - accuracy: 0.7757 - val_loss: 3.5996 - val_accuracy: 0.3875
Epoch 3/500
45/45 [===========================] - 76s 2s/step - loss: 0.3293 - accuracy: 0.9378 - val_loss: 1.5756 - val_accuracy: 0.6219
Epoch 4/500
45/45 [==========================] - 76s 2s/step - loss: 0.0653 - accuracy: 0.9868 - val_loss: 1.6114 - val_accuracy: 0.6156
Epoch 5/500
45/45 [==========================] - 76s 2s/step - loss: 0.0197 - accuracy: 0.9951 - val_loss: 1.4823 - val_accuracy: 0.6656
Epoch 6/500
45/45 [=========================] - 76s 2s/step - loss: 0.0398 - accuracy: 0.9969 - val_loss: 1.4199 - val_accuracy: 0.7250
Epoch 7/500
45/45 [==========================] - 76s 2s/step - loss: 0.0462 - accuracy: 0.9986 - val_loss: 1.4150 - val_accuracy: 0.7594
Epoch 8/500
45/45 [==========================] - 76s 2s/step - loss: 0.0391 - accuracy: 0.9969 - val_loss: 1.3417 - val_accuracy: 0.7781
Epoch 9/500
45/45 [==========================] - 76s 2s/step - loss: 0.0242 - accuracy: 0.9972 - val_loss: 0.9323 - val_accuracy: 0.8531
Epoch 10/500
45/45 [==========================] - 77s 2s/step - loss: 0.0080 - accuracy: 0.9997 - val_loss: 1.2407 - val_accuracy: 0.8500
Epoch 11/500
45/45 [==========================] - 76s 2s/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 1.3000 - val_accuracy: 0.8594
Epoch 12/500
45/45 [==========================] - 76s 2s/step - loss: 0.0558 - accuracy: 0.9993 - val_loss: 1.1996 - val_accuracy: 0.8750
Epoch 13/500
45/45 [====================================
Epoch 14/500
45/45 [====================================
Epoch 15/500
45/45 [====================================
Epoch 16/500
45/45 [====================================
5/5 [=======================] - 3s 576ms/step - loss: 0.7757 - accuracy: 0.8969
Trained Data Folder: /content/drive/MyDrive/trained
Test loss: 0.7756503224372864

Test accuracy: 0.8968750238418579



Train 보다 Valuation 이 좀 더 정확. 즉, Train Acc 는 학습한 것이고 Val Acc 는 학습하지 않은 일반적인 것 이다. 따라서 Valid Acc

## 1-5. Copy Main : non-defaults



일반적으로 내용이 길어 자세한 코드는 colab에 붙여놓기한 후 볼 수 있다.

- 사용하는 데이터 셋(이 경우 mnist)과 - "Ready to paste Colab notebook~!" (코랩 노트를 붙여 넣을 준비가 됨~!) 만 출력한다.

실제로 학습시킬 tensorflow 코드 가 클립보드에 저장된다. 이를 colab 에 "Ctrl + V" 로 붙여 넣기 하고 실행시키면 된다. try

(x\_train, y\_train), \_ = keras.datasets.mnist.load\_data() x\_train = x\_train / 255.0 x\_train = x\_train[..., tf.newaxis] # 채널 차원 추가

#1-hot encoding
y\_train = to\_categorical(y\_train, num\_classes=CLASS\_NUM)

#### # 데이터 분할

x\_train, x\_val\_test, y\_train, y\_val\_test = train\_test\_split(x\_train, y\_train, test\_size=0.2, random\_state=42)
x\_val, x\_test, y\_val, y\_test = train\_test\_split(x\_val\_test, y\_val\_test, test\_size=0.5, random\_state=42)

train\_num = len(x\_train)
val\_num = len(x\_val)
test\_num = len(x\_test)

# tf.data.Dataset 객체 생성 train\_ds = tf.data.Dataset.from\_tensor\_slices((x\_train, y\_train)).shuffle(10000).batch(BATCH\_NUM) val\_ds = tf.data.Dataset.from\_tensor\_slices((x\_val, y\_val)).batch(BATCH\_NUM) test\_ds = tf.data.Dataset.from\_tensor\_slices((x\_test, y\_test)).batch(BATCH\_NUM)

except Exception as e: print("Error: (Load and Process Dataset (Nondefaults))", e) #<Dataset Processing Functions>###와 #<Load and Process Dataset>### 대신

여기의 #<Load and Process Dataset (Nondefaults)>### 로 대체했다.

이것은 데이터 셋으로 tensorflow 자체 보유 데이터 셋을 로드하기 때 문에 시간을 절약할 수 있 다.

Downloading data from <u>https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz</u>	lr , loorning roto
11490434/11490434 [===================================	In Tearning rate
Epoch 1/500	
750/750 [====================================	0.0020 🚩
Epoch 2/500	
750/750 [====================================	0.0020
Epoch 3/500	
750/750 [====================================	0.0020
Epoch 4/500	
750/750 [============] - 63s 85ms/step - loss: 0.0427 - accuracy: 0.9861 - val_loss: 0.0860 - val_accuracy: 0.9763 - Ir:	0.0020
Epoch 5/500	
750/750 [===========] - 68s 91ms/step - loss: 0.0344 - accuracy: 0.9885 - val_loss: 0.0811 - val_accuracy: 0.9788 - Ir:	0.0020
Epoch 6/500	
750/750 [============] - 61s 82ms/step - loss: 0.0285 - accuracy: 0.9900 - val_loss: 0.0854 - val_accuracy: 0.9802 - Ir:	0.0020
Epoch 7/500	
750/750 [====================================	0.0020
Epoch 8/500	
750/750 [====================================	0.0020
93/93 [====================================	
Trained Data Folder: /content/drive/MyDrive/trained	

Test loss: 0.10373567044734955

Test accuracy: 0.9749664068222046





Epoch 1/500 750/750 [-------] - 6s 6ms/step - loss: 0.1683 - accuracy: 0.9488 - val\_loss: 0.1028 - val\_accuracy: 0.9664 - lr: 0.0020 Epoch 2/500 750/750 [-------] - 4s 6ms/step - loss: 0.0708 - accuracy: 0.9788 - val\_loss: 0.0715 - val\_accuracy: 0.9778 - lr: 0.0020 Epoch 3/500 750/750 [------] - 6s 8ms/step - loss: 0.0539 - accuracy: 0.9828 - val\_loss: 0.0652 - val\_accuracy: 0.9793 - lr: 0.0020 Epoch 4/500 750/750 [------] - 4s 6ms/step - loss: 0.0423 - accuracy: 0.9868 - val\_loss: 0.0757 - val\_accuracy: 0.9800 - lr: 0.0020 Epoch 5/500 750/750 [------] - 6s 7ms/step - loss: 0.0354 - accuracy: 0.9879 - val\_loss: 0.0740 - val\_accuracy: 0.9772 - lr: 0.0020 Epoch 6/500 750/750 [------] - 4s 6ms/step - loss: 0.0297 - accuracy: 0.9879 - val\_loss: 0.0845 - val\_accuracy: 0.9788 - lr: 0.0020 Epoch 6/500 750/750 [------] - 0s 3ms/step - loss: 0.0297 - accuracy: 0.9901 - val\_loss: 0.0845 - val\_accuracy: 0.9788 - lr: 0.0020 Epoch 6/500 750/750 [------] - 0s 3ms/step - loss: 0.0898 - accuracy: 0.9901 - val\_loss: 0.0845 - val\_accuracy: 0.9788 - lr: 0.0020 Epoch 6/500 750/750 [------] - 0s 3ms/step - loss: 0.0898 - accuracy: 0.9901 - val\_loss: 0.0845 - val\_accuracy: 0.9788 - lr: 0.0020 Epoch 6/500 750/750 [------] - 0s 3ms/step - loss: 0.0898 - accuracy: 0.9901 - val\_loss: 0.0845 - val\_accuracy: 0.9788 - lr: 0.0020 Epoch 6/500

Test accuracy: 0.9763104915618896



# 2. Start Prediction 버튼




#### #<Load and Preprocess and ImagePath>###

예측할 Prediction Data 가 들어있는 디렉토리가 있 ♥ 고

< Drive 사이트 > content/drive/MyDrive/

토리

\_\_\_\_\_\_ (2) 답을 아는 CLASS\_NUM 만큼의 하 위 디렉토리가 있고 그 안에 예측할 해당 이미지 파일들 있는 디렉토리

(1) 예측할 이미지 파일들만 있는 디렉



한번에 계산하여 accuracy 를 계산 해 볼 수도 있다









#### ••• 하위 디렉토리가 없고 파일만 있습니다. 이미지: /content/drive/MyDrive/predistion/mnist/10.png 예측할



하위 디렉토리가 있습니다 이미지: /content/drive/MyDrive/prediction/mnist/10.png



[Prediction 옵션]

물론, 제대로 대응되는 하위 디렉토리에 이미지 데이터가

없으면 예측할 수 없다

'y' : 이미지 데이터 1개씩 호출 'n': 종료

'a' (all) : CLASS\_NUM 개수의 하위 디렉토리가 있고 이 하위 디렉토리 아래 해당 이미지 파일들이 있으면 전체 정확도(accuracy)를 계산할 수 있다.

# 3. Hide/Show 버튼



#### Start Training 버튼을 클릭하면 좌측과 같은 출력창이 뜬다. 이 창을 감추거나 다시 보이게 할 때 Hide/Show 버튼을 이용한다.



가끔 우측과 같이 메시지가 상기 출력창 속에 감추어질 수 있다. 이럴 때 Hide/Show 버튼을 이용할 수 있다.



# **APPENDICES**

I. AI Lab Newton 설치 및 준비사항
II. AI 수학
III. 단축키 및 레이어 shape 보기
IV. Colab 코드 요약 설명
V. Colab 오류 설명
VI. AI Lab Newton 을 이용한 개발 방법
VII. Datasets & Samples
VIII. History

# APPENDIX I : AI Lab Newton 설치 및 준비사항

- Al Lab Newton 은 인공지능을 학습하기 위하여 Google Colab 및 Drive를 사용한다. 따라 서 구글 gmail 계정을 갖고 있어야 한다.
- 학생 개인 컴퓨터에 AI Lab Newton 을 설치하여 사용할 수 있다. 반면
- 학교에서 1대의 컴퓨터를 여러 명이 사용하는 경우 각자의 계정으로 로그인해서 사용 해야 한다.

# 1. Google gmail 계정 갖고 로그인 해보기

- 1대의 컴퓨터를 여러 명이 사용하는 경우 각자 계정으로 로그인하여 gmail 을 사용한다.

[참고: 학교 컴퓨터에서 게스트 모드 열기]

1. 컴퓨터에서 Chrome을 엽니다.

2. 오른쪽 상단에서 프로필 🕒을 클릭합니다.

3. 게스트를 클릭합니다.

# 2. Google Colab 에 접속해 보기

### 2-1 : Google에서 Colab을 입력

G Google https://colab.research.google.com

Google Colab

2-2 : Colab 사이트 입장

CO Colaboratory에 오신 것을 환영합니다

파일 수정 보기 삽입 런타임 도구 도움말



2-3 : 사용자 GPU 계산기

▼ 시작하기





다른 값을 계산해 본다. GPU 계산기 이다! 이것으로 인공지능을 학습 시킬 예정이다~

# 3. Google Drive 에 접속해 보기

### 3-1 : Google에서 Drive 를 입력

G Google Accounts https://accounts.google.com > AccountChooser

Google Drive

### 3-2 : 계정 선택하고 입장





정보, 개인 정보 보호 및 보안 설정을 관리하여 나에게 맞는 방식으로 Google을 사용할 수 있습니다. 자세히 알아보기 ①



#### 입장 후 우측 상단 Google apps 클릭하고 그 아래 Drive 를 클릭한다 ? \*\*\* Google apps G 성구 Account Search Maps GE YouTube Play News M Gmail Meet Chat 31 Calendar Contacts Drive G ...

#### Drive 로 입장. 이곳은 구글에서 무료로 제 공하는 내 데이터 저장소(MyDrive)이다. 이 곳에 인공지능 학습 데이터들을 올리고 학 습 시킨다.

	Drive	Q Search in Drive
+	New	My Drive -
• 🖸	My Drive	Type • People • Modified •
•	Computers	*
å	Shared with me	Name 🗸
0	Recent	
☆	Starred	
()	Spam	
Ū	Trash	
$\bigcirc$	Storage	
• 310.	5 MB of 15 GB used	
	Get more storage	

참고 : MyDrive 가 "내 드라이브" 처럼 한글로 되어 있으면 영어로 바꾸기를 권장한다. 한글이면 인식을 못 할 수도 있다

## 4. Google Drive 는 내 컴퓨터에도 동시 존재하게 할 수도 있다

3-1 : Google Drive 동시 존재



#### 데이터를 인터넷상의 내 Drive 에 직 접 올려도 되지만 내 컴퓨터의 Google Drive (G:) 에 저장해도 자동으 로 인터넷상의 Drive 로 저장된다

#### 따라서, <u>1대의 컴퓨터를 여러 명이 사용하는 경우</u> <u>각자 계정으로 로그인하여 google drive 를 사용한다</u>.

### [참고: ChatGPT]

컴퓨터 1개에 2명의 사용자가 사용할 때 google drive 는 어떻게 각자 사용할 수 있는가?

한 컴퓨터에서 여러 사용자가 Google Drive를 사용하기 위해서는 각 사용자가 개별적인
 Google 계정으로 로그인하여 Google Drive에 액세스해야 합니다. 각 사용자는 자신의 Google
 계정을 사용하여 로그인한 후 개인적인 파일을 저장, 공유 및 동기화할 수 있습니다.

다음은 컴퓨터에서 여러 사용자가 Google Drive를 사용하는 방법입니다:

- 사용자 계정 전환: 컴퓨터에서 여러 사용자가 로그인할 수 있는 운영체제에서는 각 사용자가 자신의 계정으로 로그인하고 사용자 간에 계정을 전환할 수 있습니다. 이렇게 함으로써 각 사 용자는 자신의 Google 계정으로 Google Drive에 로그인할 수 있습니다.
- 웹 브라우저: 각 사용자는 웹 브라우저를 사용하여 Google Drive 웹 인터페이스에 접속하고 자 신의 계정으로 로그인할 수 있습니다. 이렇게 하면 각 사용자는 자신의 파일을 업로드, 다운로 드, 공유 및 관리할 수 있습니다.
- Google Backup and Sync 또는 Google Drive 앱: 각 사용자는 Google Backup and Sync 또는 Google Drive 앱을 설치하여 로컬 컴퓨터와 Google Drive를 동기화할 수 있습니다. 이렇게 함 으로써 각 사용자는 자신의 로컬 파일을 Google Drive에 업로드하고 Google Drive에서 파일 을 다운로드하여 사용할 수 있습니다.

위의 방법을 통해 여러 사용자는 동일한 컴퓨터에서 각자의 Google 계정으로 Google Drive를 사용할 수 있습니다. 각 사용자는 자신의 계정으로 로그인하여 개인적인 파일을 관리하고 공유 할 수 있습니다.













파일과 폴더를 Drive에

안전하게 저장하세요.



컴퓨터에서 애플리케이션을

사용해 Drive 파일을 여세요.



최신 상태로 유지하세요.





모든 Drive 파일을 자동으로 사진을 Google 사진에 백업 ..... 나 PC 🧊 3D 개체 🖊 다운로드 🛃 동영상 🗄 문서 🔜 바탕 화면 💽 사진 🎝 음악 🐛 로컬 디스크 (C:) 🧼 oldrock9 (D:) 🛆 Google Drive (G:)

시작하기

 $\times$ 

:

# 5. 학교 컴퓨터에 내 계정으로 로그인

5-1. 학교에서 컴퓨터 1대에 여러 명이 사용하는 경우 각자 개인의 계정으로 로그인한다.

← 설정	- 0
ŵ 홈	가족 및 다른 사용자
설정 검색 오	가족
계정	가족을 추가하여 모든 구성원에게 로그인 및 바탕 화면을 제공합니다. 적절한 웹 사이트, 시간 제한, 앱 및 게임을 사용하도록 설정하여 자녀를 안전하게 보호할 수 있습니다.
RE 사용자 정보	
🖸 이메일 및 계정	+ 가족 구성원 추가
Q, 로그인 옵션	자세한 정보
🖻 회사 또는 학교 액세스	기타 사용자
♀, 가족 및 다른 사용자	가족 구성원이 아닌 사람이 자신의 계정으로 로그인할 수 있도록 허용 합니다. 이 경우 이 사용자는 가족에 추가되지 않습니다.
	+ 이 PC에 다른 사용자 추가

#### < 구글 드라이브>

- 1. 학교 PC 에 각자의 계정을 만들어 각자의 계정으로 로그 인하여 그곳에 데스크탑 Drive 를 만들어 놓고 동기화 시 킨다. 이후 선택 버튼을 이용하여
- 2. 학교 PC 에 각자 계정을 만들기 어려우면 구글 드라이브 사이트에 저장된 경로를 직접 입력한다.

< 추가 코드 넣기 위해서>



학생들은 여기 보이는 코드는 그냥 컴퓨터 껍데기를 열었을 때 그 안에 보이는 것들과 마찬 가지 라고 생각하라. 컴퓨터 안의 모든 칩과 회로들에 대해 굳이 알지 못해도 컴퓨터를 얼마든지 잘 사용할 수 있다. 마찬가지로 여기 나오는 코드들(python/tensorflow 언어)은 굳이 몰라도 얼마든지 인공지능의 원리 를 이해할 수 있고 사용할 수 있다. 만약에 나중에 코드를 하고 싶은 학생들은 이것들을 눈 여겨 보는 것도 python/tensorflow 언어를 배우는데 도움이 될 수는 있다.

## 6. AI Lab Newton 설치

6-1. 학교에서 컴퓨터 1대에 AI Lab Newton 을 설치하여 여러 명이 사용할 수 있다. 이때 크롬을 구글 사용자 아이디로 열어 사용하도록 한다

6-2. 학교에서 뿐만 아니라 집에서 개인 컴퓨터에도 AI Lab Newton 을 설치하여 사용할 수 있다.

## APPENDIX II : AI 수학

### 1. Dense 변환의 수학적 표현(노드 값 구하기)

### 2. CONV 변환의 수학적 표현(노드 값 구하기)

For 
$$j = 0, 1, \dots, ny - 1, k = 0, 1, \dots, nz - 1$$
 and  $c = 0, 1, \dots, ch - 1,$   
 $y[j,k,c] = \sum_{c'=0}^{pch-1} \sum_{d_j=0}^{h-1} \sum_{d_k=0}^{w-1} x[s_y \cdot j + d_j, s_z \cdot k + d_k, c']w[d_j, d_k, c', c] + w[j, k, c]$  bias 항

#### 2.1 padding = 0, 1 일 때 CONV 레이어의 크기 ny, nz 구하는 방법

가정 : 앞 레이어(previous layer) 의 shape = (pny, pnz, pch) CONV 레이어의 shape = (ny, nz, ch)에서 (ny, nz) 구하기

< padding = 0 때>

ny = ceil((pny - h + 1)/sy)nz = ceil((pnz - w + 1)/sy)

< padding = 1 때>

ny = ceil(pny/sy) nz = ceil(pnz/sy)

ceil()은()안의 값보다 크거나 같은 가장 작은 정수를 나타낸다. 예: ceil(0.1) = 1, ceil(0.5) = 1, ceil(1.0) = 1, ceil(1.1) = 2 가정 : (pny, pnz, pch) = (5,5,3) 일 때 CONV 레이어 CONV 3 3 10 0 1 1 0 의 shape = (ny, nz, ch) = (ny, nz, 10)

**예**: padding= 0 인 경우, 앞 레이어 pny = 5, h = 3, sy = 1 이니까 CONV 레이어의 ceil((pny-3+1)/sy) = ceil(3) =3이다. 따라서 CONV 레이어의 (ny, nz) = (3,3) 이 된다

**예**: padding = 1 이고 sy = sz = 1 인 경우는 CONV 레이어의 크기 ny, nz 는 앞 레이어의 크기 pny, pnz 와 동일하다. 즉 ny = ceil(pny/sy) = ceil(5/1) = 5

반면 sy = sz = 2 인 경우는 CONV 레이어 크기는 (3,3) 이 된다. 즉, ny = ceil(5/2) = ceil(2.5) = 3, 마찬가지로 nz = 3(다음쪽 그림 참조).



sz=1 씩 움직이면 3번 간다

#### 2.2 padding = 1 일 때 앞 레이어에 덧붙일 pad 개수 구하는 방법

앞 레이어의 세로 pny 를 sy 로 나누어 떨어지면 필터 세로 h 에 대해 총 세로 padding 갯수는 h-sy 와 0 중에 더 큰 값이다. pny 를 sy 로 나누어 떨어지지 않으면 pny를 sy 로 나누어 나머 지를 rem 라고 할 때 h-rem 와 0 중에 더 큰 값이다.

가로 방향도 같은 방법으로 구한다

padding 할 값이 짝수이면 위아래 똑같이 나누어 붙이고, 홀수이면 아래로 하나 더 붙인다. 각 노드 값은 0으로 놓는다







예: 앞 레이어 pny = 5, h = 3, sy = 2 이라면 totalPad = max(h-1, 0) = 2 따라서 위아래좌우로 1줄 씩 pad 가 덧 붙여져서 크기가 7 x 7 이미지가 된다(위 그림).

이렇게 padded 된 앞 레이어로 부터 CONV 레이어의 ny 는 ny = ceil(pny/sy) = ceil(2.5) = 3.

따라서 CONV 레이어는 (3,3,10) 가 된다

3. Activation 변환에서 Softmax 변환 수학식

$$y[j] \ = \ \frac{e^{x[j]}}{e^{x[0]} + e^{x[1]} + \dots + e^{x[n-1]}} \qquad \text{with}$$

$$y[0] + y[1] + \dots + y[n-1] = 1$$

따라서 softmax 함수로 표현된 값들은 확률이다

### 4. BN 변환 수학식

이미지들을 학습시킬 때 이미지들이 세로 ny, 가로 nz, 채널 ch 이고 배치 크기가 B 이라면, M = ny x nz x B 이다.

$$m = \frac{1}{M} \sum_{i=0}^{M-1} x[i] \text{ (mean)}$$
(1)  

$$v = \frac{1}{M} \sum_{i=0}^{M-1} (x[i] - m)^2 \text{ (variance)}$$
(2)  

$$\hat{x}[i] = \frac{x[i] - m}{\sqrt{v + \epsilon}} (\epsilon = 0.001) \text{ (normalized node value)}$$
(3)  

$$y[i] = \hat{x}[i] * w[c] + b[c] \text{ (BN transformation)}$$
(4)

where  $c = 0, 1, \dots, ch - 1$ .

BN 변환 매개 변수 N = 0 이면 b[c] 항은 삭제

### III. 단축키 및 레이어 shape 보기

### 1. 레이어 모델 및 3D 레이어

	InputData 28 28 1
	CONV 33641111
	Activation 10
	CONV 33641111
	Activation 10
	Padding 5 5 5 5
	Pooling 220011
-	CONV 3 3 128 1 1 1 1



- 레이어 패널에 나열된 레 이어들은 3D 그림으로 표 시된다. - 각 3D 레이어의 shape 와
- 노란색 띠로 표시되는 pads 는 다음과 같이 볼 수 있다.

2. 3D 레이어에서 선택(Ctrl + Mouse Left button)

(3)
CONV 33641111
Activation 10
CONV 33641111
Activation 1.0
Padding 5 5 5 5
Pooling 220011
CONV 3 3 128 1 1 1 1



- (1) 마우스를 놓고 Ctrl + Mouse Left Button 을 클릭하면 레이 어가 선택된다. 이때 레이어 의 노드들이 보여줄 뿐 아니 라
- (2) 레이어의 shape 가 상단에 표 시된다. 이 예에서 (28,28,1) (3) 레이어 패널에서는 해당 레 이어의 글씨가 초록색으로 변한다.

#### <선택/취소하기 >

**Ctrl+ Mouse Left Button** ٠

### III. 단축키 및 레이어 shape 보기

### 1. 레이어 모델 및 3D 레이어

	InputData 28 28 1
	CONV 33641111
	Activation 10
	CONV 33641111
	Activation 10
	Padding 5 5 5 5
	Pooling 220011
-	CONV 3 3 128 1 1 1 1



- 레이어 패널에 나열된 레 이어들은 3D 그림으로 표 시된다. - 각 3D 레이어의 shape 와
- 노란색 띠로 표시되는 pads 는 다음과 같이 볼 수 있다.

2. 3D 레이어에서 선택(Ctrl + Mouse Left button)

(3)
CONV 33641111
Activation 10
CONV 33641111
Activation 1.0
Padding 5 5 5 5
Pooling 220011
CONV 3 3 128 1 1 1 1



- (1) 마우스를 놓고 Ctrl + Mouse Left Button 을 클릭하면 레이 어가 선택된다. 이때 레이어 의 노드들이 보여줄 뿐 아니 라
- (2) 레이어의 shape 가 상단에 표 시된다. 이 예에서 (28,28,1) (3) 레이어 패널에서는 해당 레 이어의 글씨가 초록색으로 변한다.

#### <선택/취소하기 >

**Ctrl+ Mouse Left Button** ٠

### III. 단축키 및 레이어 shape 보기

### 1. 레이어 모델 및 3D 레이어

	InputData 28 28 1
	CONV 33641111
	Activation 10
	CONV 33641111
	Activation 10
	Padding 5 5 5 5
	Pooling 220011
-	CONV 3 3 128 1 1 1 1



- 레이어 패널에 나열된 레 이어들은 3D 그림으로 표 시된다. - 각 3D 레이어의 shape 와
- 노란색 띠로 표시되는 pads 는 다음과 같이 볼 수 있다.

2. 3D 레이어에서 선택(Ctrl + Mouse Left button)

(3)
CONV 33641111
Activation 10
CONV 33641111
Activation 1.0
Padding 5 5 5 5
Pooling 220011
CONV 3 3 128 1 1 1 1



- (1) 마우스를 놓고 Ctrl + Mouse Left Button 을 클릭하면 레이 어가 선택된다. 이때 레이어 의 노드들이 보여줄 뿐 아니 라
- (2) 레이어의 shape 가 상단에 표 시된다. 이 예에서 (28,28,1) (3) 레이어 패널에서는 해당 레 이어의 글씨가 초록색으로 변한다.

#### <선택/취소하기 >

**Ctrl+ Mouse Left Button** ٠

## APPENDIX IV : Colab 코드 요약 설명

1. Copy Mount 한 내용 설명 2. Training 코드 설명 : Copy Main 한 내용 설명 3. Prediction 코드 설명

### 1. Copy Mount 한 내용 설명 (컴퓨터 하드웨어나 소프트웨어를 모두 이해하지 못해도 컴퓨터를 얼마든지 잘 사용할 수 있으나 이해하고 사용하면 좋은 것 처럼 여기 내용을 그냥 스윽~ 살펴보고 대략적으로도 이해하기를 권장한다)

다음은 python 이라는 컴퓨터 언어로 출력된 Copy Mount 한 내용이다. 자세한 내용을 알 필요는 없다. 그러나 대략적으로 무슨 내용인지를 알고 싶은 학생들을 위하여 설명한다. Python을 몰라도 영어 단어 몇 개 알면 이해 하기 어렵지 않다.



(1) import os 는 아래 (3,4,5) 에서 경로 관련 함수를 사용하기 위하여 도입한다, (2) Google Drive 의 공통된 path 이다 (3) 사용자 계정 Google Drive 의 path 이다 (4) 사용자 계정의 colab 과 MyDrive 가 연결 시키고 "Mounted at /content/drive" 라고 자동 출력된다(표준 출력). 연결이 이미 되었으면 "Google Drive is already mounted."라고 출력한다. 이때 앞의 Drive 엑세스 허용 요청을 받는다. (5) AI Lab Newton 의 Const Panel 에서 설정한 폴더들의 이름을 training\_dir, trained\_dir, prediction dir 로 정한다. (6) 인공지능에서 사용되는 많은 수행할 수 있는 많은 함 수들을 사용하기 위하여 도입하고 tensorflow를 tf 로 부른 다는 뜻이다. Tensorflow 는 python으로 코딩 됨 (7) Colab 에서 GPU가 있는지 없는지 체크한다. 있으 면 GPU: /device:GPU:0 등으로 출력하고, 없으면 "No GPU available"이라 출력한다. (8) 출력된 내용들이 아래 표시된다.

### 2. Training 코드 설명 : Copy Main 한 내용 설명 (앞의 Copy Mount 처럼 대략 무슨 내용 인 지만 대충 훑어보기 권장한다)

"""<model> 현재 학습할 파일 : sample\_layer10.py 의 model 내용이다. 본 AI Lab Newton 에 익숙한 다음에는 파일을 직 접 수정할 수 있다[see Appendix]. trv: 👞 import numpy as np try import tensorflow as tf from tensorflow import keras • • • import os except ... import shutil import time 이 부분은 try 부 import sys import matplotlib.pyplot as plt 분에서 오류 나 import matplotlib.image as mpimg 면 except 부분 from sklearn.utils import shuffle from sklearn.model\_selection import train\_test\_split 에 무슨 오류가 from skimage.io import imread 나는지 체크하 from skimage.transform import resize import pathlib 는 기능이다 from tensorflow.keras.utils import to\_categorical import tempfile from glob import glob import random except Exception as e: print("Error: (Import) ", e)

#<import>### 이 부분은 소위 인공지능을 코 딩하고 학습하는 하나의 프레임인 tensorflow 를 사용하기 위하여 미리 코딩 된 다양한 함수들을 불러오는 부분이다.

```
try
   t1 = time.ctime()
   CLASS_NUM = 10
   EPOCH_NUM = 500
   BATCH_NUM = 64
   LEARNING_RATE = 0.002
   imY = 28
   imZ = 28
   imCH = 1
   AUTOTUNE = tf.data.AUTOTUNE
   trained_dir_filepath = os.path.join(trained_dir, "sample_layer10")
   class_names = np.array(sorted(os.listdir(training_dir)))
except Exception as e:
```

print("Error: (Constants) ", e)

#<Constants>### 앞에서 settings 한 내용들을 코드로 바꾼 것이다.

try:

def get\_dataset\_partitions\_tf(ds, ds\_size, train\_split=0.8, val\_split=0.1, test\_split=0.1, shuffle=True, shuffle\_size=T0000):

assert (train\_split + test\_split + val\_split) == 1

if shuffle:

```
ds = ds.shuffle(shuffle_size, seed=12)
```

```
train_size = int(train_split * ds_size)
val_size = int(val_split * ds_size)
```

```
train_ds = ds.take(train_size)
val_ds = ds.skip(train_size).take(val_size)
test_ds = ds.skip(train_size).skip(val_size)
```

```
return train_ds, val_ds, test_ds
```

```
def get_label(file_path):
    parts = tf.strings.split(file_path, os.path.sep)
    one_hot = parts[-2] == class_names
```

return one\_hot

\* 여기를 수정할 필요는 없다. 나중에 인공 지능을 전문적으로 코딩하는 사용자는 이러 한 코드를 수정할 수 있다. 

 #<Dataset Processing Functions> ###

 여기서 눈 여겨 볼 것은

 전체 데이터 셋에 대해 학습하는 것이 아니라

 전체의 10%는 valuation 데이터로 사용하고

 10%는 test 데이터로 사용하고

 나머지 80% 만 train 데이터로 학습한다는 뜻

 이다. 여기서

Validation 데이터는 전체 데이터 셋에 대해 1회(1 epoch) 학습할 때 마다 과적합 (overfitting; 즉 너무 과하게 억지로 맞춰지 는 것)이 발생하고 있는지 판단한다. 과적 합이 일어나지 않게 모델을 조절한다.



Test 데이터는 학습이 종료된 후 모델의 정확 도를 평가하거나 임의의 입력 데이터에 대해 결과를 예측하기 위해 사용된다. try:

saved\_list\_ds = tf.data.Dataset.list\_files(str(training\_dir+'/\*/\*'), shuffle=True)
image\_count = saved\_list\_ds.cardinality().numpy()
print('image\_count:', image\_count)

```
class_names = np.array(sorted(os.listdir(training_dir)))
print('class_names: ', class_names)
```

train\_ds0, val\_ds0, test\_ds0 = get\_dataset\_partitions\_tf(saved\_list\_ds, image\_count)

x\_train\_maped = train\_ds0.map(process\_path, num\_parallel\_calls = AUTOTUNE)
x\_val\_maped = val\_ds0.map(process\_path, num\_parallel\_calls = AUTOTUNE)
x\_test\_maped = test\_ds0.map(process\_path, num\_parallel\_calls = AUTOTUNE)

train\_num = len(x\_train\_maped)
val\_num = len(x\_val\_maped)
test\_num = len(x\_test\_maped)

train\_ds = prepare\_for\_training(x\_train\_maped)
val\_ds = prepare\_for\_training(x\_val\_maped)
test\_ds = prepare\_for\_training(x\_test\_maped)

except Exception as e: print("Error: (Load and Process Dataset)", e) #<Load and Process Dataset>###

Drive 에 저장되어 있는 데이터들을 Colab 으로 Load 하고, 위의 Data processing functions 를 이용하여 아래 #<fit>### 에 서 사용할 학습 시키는 형태로 바꾼다. trv:

```
def define_model():
        inputs = keras.Input(shape = (28,28,1))
       x = inputs
       x = keras.layers.Conv2D(filters = 10, kernel_size = (2,2), strides = (1,1), padding = 'valid', use_bias = True)(x)
       x = keras.layers.BatchNormalization(axis = -1, momentum=0.99, epsilon=0.001, center = True)(x)
       x = keras.lavers.Activation('relu')(x)
       x = keras.layers.Conv2D(filters = 10, kernel_size = (3,3), strides = (1,1), padding = 'valid', use_bias = True)(x)
       x = keras.layers.BatchNormalization(axis = -1, momentum=0.99, epsilon=0.001, center = True)(x)
       x = keras.lavers.Activation('relu')(x)
       x = keras.layers.Conv2D(filters = 10, kernel_size = (2,2), strides = (1,1), padding = 'valid', use_bias = True)(x)
       x = keras. layers. Flatten()(x)
       x = keras.layers.Dense(units = 10, use_bias = True)(x)
       x = keras.layers.Activation('softmax')(x)
       outputs = x
        model = keras.Model(inputs = inputs, outputs = outputs)
       return model
except Exception as e:
```

print("Error: (Define Model)", e)

#<Model>### 사용자가 AI Lab Newton 으로 모델링한 것이 여기서 tensorflow 코드로 변환되었다.

#### <딥러닝 과정>





#<Strategy & Optimizer>### GPU 가 몇 개 있는지 확인 : 없으면 CPU 사용, 있으면 GPU 사용하게 환경 설정하고 Optimizer 를 설정한다. → 이것을 이용하여 Compile 한다.

Compile 한다는 것은 모델을 학습 전 optimizer, loss, 모델 성능 평가 지표(예: accuracy) 등을 설정하는 것이다.

```
try:
```

```
options = tf.data.Options()
options.experimental_distribute.auto_shard_policy = tf.data.experimental.AutoShardPolicy.OFF
train_ds = train_ds.with_options(options)
val_ds = val_ds.with_options(options)
test_ds = test_ds.with_options(options)
```

```
except Exception as e:
    print("Error: (Disable AutoShard) ", e)
```

```
#<Disable AutoShard>###
AutoShard는 다수의 GPU 나 TPU를 사용할 경우
Tensorflow 가 데이터를 자동으로 적절하게 분할하
여 계산하게 한다.
그러나 Colab 에서는 CPU 또는 GPU 를 1개 사용하
니까 분산 처리 설정할 정도로 복잡하지 않아 이
기능을 껐다.
```

```
try:
   check_point = keras.callbacks.ModelCheckpoint(filepath = trained_dir_filepath.
                                     monitor = 'val_accuracy',
                                     mode = 'max',
                                     save_weights_only = True,
                                     save_best_only = True,
                                                                                #<CheckPoint & Callbacks> ###
   early_stopping = keras.callbacks.EarlyStopping(monitor = 'val_accuracy', patience = 2,
                                                                                 학습된 결과 중 무엇을 저장할지 언제 학습
                restore best weights = True )
                                                                                을 끝낼지를 정한다. 보편적인 내용이다.
   reduce Ir = keras.callbacks.ReduceLROnPlateau(monitor='val loss', factor=0.2.
                patience=5. min lr=0.0001)
   train_callbacks = [check_point, early_stopping, reduce_lr]
except Exception as e:
   print("Error: (CheckPoint & Callbacks) ", e)
```


## <딥러닝 과정>



#<Evaluate>### 학습이 종료되었을 때 모델의 학습 결과 를 평가한다.

학습이 종료되었을 때 test 데이터에 대 하여 모델의 학습 결과를 평가한다.



Loss(손실) : 인공지능 학습은 loss 값을 작 게 하는 목표를 갖고 학습한다. 따라서 이 값이 작을 수록 목표에 많이 다가갔음 을 뜻한다. Loss 이 0 에 가까울 수록 목표 에 근접.

Accuracy(정확도) : test 데이터에 대하 여 '몇개나 예측이 맞았나' 를 실제로 해본 결과이다. Accuracy = 1 이면 모두 맞춘 것이다. 예를 들어 0.81 이면 81% 맞추었다는 뜻. try:

loss = history.history['loss']
v\_loss = history.history['val\_loss']

acc = history.history['accuracy']
v\_acc = history.history['val\_accuracy']

epochs = range(1, len(loss)+1)

## # 손실 값들의 최솟값을 찾고, 이를 기준으로 하한값 설정

min\_loss = min(min(loss), min(v\_loss)) lower\_bound = np.floor(min\_loss) # 최솟값보다 낮은 정수 값

### # Loss 최댓값 계산 및 ylim 설정

max\_loss = max(max(loss), max(v\_loss)) + 1.0

fig = plt.figure(figsize=(9, 5)) plt.subplot(1, 2, 1) plt.yscale('log') plt.plot(epochs, loss, linestyle='--', linewidth=3, color='orange', alpha=0.7, label='Train Loss') plt.plot(epochs, v\_loss, linestyle='--', linewidth=2, color='lime', alpha=0.8, label='Val Loss') plt.ylim(lower\_bound, max\_loss) plt.xlabel('Epochs', fontsize=11) plt.ylabel('Loss', fontsize=12) plt.legend(fontsize=12) plt.subplot(1, 2, 2) plt.plot(epochs, acc, linestyle='-', linewidth=3, color='orange', alpha=0.7, label='Train Acc') plt.plot(epochs, v\_acc, linestyle='-', linewidth=2, color='lime', alpha=0.8, label='Val Acc') plt.xlabel('Epochs', fontsize=11) plt.ylabel('Accuracy', fontsize=12) plt.legend(fontsize=12) plt.tight\_layout() plt.show()

#<Draw History>###

앞에서 학습한 결과를 epoch 단위로 그 래프를 그려준다

## 3. Prediction 코드 설명

try÷

```
def load_and_preprocess_image(path):
    img = tf.io.read_file(path)
    img = decode_img(img)
```

return img

def decode\_img(img):

img = tf.image.decode\_jpeg(img, channels=imCH) img = tf.image.convert\_image\_dtype(img, tf.float32) img = tf.image.resize(img, [imY, imZ])

return img

```
# 이미지 파일 경로 수집

def get_image_paths(root_dir):

    image_paths = []

    for root, dirs, files in os.walk(root_dir):

        for file in files:

            if file.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif', '.tiff', '.webp', '.svg')):

            image_paths.append(os.path.join(root, file))
```

return image\_paths

# 하위 폴더 확인

```
def has_subdirectories(directory):
    return any(os.path.isdir(os.path.join(directory, d)) for d in os.listdir(directory))
```

except Exception as e: print("Error: (Load and Preprocess and ImagePath)", e)

#### try:

### # 이미지 예측 함수

def predict\_image(model, image\_path):

image = load\_and\_preprocess\_image(image\_path) image = tf.expand\_dims(image, 0) # 배치 차원 추가 predictions = model.predict(image)

return np.argmax(predictions, axis=1)

## # 전체 이미지에 대한 예측 및 결과 출력

def predict\_all\_images(model, image\_paths, batch\_size=None): # 이미지 데이터셋 생성 dataset = tf.data.Dataset.from\_tensor\_slices(image\_paths)

dataset = dataset.map(load\_and\_preprocess\_image)

#### #배치 처리

if batch\_size: dataset = dataset.batch(batch\_size) else: dataset = dataset.batch(len(image\_paths))

### # 예측 진행 상황 표시를 위한 설정

total\_images = len(image\_paths) progress\_interval = max(total\_images // 10, 1) # 예측 진행 상황 표시 print("예측을 진행 중입니다...", end="")

#### # 예측 수행 predictions = model.predict(dataset, verbose=0)

### # 전체 이미지에 대한 예측 및 결과 출력

def predict\_all\_images(model, image\_paths, batch\_size=None): # 이미지 데이터셋 생성

dataset = tf.data.Dataset.from\_tensor\_slices(image\_paths)
dataset = dataset.map(load\_and\_preprocess\_image)

### #배치 처리

if batch\_size: dataset = dataset.batch(batch\_size) else:

dataset = dataset.batch<mark>(len</mark>(image\_paths))

#### # 예측 진행 상황 표시를 위한 설정

total\_images = len(image\_paths) progress\_interval = max(total\_images // 10, 1) # 예측 진행 상황 표시 print("예측을 진행 중입니다...", end="")

#### # 예측 수행 predictions = model.predict(dataset, verbose=0)

#### # 결과 요약

correct\_predictions = 0
for i, image\_path in enumerate(image\_paths):
 true\_label = os.path.basename(os.path.dirname(image\_path))
 pred\_label = np.argmax(predictions[i])
 if str(pred\_label) == true\_label:
 correct\_predictions += 1

if i % progress\_interval == 0 : print(".", end="") # 진행 상황 표시

## # 최종 결과 출력

print(f"전체 이미지 수: { total\_images}, 정확히 예측된 이미지 수: { correct\_predictions}") print(f"Accuracy: { correct\_predictions / total\_images}")

## <딥러닝 과정>



#### try≑

# 학습한 weights 를 불러온다 model.load\_weights(trained\_dir\_filepath)

# 이미지 Path 얻는다 image\_paths = get\_image\_paths(prediction\_dir)

## # 옵션들

if has\_subdirectories(prediction\_dir): options = 'yes' print('하위 디렉토리가 있습니다.') else: options = 'no' print('하위 디렉토리가 없고 파일만 있습니다.')

#### # 예측

user\_input\_predict(model, image\_paths, options)

## except Exception as e: print("Error: (Load Weights and Predict)", e)

## APPENDIX V: Colab 오류 설명

Error: (Predict) Unsuccessful TensorSliceReader constructor: Failed to find any matching files for /content/drive/MyDrive/trained/sample\_layer10\_flowers

Error: (Predict) Failed to find any matching files for ... 오류: (Predict에서 발생) 이유는 ...에 해당되는 파일을 찾을 수 없다. 즉 flowers 에 매칭되는 파 일을 찾을 수 없기 때문에 오류가 발생. 실제 있는 파일들은 mnist 파일들.

```
1 import os
2 colab_path = '/content/drive'
3 colab_path1 = os.path.join(colab_path, "MyDrive")
4 print(colab_path1)
5 if not os.path.exists(colab_path1):
6 from google.colab import drive
7 drive.mount(colab_path)
8 else:
9 print('Google Drive is already mounted.')
```

## /content/drive/MyDrive

MessageError

Traceback (most recent call last)

```
<ipython-input-1-060f4a82511b> in <cell line: 5>()
```

5 if not os.path.exists(colab\_path1):

- 6 from google.colab import drive
- ----> 7 drive.mount(colab\_path)

## 8 else:

9 print('Google Drive is already mounted.')

🗘 3 frames –

/usr/local/lib/python3.10/dist-packages/google/colab/\_message.py in read\_reply\_from\_input(message\_id, timeout\_sec)

101 ): 102 if 'error' in reply: --> 103 raise MessageError(reply['error']) 104 return reply.get('data', None) 105

MessageError: Error: credential propagation was unsuccessful

## APPENDIX VI : AI Lab Newton 을 이용한 개발 방법

## 1. 초급 개발자

인공지능에 대한 기본 개념이 없는 초급 개발자 레이어에 대한 개념 등에 익숙하도록 본 AI Lab Newton 이용

## 2. 중급 개발자

인공지능에 대하여 기본 개념을 이해하고 AI Lab Newton 에 나오는 기본 레이어에 대한 개념 등에 익숙한 중급 개발자

Sample을 직접 작성 가능하다.

## <Sample 작성 방법 예>

## 메모장 등 에디터를 열고 py(python 파일을 의미) 확장자로 끝 나는 파일(예: sample.py)을 만들어 직접 작성한다.



## 3. 고급 개발자

인공지능에 대하여 기본 개념을 이해할 뿐 아니라 tensorflow 로 코딩할 수 있는 개발자들은 UDM (고급 버전) 을 tensorflow 로 직접 작성하여 AI Lab Newton 을 이용한다.

학생들이 인공지능을 어려워 하는 이유는 기본 개념이나 원리를 이해 못하기 때문 이다. 본 AI Lab Newton에 익숙하면 인공지능 개념과 원리를 이해할 수 있기 때문에 python 언어는 물론 tensorflow 언어도 쉽게 배울 수 있다. 이를 이용하여 인공지능 전문 프로그래머가 될 수 있다.

## **APPENDIX VII :** Datasets & Samples

본 AI Lab Newton에서 제공하는 Datasets 과 models 에 대해 소개한다. 사용자는 본 데이터 셋과 모델들을 이용하여 AI Lab Newton 을 공부할 수 있다.

## **1. MNIST Dataset**

	내용	비고
출처	http://yann.lecun.com/exdb/mnist/	
이미지 크기	28 픽셀 x 28 픽셀, 즉 (28,28,1)	단색 <i>,</i> 1 채널
데이터 갯수	Train : 60000, Test : 10000	
이미지 종류	10	





## 2. FASHION-MNIST Dataset

	내용	비고
출처	https://research.zalando.com/project/ fashion_mnist/fashion_mnist/	
이미지 크기	28 픽셀 x 28 픽셀, 즉 (28,28,1)	단색
데이터 갯수	Train : 60000, Test : 10000	
이미지 종류	10 (Class Num)	

MNIST 는 인공지능 알고리즘의 유효성을 진단하는 가장 처음 사용된 dataset 이었다. 당시 "MNIST" 에서 작 동되지 않는 알고리즘은 다른 데이터셋에서도 잘 작동되지 않을 것이라 생각하였다. 이에 대해 FASHION-MNIST 는 MNIST 에서 작동된다 하더라도 다른 데이터 셋에서는 작동 되지 않을 수도 있다는 것을 보여주기 위해 만들어졌다(Zalando 논문).



라벨	설명
0	티셔츠/상의
1	바지
2	점퍼
3	드레스
4	코트
5	샌들
6	셔츠
7	운동화
8	가방
9	앵클 부츠

## **3. CIFAR10**

	내용	비고
출처	https://www.cs.toronto.edu/~kriz/cifar.html	
이미지 크기	32 픽셀 x 32 픽셀 x 3 채널, 즉 (32,32,3)	칼라
데이터 갯수	Train : 50000, Test : 10000	
이미지 종류	10 (Class Num)	

airplane	the second		1	- 3			-
automobile			1			-	*
bird	R J	2		4.17		1	V
cat					A.	A.S.	1
deer	1			YY	1	-	
dog	We de	1		<b>9</b>	1	1	N.
frog		3			37		5.00
horse	the aff	AM 2			N.	(a)	1
ship	2	-		- 2	NET	p-1	
truck					1	-	dela

## 4. CIFAR100

	내용	비고
출처	https://www.cs.toronto.edu/~kriz/cifar.html	
이미지 크기	32 픽셀 x 32 픽셀 x 3 채널, 즉 (32,32,3)	칼라
데이터 갯수	Train : 50000, Test : 10000	
이미지 종류	100 (Class Num)	



#### Superclass aquatic mammals fish flowers food containers fruit and vegetables household electrical devices household furniture insects large carnivores large man-made outdoor things large natural outdoor scenes large omnivores and herbivores medium-sized mammals non-insect invertebrates people reptiles small mammals trees vehicles 1 vehicles 2

#### Classes

beaver, dolphin, otter, seal, whale aquarium fish, flatfish, ray, shark, trout orchids, poppies, roses, sunflowers, tulips bottles, bowls, cans, cups, plates apples, mushrooms, oranges, pears, sweet peppers clock, computer keyboard, lamp, telephone, television bed, chair, couch, table, wardrobe bee, beetle, butterfly, caterpillar, cockroach bear, leopard, lion, tiger, wolf bridge, castle, house, road, skyscraper cloud, forest, mountain, plain, sea camel, cattle, chimpanzee, elephant, kangaroo fox, porcupine, possum, raccoon, skunk crab, lobster, snail, spider, worm baby, boy, girl, man, woman crocodile, dinosaur, lizard, snake, turtle hamster, mouse, rabbit, shrew, squirrel maple, oak, palm, pine, willow bicycle, bus, motorcycle, pickup truck, train lawn-mower, rocket, streetcar, tank, tractor

## **5. FLOWERS**

	내용	비고
출처	https://www.tensorflow.org/tutorials/ load_data/images	
이미지 크기	500 픽셀 x 375 픽셀 x 3 채널, 즉 (500,375,3) 320 픽셀 x 263 픽셀 x 3 채널, 즉 (320,263,3) 등 다양한 크기	칼라
데이터 갯수	Train : 3670	
이미지 종류	5 (Class Num)	



이미지 크기가 다른 것들로 구성되어 있다. 이때 InputData 의 shape를 예를 들어 (180, 180, 3)로 할 수 있다.

## **APPENDIX VIII : History**

인공지능 역사에 대해서 인터넷에 많은 자료들이 있다. 여기서 ChatGPT 를 이용하여 찾아 본 것들을 참고로 올려 놓았다.



## <딥러닝의 기념비적 논문들 >

## <mark>답러닝의 발전과 대중화에 기여한 몇 <u>가지 기념비적인 논문과 간단한 특징</u>에 대해 소개해드리 겠습니다:</mark>

- 1. "Deep Learning" (Hinton et al., 2006):
  - 특징: 딥러닝의 개념과 중요성을 강조한 논문. 제한된 볼츠만 머신(RBM)을 사용하여 깊은 신경망의 사전 훈련을 소개함.
- "ImageNet Classification with Deep Convolutional Neural Networks" (Krizhevsky et al., 2012):
  - 특징: 알렉스넷(AlexNet)이라는 딥러닝 모델을 제안하고 ImageNet 대회에서 우승함. 딥 러닝을 컴퓨터 비전 분야에 대중화시키는 역할을 했으며, 깊은 CNN 구조와 GPU 활용을 통해 높은 성능을 달성함.
- 3. "Playing Atari with Deep Reinforcement Learning" (Mnih et al., 2013):
  - 특징: 딥 Q-네트워크(Deep Q-Network, DQN)라는 개념을 소개하고, 딥러닝을 강화학습에 적용하여 비디오 게임에서 인간 수준의 성능을 달성함.
- 4. "Generative Adversarial Networks" (Goodfellow et al., 2014):
  - 특징: 생성적 적대 신경망(GAN)을 소개하여 이미지와 같은 데이터를 생성하는 방법을 제시함. GAN은 생성자와 판별자로 구성되며, 두 모델이 적대적인 경쟁을 통해 서로 학습 함.
- 5. "Deep Residual Learning for Image Recognition" (He et al., 2015):
  - 특징: 잔차 학습(ResNet)이라는 개념을 도입하여, 더 깊은 신경망에서의 효과적인 학습 과 경사 소실 문제를 해결함. 많은 컴퓨터 비전 과제에서 높은 성능을 보여줌.
- 6. "Attention Is All You Need" (Vaswani et al., 2017):
  - 특징: 트랜스포머(Transformer)라는 딥러닝 아키텍처를 소개하여, 기존의 순환 신경망 (RNN) 대신 어텐션 메커니즘을 활용하여 자연어 처리 태스크에서 뛰어난 성능을 달성함. 자연어 처리 분야에서 혁신적인 모델 구조로 인정받음.

## 네, 딥러닝의 기념비적 논문들은 다음과 같습니다.

- 1958년: Frank Rosenblatt의 "Perceptron: A Probabilistic Model for Information Storage and Organization"
- 1969년: Geoffrey Hinton의 "Learning Internal Representations by Error Propagation"
- 1986년: Yann LeCun의 "Backpropagation Applied to Handwritten Zip Code Recognition"
- 1998년: Yann LeCun의 "Gradient-Based Learning Applied to Document Recognition"
- 2012년: Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton의 "ImageNet Classification with Deep Convolutional Neural Networks"

이 논문들은 딥러닝의 발전에 큰 영향을 미쳤으며, 이후 발표된 딥러닝 모델에 큰 영향을 미쳤습니다.

- Frank Rosenblatt의 "Perceptron: A Probabilistic Model for Information Storage and Organization"은 퍼셉트론의 개념을 소개한 논문입니다. 퍼셉트론은 단층 신경망으로, 이미지 분류, 텍스트 인식 등의 분야에서 사용되고 있습니 다.
- Geoffrey Hinton의 "Learning Internal Representations by Error Propagation"은 역전파 알고리즘의 개념을 소개한 논문입니다. 역전파 알고리즘은 신경망의 학습에 사용되는 중요한 알고리즘입니다.
- Yann LeCun의 "Backpropagation Applied to Handwritten Zip Code Recognition"은 역전파 알고리즘을 사용하여 손글씨 우편번호 인식 문제를 해결한 논문입니다. 이 논문은 딥러닝이 실제 문제에 적용될 수 있음을 보여주었습니다.
- Yann LeCun의 "Gradient-Based Learning Applied to Document Recognition"은 딥러닝을 사용하여 문서 인식 문제 를 해결한 논문입니다. 이 논문은 딥러닝이 컴퓨터 비전 분야에서 주류 기술로 자리 잡는 계기가 되었습니다.
- Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton의 "ImageNet Classification with Deep Convolutional Neural Networks"은 알렉스넷을 발표한 논문입니다. 알렉스넷은 ILSVRC에서 우승을 차지하며 딥러닝의 시대를 열었습니다.

이러한 논문들은 딥러닝의 발전에 큰 영향을 미쳤으며, 이후 발표된 딥러닝 모델에 큰 영향을 미쳤습니다.



### ImageNet Large Scale Visual Recognition Challenge(ILSVRC)는 2010년부터 매년 개최되는 컴퓨터 비전 대회입니다. 이 대회는 1000개의 클래스에 속하는 1400만 개의 이미지 데이터 세트에서 객체를 식별하는 정확성을 측정합니다.

ILSVRC는 컴퓨터 비전 연구의 발전에 있어 중요한 역할을 해왔습니다. 이 대회는 딥러닝의 발전을 가속화하는 데 도움이 되었으며, 딥러닝 모델의 정확성을 크게 향상시켰습니다.

2012년부터 2021년까지 ILSVRC에서 우승한 팀은 다음과 같습니다.

- 2012: AlexNet (Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton)
- 2013: VGGNet (Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun)
- 2014: GoogLeNet (Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna)
- 2015: ResNet (Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun)
- 2016: Inception-v4 (Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Jonathon Shlens)
- 2017: DenseNet (Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger)
- 2018: NASNet (Quoc V. Le, Oriol Vinyals, Quoc V. Le, Trevor Darrell)
- 2019: EfficientNet (Mingxing Tan, Quoc V. Le)
- 2020: MixNet (Shang-Hua Gao, Han Zhang, Jing Yu, Wei Yang, Yi Li, Jifeng Dai)
- 2021: Swin Transformer (Zeming Zhang, Xiangyu Zhang, Zhoutong Zhang, Han Zhang, Yizhou Sun)

ILSVRC는 컴퓨터 비전 분야의 발전에 있어 중요한 역할을 해왔습니다. 이 대회는 딥러닝의 발전을 가속화하는 데 도움이 되었으며, 딥러닝 모델의 정확성을 크게 향상시켰습니다.

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)는 매년 수많은 참가팀이 경쟁 하는 대회입니다. 아래는 ILSVRC에서 우승한 몇 가지 팀들을 소개합니다:
  - 1. 알렉스넷(AlexNet) 2012년 우승:
    - 팀 멤버: Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton
    - 특징: 알렉스넷은 딥러닝 모델로서 처음으로 ILSVRC에서 우승한 모델입니다. 2012년에 도입된 알렉스넷은 깊은 신경망 구조와 GPU를 활용하여 높은 성능을 달성하여 컴퓨터 비전 분야에서 딥러닝의 성공적인 적용을 입증했습니다.
  - 2. ZF넷(ZFNet) 2013년 우승:
    - 팀 멤버: Matthew Zeiler, Rob Fergus
    - 특징: ZF넷은 알렉스넷의 변형 버전으로, 2013년 ILSVRC에서 우승한 모델입니다. ZF넷 은 알렉스넷보다 더 깊은 신경망 구조를 사용하여 성능을 향상시켰으며, 이후 다양한 딥 러닝 모델의 발전에 영감을 주었습니다.
  - 3. VGG넷(VGGNet) 2014년 우승:
    - 팀 멤버: Karen Simonyan, Andrew Zisserman
    - 특징: VGG넷은 매우 깊은 신경망 구조로 이루어져 있으며, 작은 필터 크기와 깊은 층 구 조를 통해 높은 정확도를 달성했습니다. VGG넷은 딥러닝 모델의 깊이에 대한 중요성을 강조하였으며, 이후 다양한 모델의 설계에 영향을 주었습니다.
  - 4. GoogLeNet 2014년 우승:
    - 팀 멤버: Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Andrew Rabinovich
    - 특징: GoogLeNet은 인셉션 모듈(Inception module)이라는 독특한 구조를 도입하여 계 산 효율성을 높였습니다. 작은 필터 크기를 병렬적으로 사용하여 깊이와 너비를 모두 키 워 성능을 향상시켰으며, ILSVRC 2014에서 우승했습니다.

- 5. ResNet 2015년 우승:
  - 팀 멤버: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
  - 특징: ResNet은 잔차 학습(Residual learning) 개념을 도입하여 매우 깊은 신경망에서의 학습을 용이하게 했습니다. 잔차 연결을 통해 경사 소실 문제를 해결하고, ILSVRC 2015 에서 우승하며, 이후 많은 컴퓨터 비전 태스크에서 널리 사용되는 모델이 되었습니다.

이 외에도 다양한 팀들이 ILSVRC에서 우수한 성과를 거두었으며, 이를 통해 딥러닝 모델의 발 전과 컴퓨터 비전 분야의 발전에 기여하였습니다.



### <u>컨벌루션 레이어</u>가 처음 발표된 논문은?

\$

실제로, 컨벌루션 레이어(Convolutional Layer)는 컴퓨터 비전 분야에서 처음 소개되었습니다. 컨벌루션 레이어의 개념을 제안하고 컨벌루션 신경망(CNN)을 구성하는 방법을 상세히 설명한 기념비적인 논문은 "Gradient-based learning applied to document recognition"이며, 저자는 Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner 입니다. 이 논문은 1998년에 발표되 었으며, 손글씨 숫자 인식을 위해 LeNet-5 아키텍처를 소개했습니다.

LeNet-5은 컨벌루션 레이어를 사용하여 이미지에서 특징을 학습하고 추출하는 효과적인 방법 을 보여주었습니다. 이 논문은 컨벌루션 레이어의 효과와 중요성을 강조하며, 컴퓨터 비전 분야 에서 컨벌루션 신경망의 핵심 구성 요소로 자리잡게 되었습니다.

따라서 "Gradient-based learning applied to document recognition" 논문은 컨벌루션 레이어 의 개념을 처음 제시하고, 컨벌루션 <u>신경망의 널리 사용되는 기</u>본 요소로 인식되는 논문입니다.

G Regenerate response



import tensorflow as tf

# LeNet-5 모델 정의 model = tf.keras.models.Sequential([ tf.keras.layers.Conv2D(6, kernel\_size=(5, 5), activation='relu', input\_shape=(32, 32, 1)), tf.keras.layers.MaxPooling2D(pool\_size=(2, 2)), tf.keras.layers.Conv2D(16, kernel\_size=(5, 5), activation='relu'), tf.keras.layers.MaxPooling2D(pool\_size=(2, 2)), tf.keras.layers.Flatten(), tf.keras.layers.Flatten(), tf.keras.layers.Dense(120, activation='relu'), tf.keras.layers.Dense(84, activation='relu'), tf.keras.layers.Dense(10, activation='softmax') ])

```
# 모델 컴파일
```

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accu

# 모델 요약 정보 출력 model.summary()



## <AlexLet (2012) >

## 알렉스넷에 대해 설명해줘

View other drafts 🛛 🗸

◆ 알렉스넷은 2012년 Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton이 발표한 인공 신경망 모델입니다. 알렉스넷은 ImageNet Large Scale Visual Recognition Challenge(ILSVRC)에서 15.3%의 에러율로 우승을 차지하며 딥러닝의 시대를 열었습니다. 알렉스넷은 LeNet-5를 기반으로 설계되었지만, ReLU 활성화 함수, 드롭아웃, 오버랩핑 풀링 등의 새로운 기 술을 도입하여 성능을 크게 향상시켰습니다. 알렉스넷의 등장은 딥러닝이 컴퓨터 비전 분야에서 주류 기술로 자리 잡는 계기가 되었습니다.

알렉스넷은 다음과 같은 특징을 가지고 있습니다.

- ReLU 활성화 함수 사용
- 드롭아웃 사용
- 오버랩핑 풀링 사용
- GPU 사용
- 1,000만 개의 파라미터 사용

알렉스넷은 ILSVRC에서 우승을 차지하며 딥러닝의 시대를 열었습니다. 알렉스넷의 등장은 딥러닝이 컴퓨터 비전 분야에 서 주류 기술로 자리 잡는 계기가 되었습니다. 알렉스넷은 이후 발표된 딥러닝 모델에 큰 영향을 미쳤으며, 현재에도 널리 사용되고 있습니다. 알렉스넷(AlexNet)은 2012년에 발표된 딥러닝 모델로, 이미지 분류를 위해 사용되는 컨벌루션 신경망(CNN) 아키텍처입니다. 알렉스넷은 컴퓨터 비전 분야에서 큰 인상을 남기며 딥러닝의 대중화에 기여한 모델로 알려져 있습니다.

알렉스넷은 헤일리 알렉스 리즌후브(Hinton Alex Krizhevsky) 등이 개발한 모델로, ILSVRC(ImageNet Large Scale Visual Recognition Challenge) 대회에서 우승하면서 주목을 받았습니다. 이 모델은 대규모 데이터셋인 ImageNet에서 1,000개의 다양한 클래스를 분류하 는 데에 사용되었습니다.

알렉스넷은 이전에 사용되던 전통적인 컴퓨터 비전 모델보다 깊고 넓은 신경망 아키텍처를 가 지고 있습니다. 주요 특징은 다음과 같습니다:

- 컨벌루션 레이어와 풀링 레이어의 깊은 스택: 알렉스넷은 5개의 컨벌루션 레이어와 풀링 레 이어를 깊게 쌓아올렸습니다. 이를 통해 점진적으로 복잡한 이미지 특징을 학습하고 추출할 수 있었습니다.
- 활성화 함수로 <u>ReLU 사용</u>: 알렉스넷은 컨벌루션 레이어와 완전 연결 레이어에서 활성화 함 수로 ReLU(Rectified Linear Unit)를 사용했습니다. 이는 비선형성을 도입하고, 그레디언트 소실 문제를 완화하여 학습을 더욱 효과적으로 만들었습니다.
- 드롭아웃(Dropout) 정규화: 알렉스넷은 드롭아웃을 사용하여 모델의 과적합을 완화했습니
   다. 드롭아웃은 학습 중에 랜덤하게 일부 뉴런을 비활성화시켜 모델의 일반화 능력을 향상 시킵니다.
- 4. GPU를 이용한 병렬 처리: 알렉스넷은 NVIDIA의 그래픽 처리 장치(GPU)를 활용하여 모델의 학습과 추론 속도를 대폭 향상시켰습니다.

알렉스넷(AlexNet)은 큰 규모의 모델이므로, 전체 모델 구현을 위한 TensorFlow 코드는 상당히 길고 복잡합니다. 하지만 간단한 예시로 알렉스넷의 일부를 구현한 TensorFlow 코드를 제공해 드리겠습니다. 다음은 입력 이미지 크기가 227x227이고, 1000개의 클래스를 분류하는 알렉스 넷의 일부를 구성한 예시 코드입니다:

### import tensorflow as tf

## # 알렉스넷 모델 정의

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(96, kernel_size=(11, 11), strides=(4, 4), activation='relu',
input shape=(227, 227, 3)),
  tf.keras.layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),
  tf.keras.layers.Conv2D(256, kernel size=(5, 5), activation='relu', padding='same'),
  tf.keras.layers.MaxPooling2D(pool size=(3, 3), strides=(2, 2)),
  tf.keras.layers.Conv2D(384, kernel size=(3, 3), activation='relu', padding='same'),
  tf.keras.layers.Conv2D(384, kernel size=(3, 3), activation='relu', padding='same'),
  tf.keras.layers.Conv2D(256, kernel_size=(3, 3), activation='relu', padding='same'),
  tf.keras.layers.MaxPooling2D(pool size=(3, 3), strides=(2, 2)),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(4096, activation='relu'),
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Dense(4096, activation='relu'),
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Dense(1000, activation='softmax')
])
```

## # 모델 컴파일

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# 모델 요약 정보 출력 model.summary() 위의 코드는 알렉스넷 모델의 일부를 구성한 것으로, 실제 알렉스넷 모델의 전체 구조와 매개변 수 설정에는 <u>더 많은 레이어와 세부 구성이 포함됩</u>니다. 하지만 이 예시 코드를 참고하여 알렉 스넷 모델을 구현하고 컴파일할 수 있습니다.

참고로, 알렉스넷은 ImageNet 데이터셋과 같은 대규모 데이터셋에서 학습되는 것이 일반적입 니다. 따라서 실제로 알렉스넷을 사용하려면 <u>데이터 전처리</u>와 학습 과정이 필요합니다. 이 예시 코드는 모델의 구조를 보여주기 위한 간단한 예시일 뿐, 완전한 알렉스넷 모델의 학습을 위해서 는 적절한 데이터셋과 학습 코드가 필요합니다.

# 감사합니다~