JetBot을 활용한 인공지능 로봇 입문

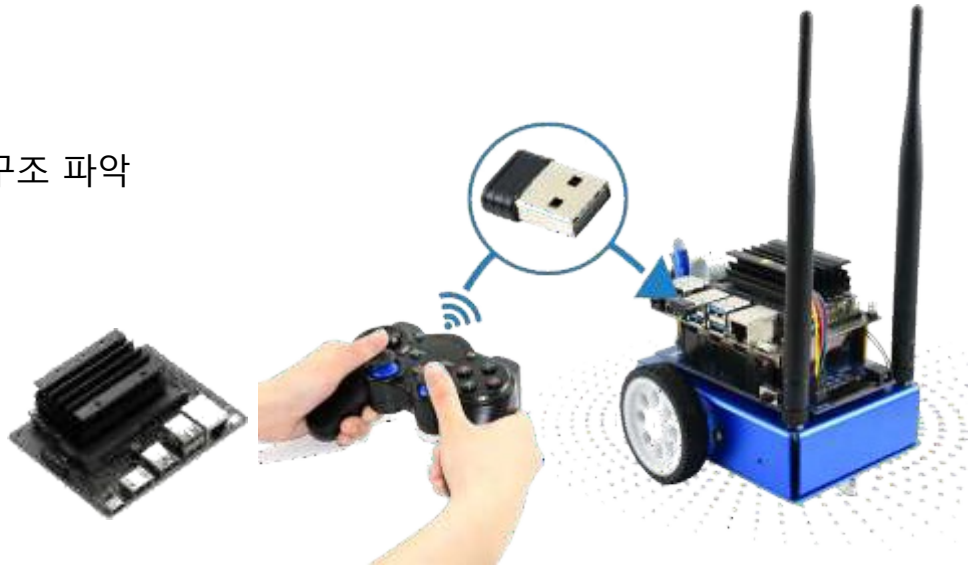
임경태

본 자료는 AIFrenz의 로봇랫 고우영 (국보연) 연구원의 자료를 바탕으로 만들었음

강의자료 https://url.kr/i2r1mj
조립영상 https://bit.ly/2VsDAOl
Jetbot github: https://github.com/NVIDIA-AI-IOT/jetbot

1. JetBot을 활용한 인공지능 로봇 입문

- **젯봇 조립**
  - 실제 젯봇을 조립하면서, 모바일 로봇의 구조 파악
- **베이직 오퍼레이션**
  - 기본적인 동작과 원격으로 젯봇을 조종
  - Jetson Nano 설치 및 사용법
- **충돌 회피(Collision avoidance)**
  - JecBot 카메라로 이미지 수집
  - 이미지 분류 모델 구축, 학습, 적용
  - 젯봇이 충돌하지 않고 주행하는 알고리즘 실습
- **물체 따라가기(Object following)**
  - Object detection을 구현해서 따라가는 젯봇 주행 알고리즘을 실습
- **길 따라가기(Line tracking)**
  - 딥러닝으로 ResNet-18 한번도 주행해 보지 않은 길을 지도 없이 주행하는 알고리즘을 실습

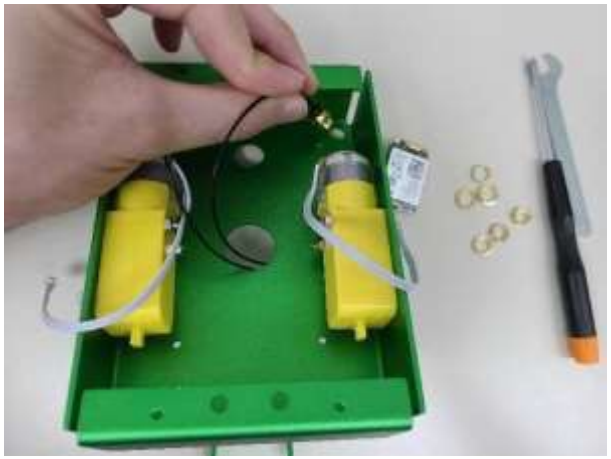


Jetbot 부품

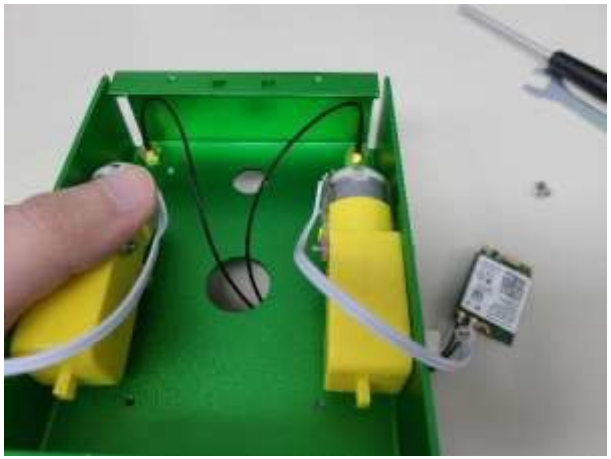
- Jetson Nano
- Jetbot kit
  - 구매링크 : https://www.waveshare.com/JetBot-AI-Kit.ht m
- Micro SD card(32G~)
- PC&WIFI

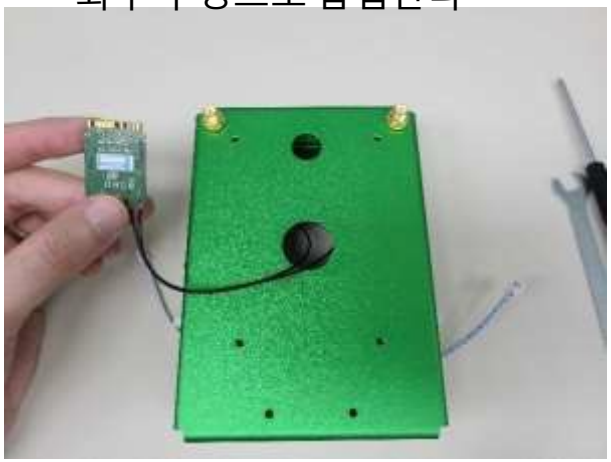
Jetbot 조립 1

DC 모터를 본체 좌우에 볼트와 너트로 조인다



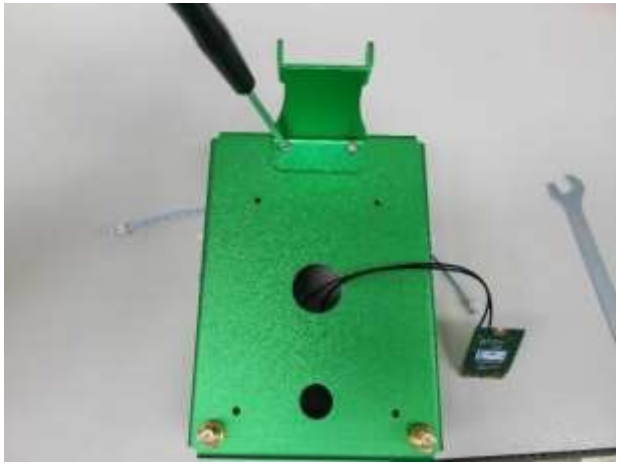
안테나를 푼 후 위에서 아래로 가운데 구멍을 통과한 후 좌우 구멍으로 삽입한다



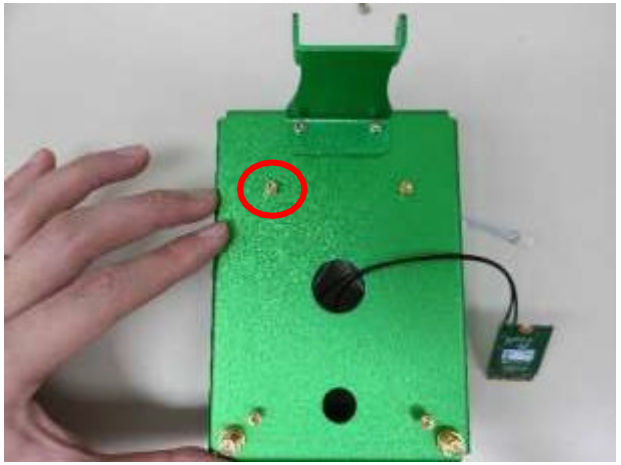


위에서 너트를 조여 고정한다

Jetbot 조립 2

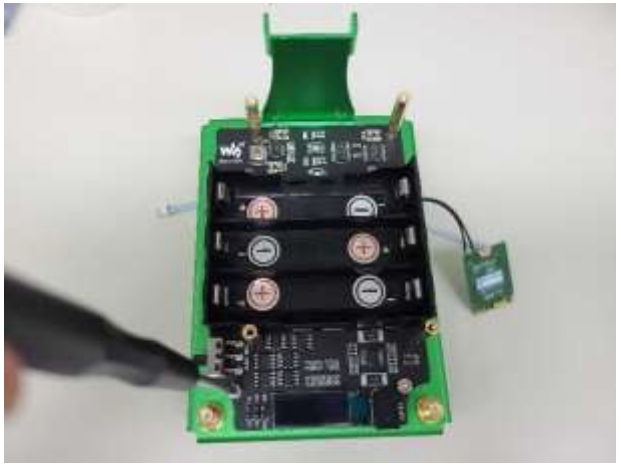


이미지 카메라 고정대를 나사로 조인다



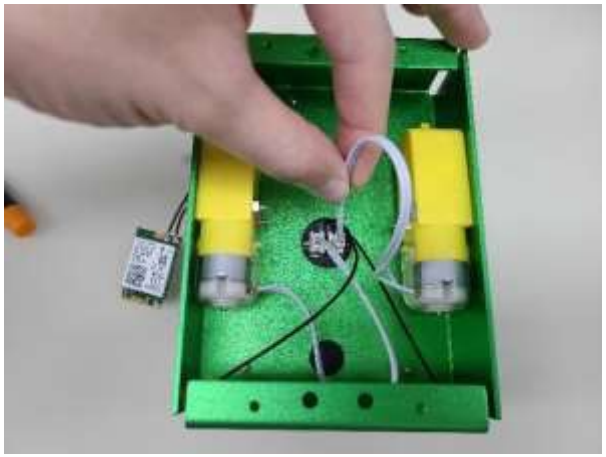
젯슨 나노를 올리기 위해 금색 고정대를 설치한다

배터리보드에 금색 긴 고정대를 설치한다



배터리보드를 본체에 고정한다

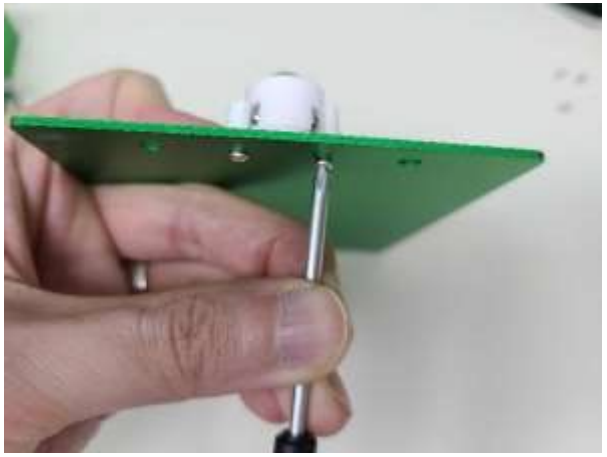
Jetbot 조립 3



본체 아래에서 모터 커넥터를 배터리 보드에 연결한다

배터리 보드에 충전지(18650) 3개를 **+/-를 주의하여 장착한다(꼭! 확인! 위험!)**

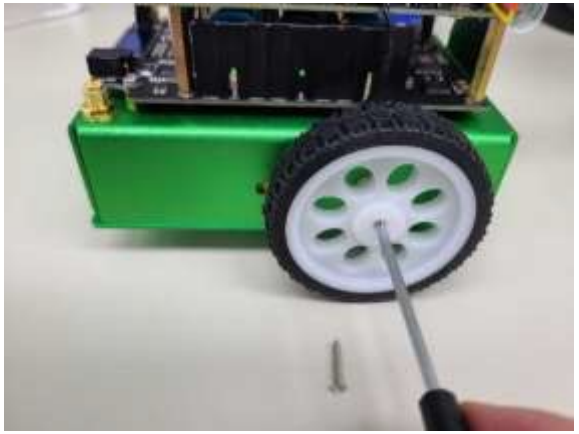
2개의 캐스터 비퀴를 분해한다



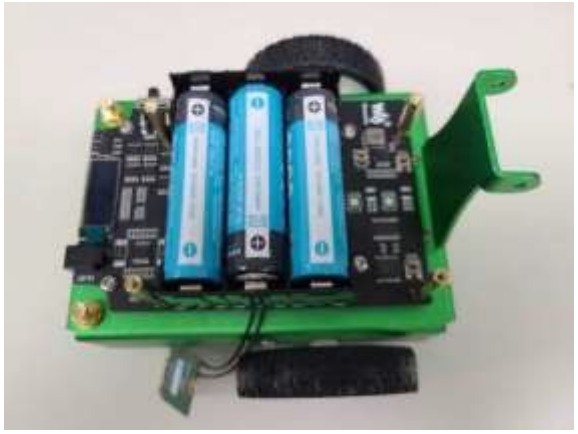
나사를 이용해 밑판에 고정한다

Jetbot 조립 4

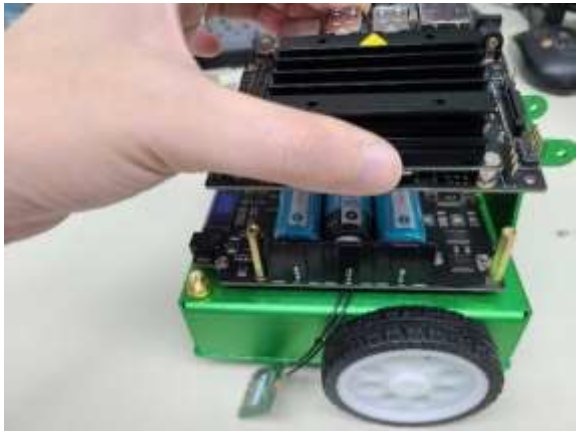
밑판과 본체를 조립한다



긴 나사로 바퀴를 DC모터와 조립한다



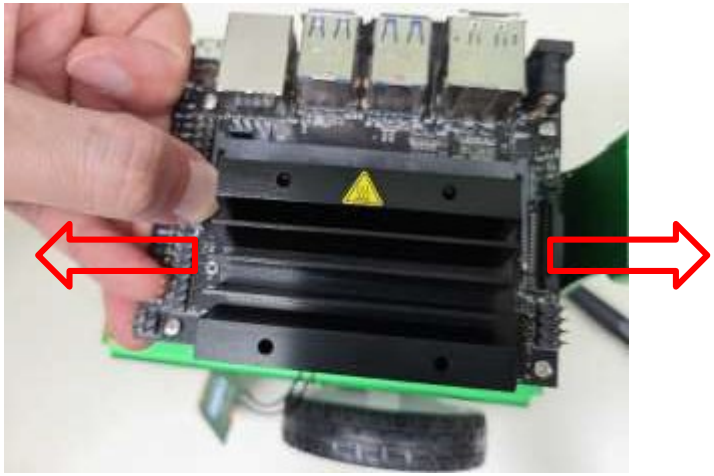
그럼 이와같이 생긴다 오오



젯슨나노를 배터리 보드위에 설치하기 위해

4개의 나사를 조인다

Jetbot 조립 5



젯슨나노에서 방열판을 분리하기 위해 좌우로 벌려서 방열판을 빼낸다

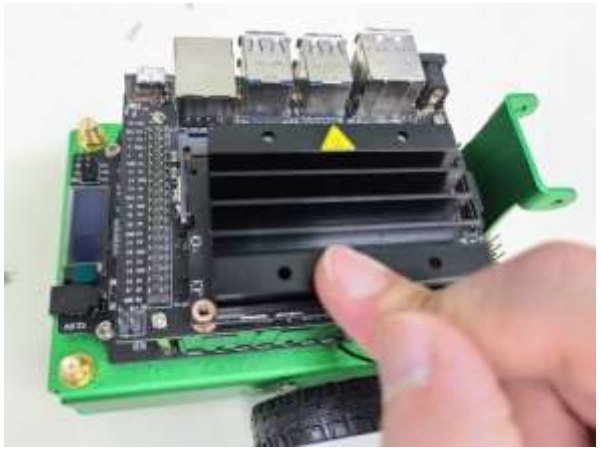
그럼 이런 상황인데

와이파이 모듈을 끼우고

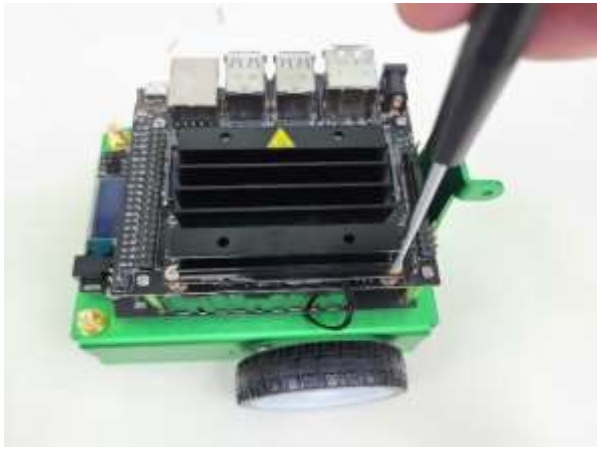
나사로 고정한다

Jetbot 조립 6

다시 방열판을 쑥 밀어넣고



손으로 꼭 누르면 딸깍 고정된다

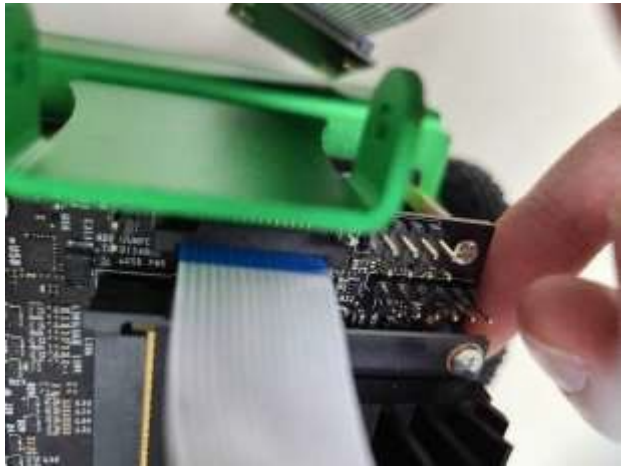


나사로 방열판을 고정한다

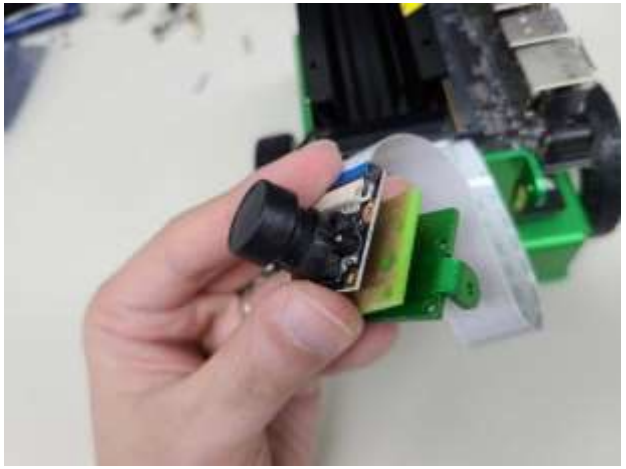
Jetbot 조립 7

웹캠을 장착하기 위해 위와같이 준비하고

웹캠 커넥터를 젯슨 나노에 쑥 하고 낀다(방향 주의)



방향 주의



카메라를 위와같이 2개의 보호판과 알루미늄 판에 고정하기 위해

Jetbot 조립 8

검은 나사와 너트를 이용해서 고정한다

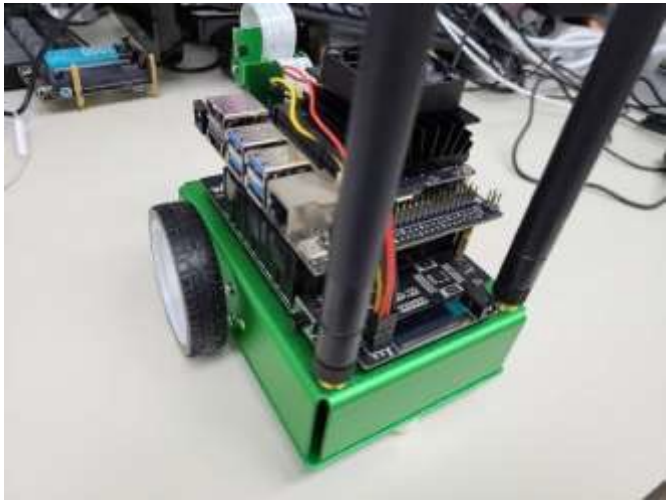
커넥터를 잘 말아넣고, 카메라의 각도를 조정 후 나사로 고정한다

Jetbot 조립 9

온도를 낮추기 위해 팬을 고정한 후 커넥터를 끼운다

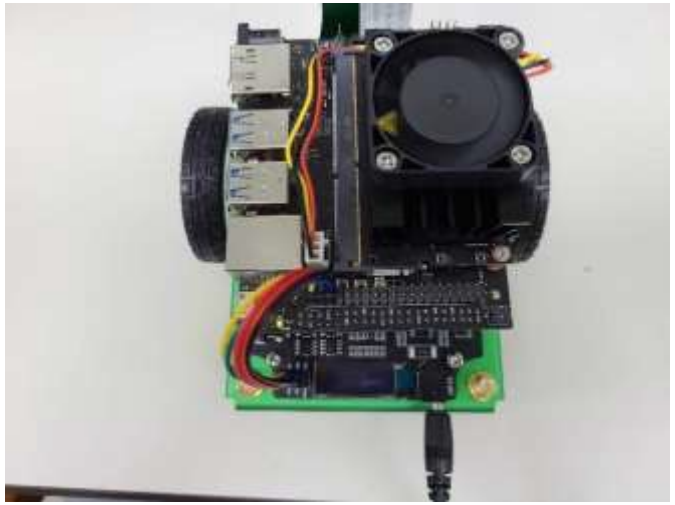
배터리보드와 젯슨나노를 위와같이 연결해준다

미리 준비한 SD카드를 젯슨나노 옆부분에 삽입한다

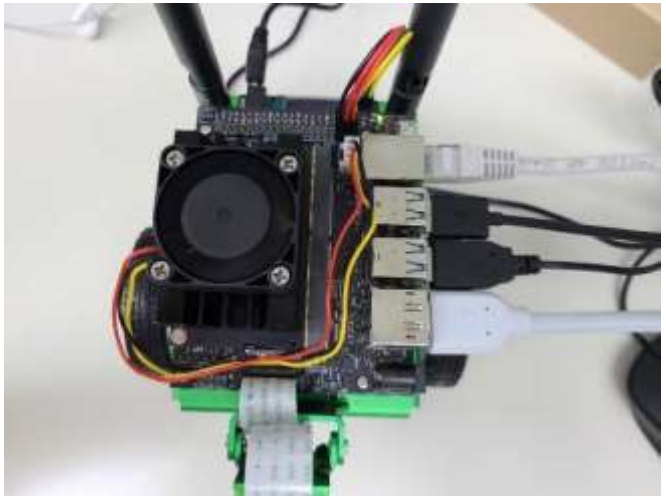


안테나를 꽂아준나

Jetbot 조립 10



배터리보드에 전원을 연결한다



모니터, 키보드, 마우스, (랜선)을 연결해 동작을 확인한다

Jetbot Software Setup

- 1) 이미지 버너 설치
- 2) Jetbot 이미지 다운로드
- 3) SD카드에 Jetbot 이미지 굽기

Jetbot Nano spec

Comparison between Jetson Nano 2GB and Jetson Nano 4GB

	Jetson Nano Developer Kit B01	Jetson Nano Developer Kit 2GB
GPU	128-core Maxwell	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz	Quad-core ARM A57 @ 1.43 GHz
Memory	4GB 64-bit LPDDR4 25.6 GB/s	2GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (not included)	microSD (card not included)
Video Encode	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)
Camera	2x MIPI CSI-2 DPHY lanes	1x MIPI CSI-2 D-PHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E	Gigabit Ethernet, M.2 Key E, Optional 802.11ac Wireless Adapter
Display	HDMI and display port	HDMI
USB	4x USB 3.0, 1×USB 2.0 Micro-B	1x USB 3.0, 2x USB 2.0 , 1×USB 2.0 Micro-B
Others	GPIO, I2C, I²S, SPI, UART	GPIO, I2C, I²S, SPI, UART
Mechanical	100 mm x 80 mm x 29 mm	100 mm x 80 mm x 29 mm

구매 링크

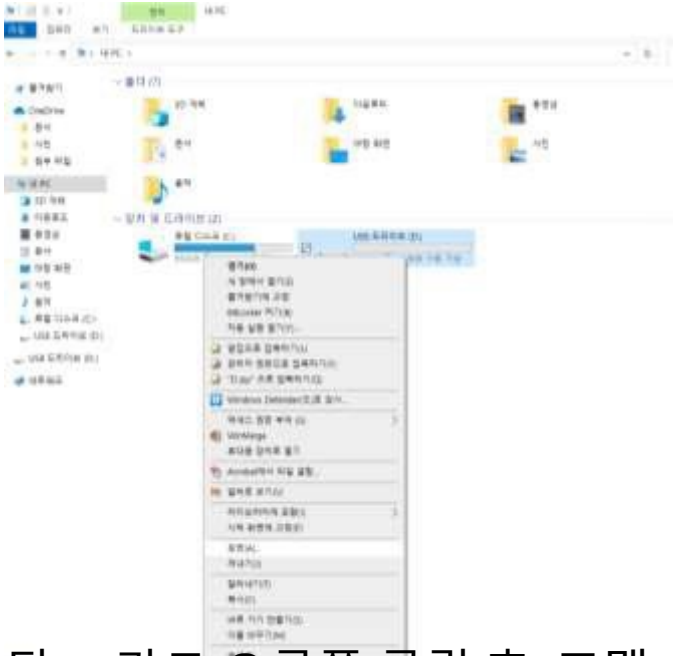
JetsonNano(4G) : 13만
https://www.devicemart.co.kr/goods/view?no=12513656&gclid=EAIaIQobChMIxsTgxpe77wIVQauWCh1W7w1KEAQYASABEgLrPPD_BwE

JetsonNano(2G) : 7만
https://www.devicemart.co.kr/goods/1vie7w?/no5=183023833&gclid=EAIaIQobChMIxsTgxpe77wIVQauWCh1W7w1KEAQYAiABEgKzU_D_BwE

Jetbot Software Setup 1

SD카드 포장을 뜯는다

SD카드를 리더기에 삽입 후 노트북에 연결



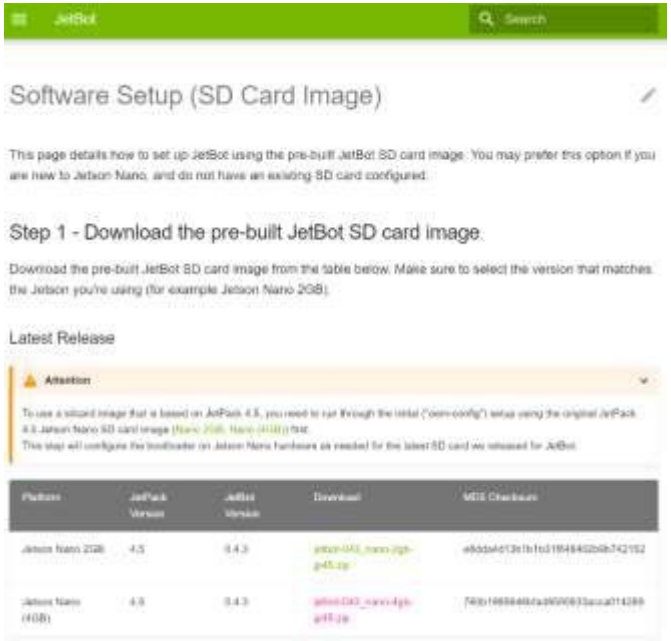
인식된 SD카드 오른쪽 클릭 후 '포맷' 선택

기본 셋팅으로 포맷 시작

Jetbot Software Setup 2

- 1) 이미지 버너 설치
  - https://www.balena.io/etcher/ 에서 다운 후 설치

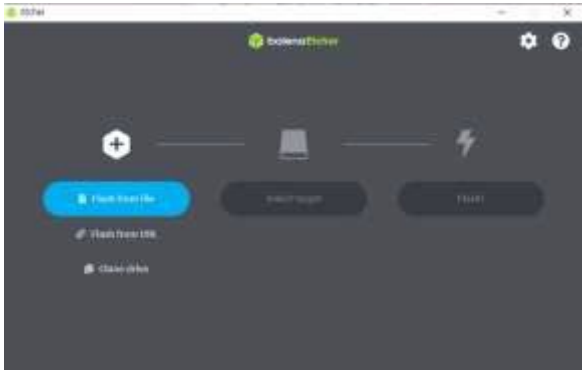
- 2) Jetbot image 다운로드
  - http://jetbot.org/v0.4.3/software_setup/sd_card.html
  - **for JetsonNano(4GB), 13G**



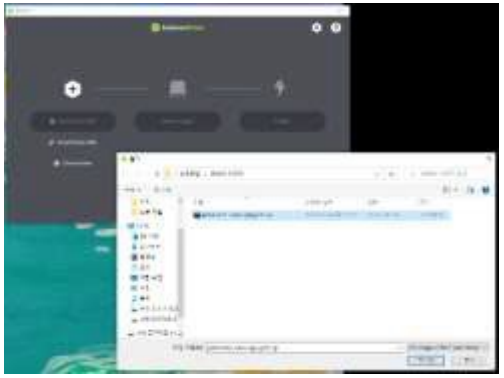
JetsonNano 버전에 맞게 선택
Memory 2G : 2G
Memory 4G : 4G

Jetbot Software Setup 3

- **3) SD카드에 Jetbot 이미지 굽기**

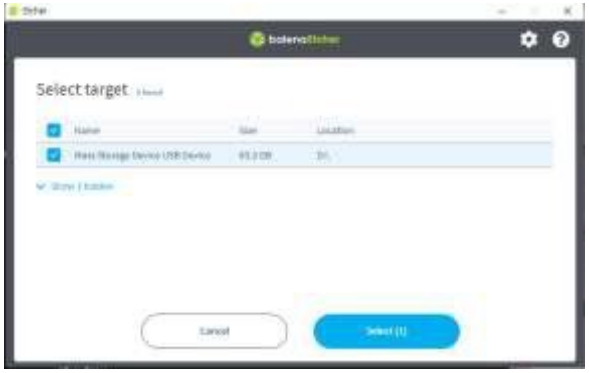


1)에서 설치한
etcher 실행



2)에서 다운로드한
Jetbot 이미지 선택

Select target 클릭

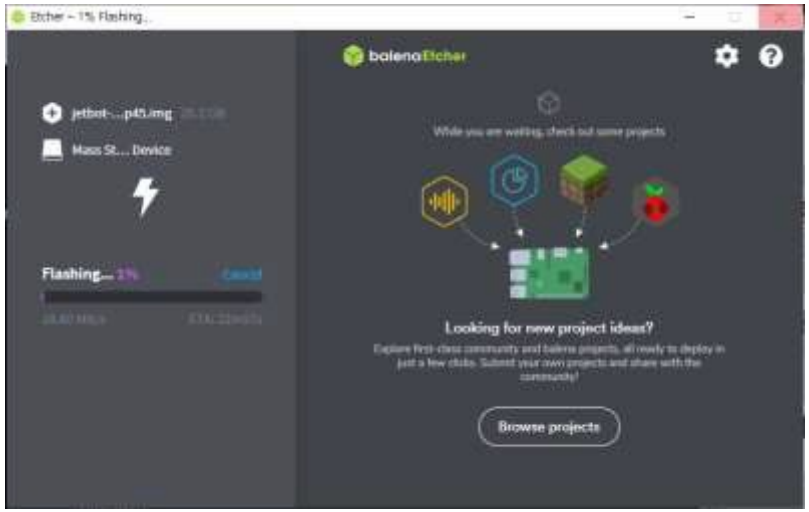


준비한 SD카드 선택(32G이상)

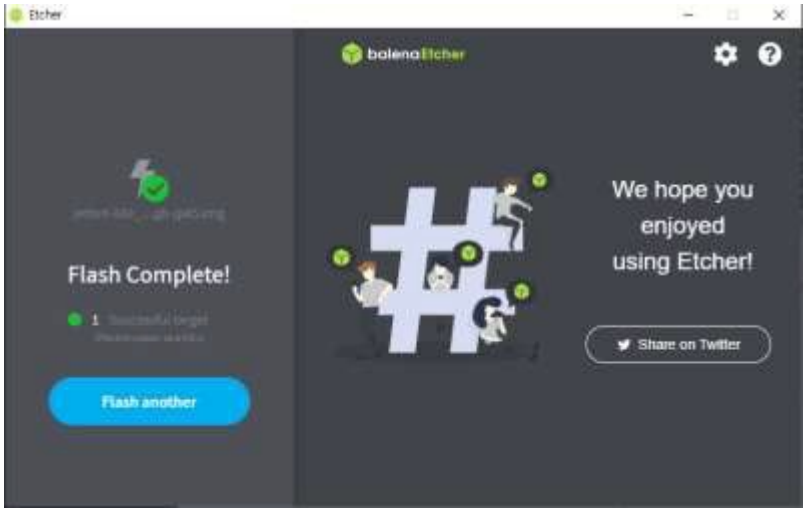
Flash 클릭!

Jetbot Software Setup 4

- **3) SD카드에 Jetbot 이미지 굽기**
  - 25분 정도 소요, valid 체크까지 하면 50분 걸림...valid는 하지 말자



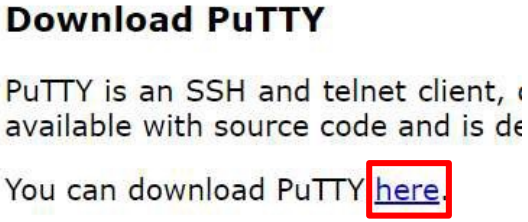
진행중



완성!

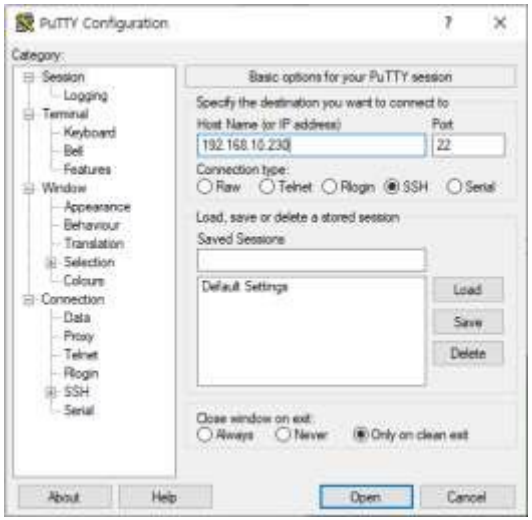
이미지 굽는 동안 원격접속 공부 1

- **Putty 설치(윈도우)**
  - https://www.putty.org/



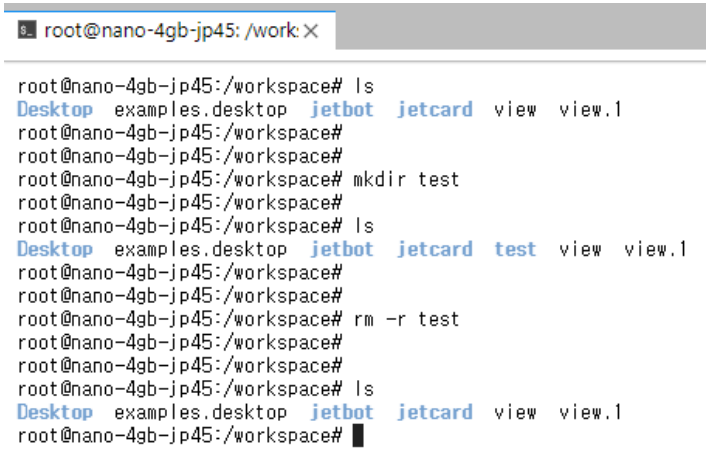
- next-> next-> next-> next->Finish

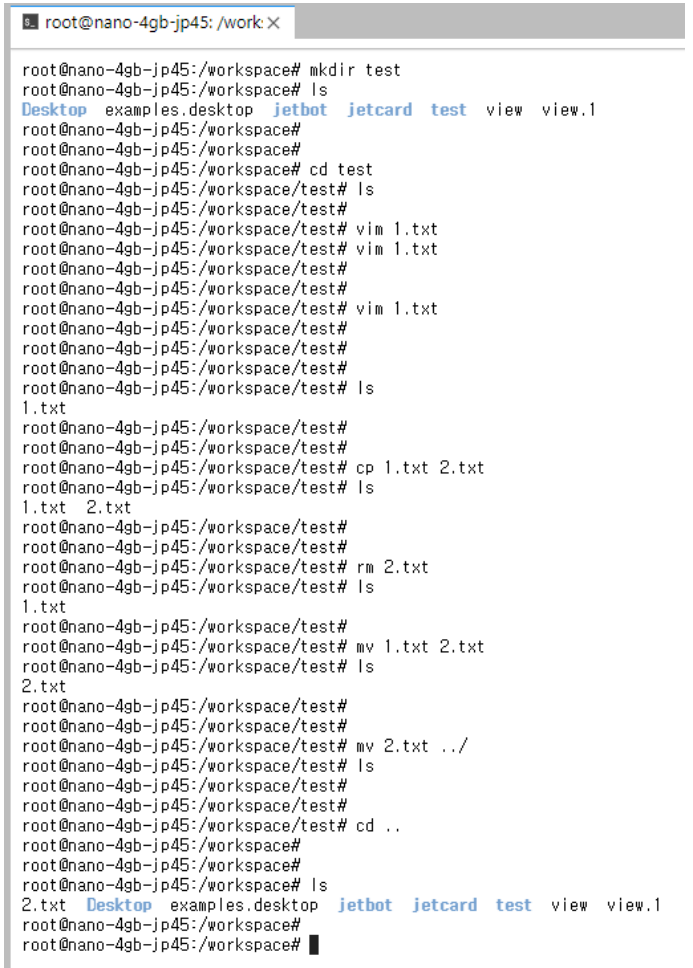
ip 입력 후 Open, id/pw 입력(jetbot/ jetbot)



Terminal 공부

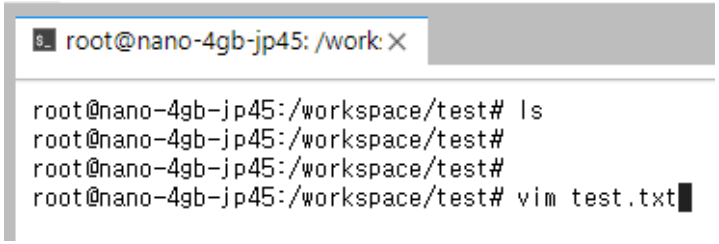
- **Terminal command**
  - ls : 현 위치의 파일 보기(윈도우에선 dir)
  - clear : 화면 깨끗이
  - mkdir
    - 폴더 만들기
    - mkdir <folder>
  - rm
    - 삭제(파일/폴더)
    - rm 1.txt # 1.txt 파일 삭제
    - rm –r <folder> # 폴더 삭제
  - cd : 이동
    - >> cd <folder>
    - >> cd ..
    - >> cd <folder1>/<folder2>
  - cp : 복사
    - cp 1.txt 2.txt # 파일 복사
    - cp –r <folder1> <folder> # 폴더 복사
  - mv : 이동 or 이름바꾸기
    - mv 1.txt 2.txt # 파일 이름 바꾸기
    - mv 1.txt ../ # 1.txt 파일을 상위 폴더로 이동
  - sudo shutdown –h now
    - PW : jetbot
    - 당장 OS 종료
  - ctrl+c : 취소





vim 공부 – 텍스트 파일 만들기

- **vim command**
  - **txt 파일 만들기**



vim 으로 파일 만들기

>> vim test.txt

빈 파일이 만들어 진다

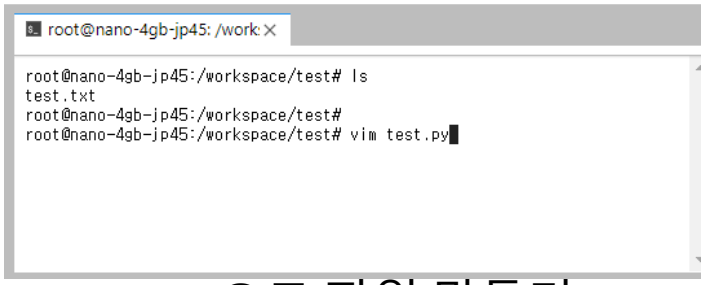
‘i’ 를 누르면 아래 –INSERT-가 뜨고 키보드로 입력할 수 있다

‘welcome to the ai robot lab’ 을 입력 후 저장하고 나가보자

ESC -> : -> wq

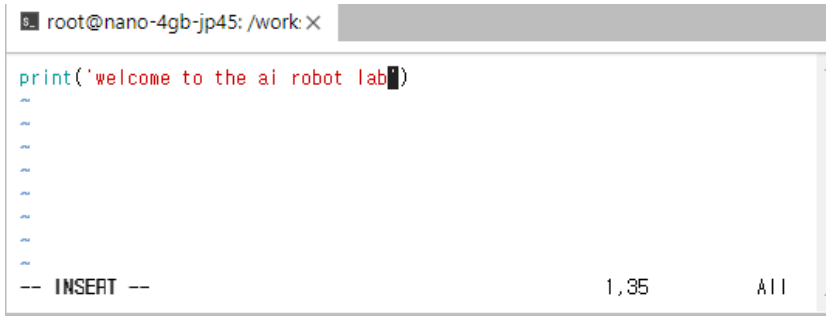
vim 공부 – python 파일 만들기

- **vim command**
  - **python 파일 만들기**



vim 으로 파일 만들기

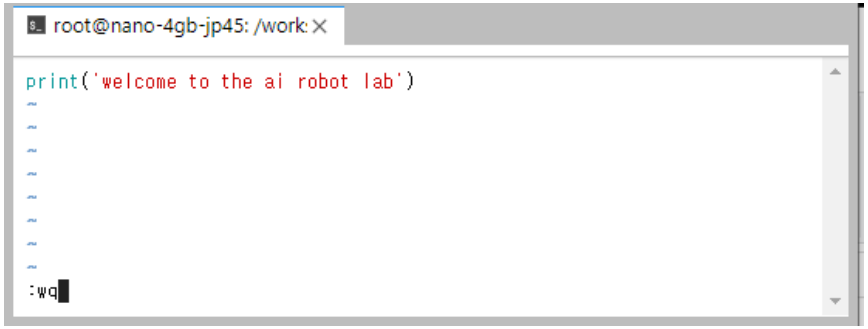
>> vim test.py



빈 파일이 만들어 진다

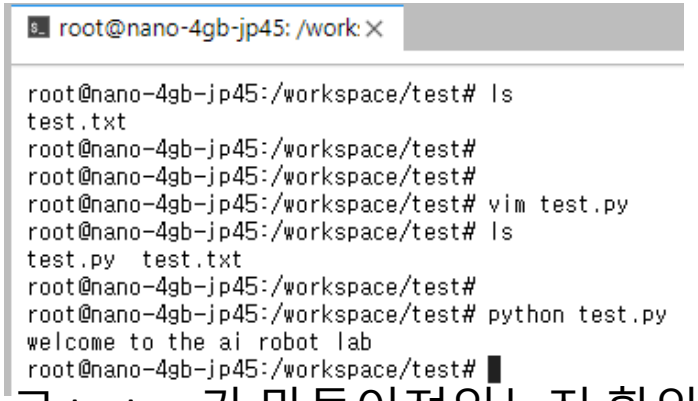
'i' 를 누르면 아래 –INSERT-가 뜨고 키보드로 코드 입력

>> print('welcome to the ai robot lab')



'welcome to the ai robot lab' 을 입력 후 저장하고 나가보자

ESC -> : -> wq



ls로 test.py가 만들어져있는지 확인

python test.py 명령어로 python 코드 실행

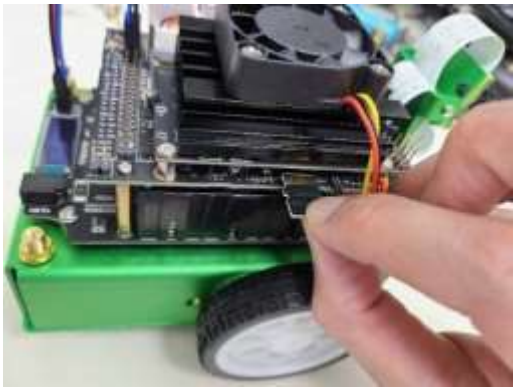
welcome to the ai robot lab 보이면 성공

Jupyter lab 공부

- **Jupyter lab**
  - **>> pip install jupyterlab**
  - **직접 실습**

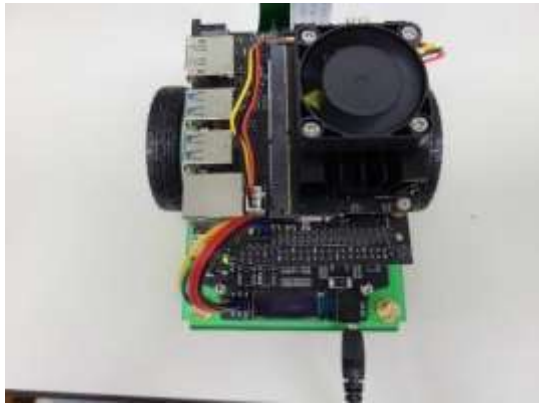
Jetbot Software Setup 5

- **4) Jetbot 부팅**
  - 1) JetsonNano에 SD카드 삽입
  - 2) 모니터, 키보드, 마우스 연결
  - 3) 배터리보드에 전원 연결
  - 4) 배터리보드 스위치 On!
    - fan이 돌고 JetsoNano에 노란불이 들어오면 성공!

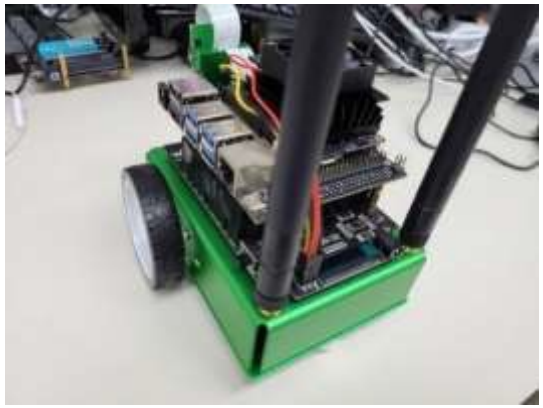


1) 미리 준비한 SD카드를 젯슨나노 옆부분에 삽입한다

2) 모니터, 키보드, 마우스, (or 랜선)을 연결해 동작을 확인한다



3) 배터리 보드에 전원을 연결한다



4) 배터리보드에 전원을 연결한다

Jetbot Software Setup 6

- **5) WIFI 연결**
  - 1) 모니터를 연결하고 부팅
  - 2) 로그인
    - id : jetbot
    - pw : jetbot
  - 3) termina에서 아래 명령어로 WIFI 연결
    - >> sudo nmcli device wifi connect <SSID> password <PASSWORD>
    - >> sudo nmcli device wifi connect aai5G password dlarudxo
    - 이후론 자동으로 연결, piOLED displa에 표시된 IP를 적어두자!!
      - Web에서 접속할 때 필요

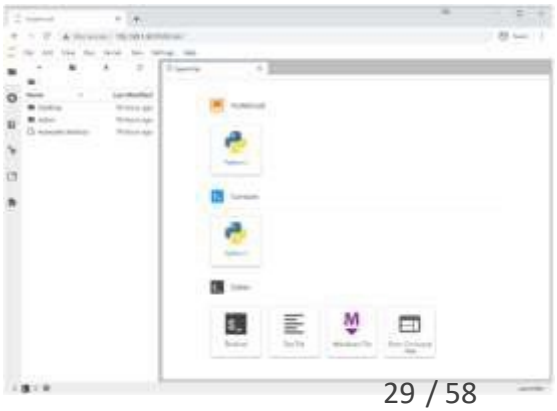
오른쪽 위, wifi 선택 밑 PW 입력

Connection information 클릭

IP주소 메모
노트북과 같은 WIFI를 쓰고있어야 한다

Jetbot Software Setup 7

- **6) 내 노트북 web browser에서 Jetbot 연결**
  - WIFI에 연결하면 모니터 없이 내 컴퓨터에서 접속 가능
  - 1) 우선 Jetbot을 끄자
    - >>sudo shutdown –h now
    - 모니터,키보드,마우스 연결선 제거
  - 2) 배터리보드 OFF 후 다시 ON하여 재부팅(부팅시간동안 대기)
  - 3) piOLED 화면에 IP주소를 체크
  - 4) 내 노트북 인터넷 브라우저를 킨 후 아래 주소 입력
    - http://<jetbot_ip_address>:8888
      - ex) http://192.168.10.xxx:8888
    - 패스워드 입력: jetbot



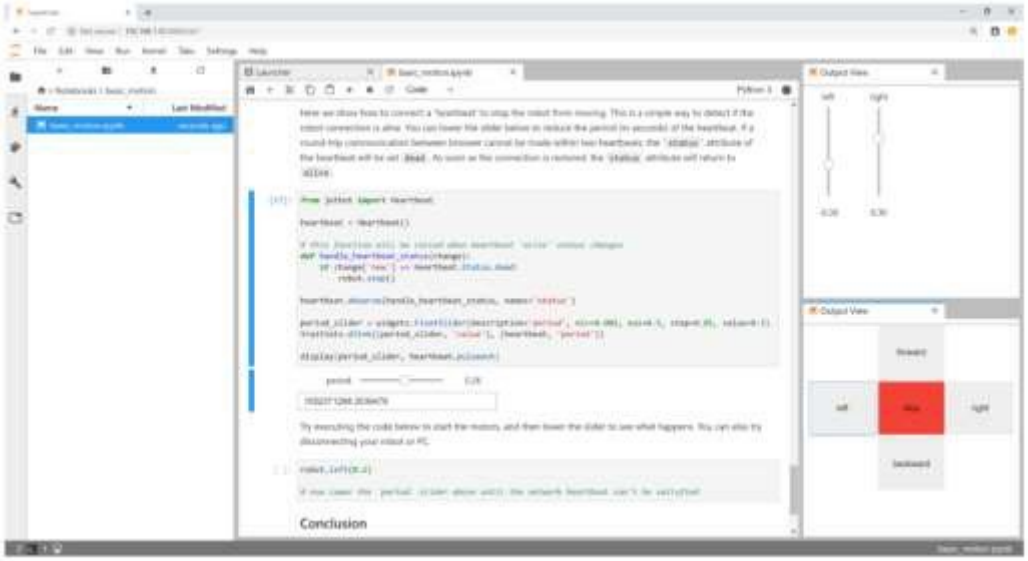
성공화면

고생하셨어요~~

조립 참고 : https://www.youtube.com/watch?v=kq7Jtj5IGiU
Jetbot Software Setup 참고 : http://jetbot.org/v0.4.3/software_setup/sd_card.html

Jetbot Software-Basic Operation

- **Basic Motion 제어 in web**
  - 노트북 웹에서 내 로봇에 접속
    - http://<jetbot_ip_address>:8888
    - pw : jetbot
  - ~/Notebooks/basic_motion/ 경로로 이동
  - basic_motion.ipynb 파일 열기



성공화면

Jetbot Software-Collision Avoidance 1

- **이미지 분류 in web**
  - ~/Notebooks/collision_avoidance/ 경로로 이동
  - 1) 이미지 분류 데이터 수집
    - data_collection.ipynb, 직접 실습하며 설명
  - 2) 이미지 분류 모델 학습
    - train_model_resnet18.ipynb, 직접 실습하며 설명
  - 3) 추론
    - live_demo_resnet18.ipynb, 직접 실습하며 설명
  - 4) 추론 with TensorRT
    - live_demo_resnet18_build_trt.ipynb, 모델 TensorRT로 변환
    - live_demo_resnet18_trt.ipynb, TensorRT로 추론

Jetbot Software-Collision Avoidance 1

Collision Avoidance - Train Model (ResNet18)

Welcome to this host side Jupyter Notebook! This should look familiar if you ran through the notebooks that run on the robot. In this notebook we'll train our image classifier to detect two classes `free` and `blocked`, which we'll use for avoiding collisions. For this, we'll use a popular deep learning library *PyTorch*

```
import torch
import torch.optim as optim
import torch.nn.functional as F
import torchvision
import torchvision.datasets as datasets
import torchvision.models as models
import torchvision.transforms as transforms
```

Upload and extract dataset

Before you start, you should upload the `dataset.zip` file that you created in the `data_collection.ipynb` notebook on the robot.

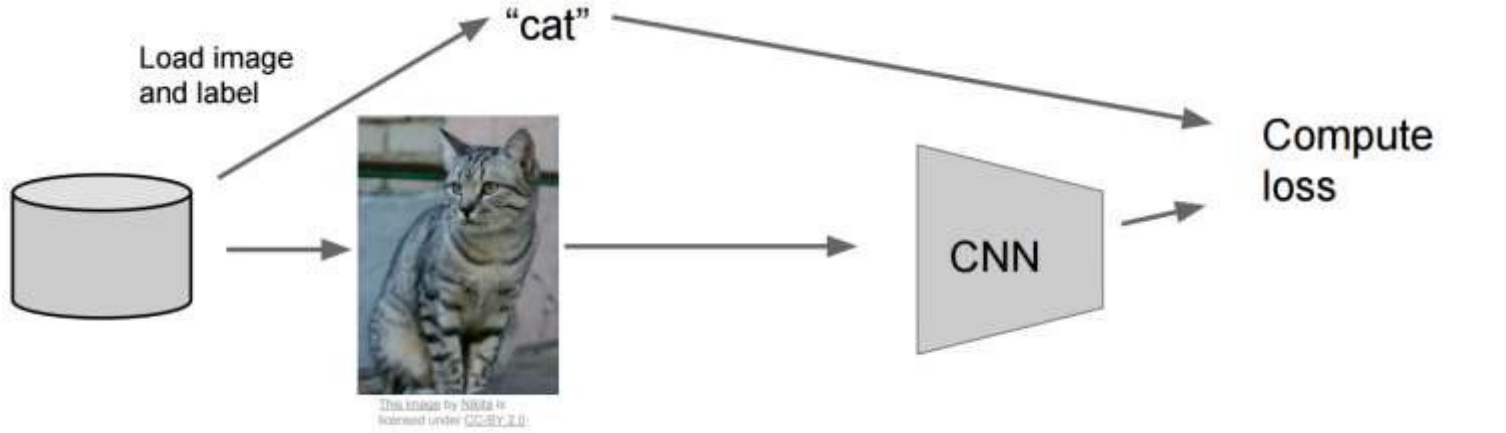
You should then extract this dataset by calling the command below

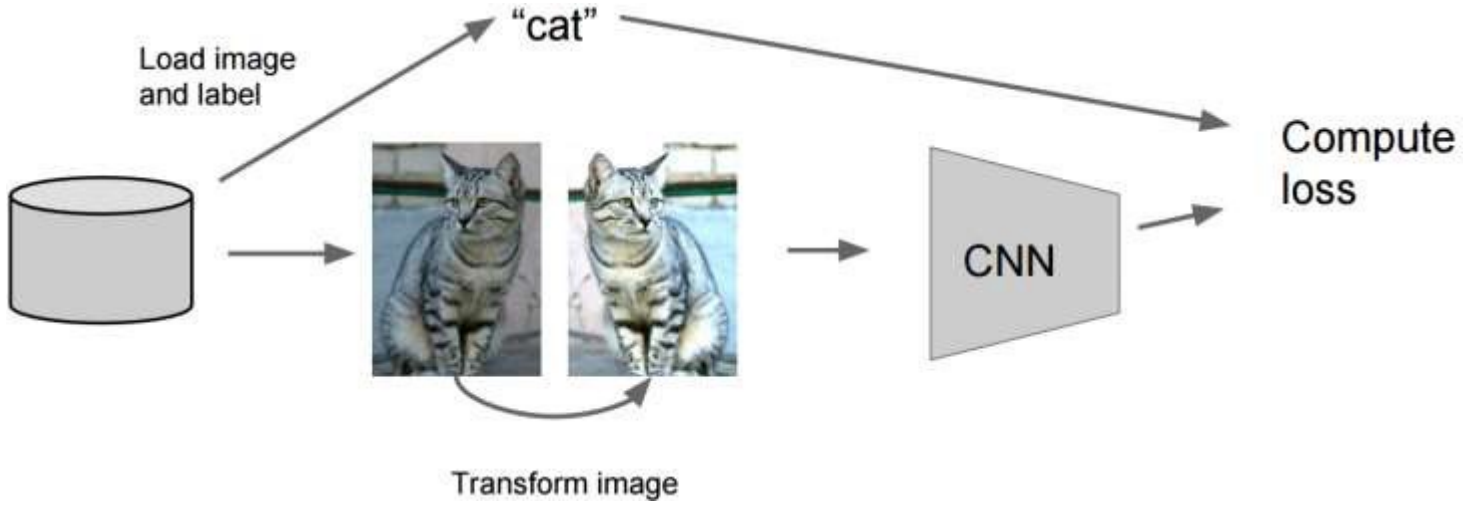
```
!unzip -q dataset.zip
```

You should see a folder named `dataset` appear in the file browser.

이미지 분류모델 학습 -1

- **Convert data & augmentation**





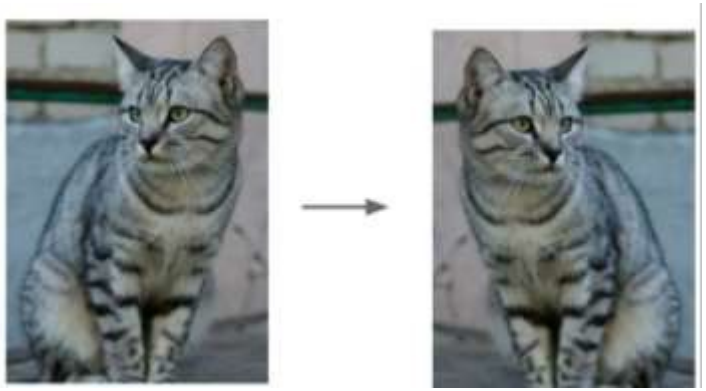
Jetbot Software-Collision Avoidance 1

Create dataset instance

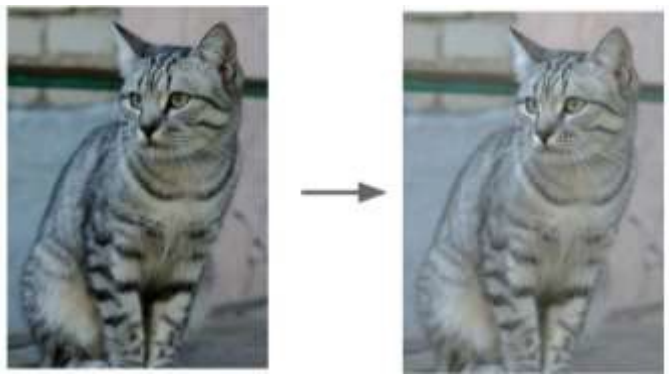
Now we use the ImageFolder dataset class available with the torchvision.datas
attach transforms from the torchvision.transforms package to prepare the data for training.

```
dataset = datasets.ImageFolder(
    'dataset',
    transforms.Compose([
        transforms.ColorJitter(0.1, 0.1, 0.1, 0.1),
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])
)
```

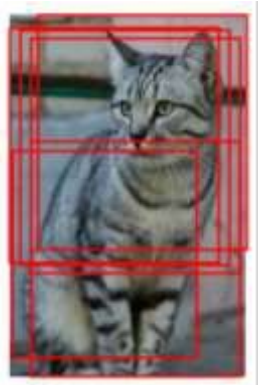
Random crops&scales: 임의로 자르고 크기변경



Flip: 수평/수직 회전



Color Jitter: 밝기 대비 변경

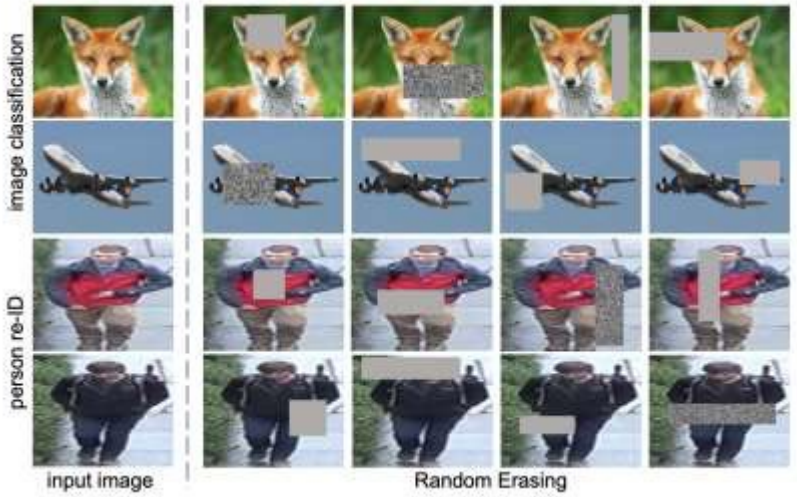


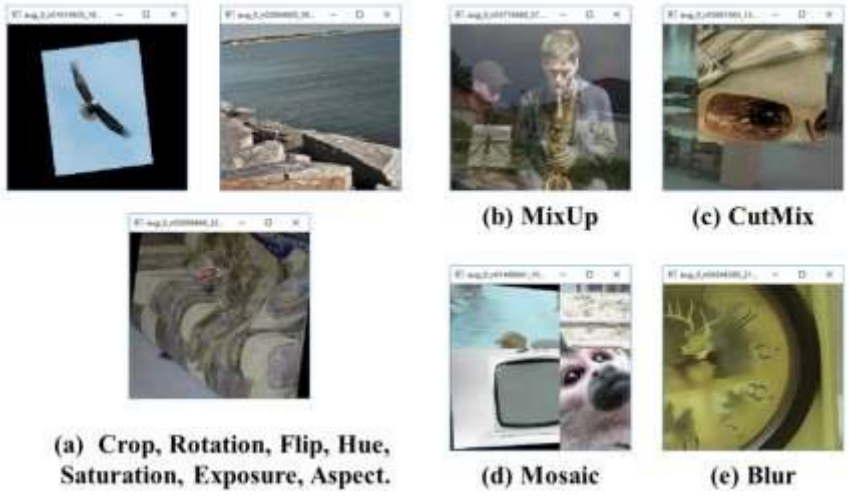
이미지 분류모델 학습 -1

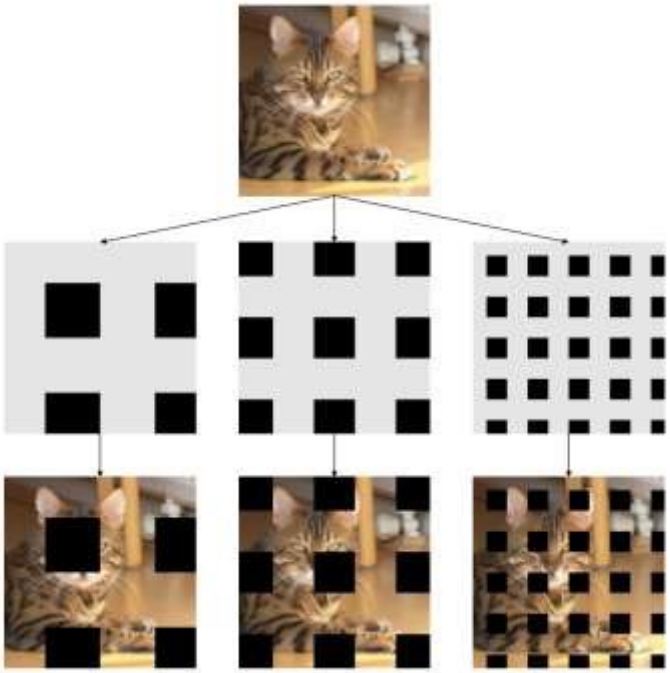
- **Convert data & augmentation**
  - transforms.ToPILImage() - csv 파일로 데이터셋을 받을 경우, PIL image로 바꿔준다.
  - transforms.CenterCrop(size) - 가운데 부분을 size 크기로 자른다.
  - transforms.Grayscale(num_output_channels=1) - grayscale로 변환한다.
  - transforms.RandomAffine(degrees) - 랜덤으로 affine 변형을 한다.
  - transforms.RandomCrop(size) -이미지를 랜덤으로 아무데나 잘라 size 크기로 출력한다.
  - transforms.RandomResizedCrop(size) - 이미지 사이즈를 size로 변경한다
  - transforms.Resize(size) - 이미지 사이즈를 size로 변경한다
  - transforms.RandomRotation(degrees) 이미지를 랜덤으로 degrees 각도로 회전한다.
  - transforms.RandomResizedCrop(size, scale=(0.08, 1.0), ratio=(0.75, 1.3333333333333333)) - 이미지를 랜덤으로 변형한다.
  - transforms.RandomVerticalFlip(p=0.5) - 이미지를 랜덤으로 수직으로 뒤집는다. p=0이면 뒤집지 않는다.
  - transforms.RandomHorizontalFlip(p=0.5) - 이미지를 랜덤으로 수평으로 뒤집는다.
  - transforms.ToTensor() - 이미지 데이터를 tensor로 바꿔준다.
  - transforms.Normalize(mean, std, inplace=False) - 이미지를 정규화한다.

이미지 분류모델 학습 -2

- 다양한 Augmentation 방법
  - Random erasing, Grid mask
  - Crop, Rotation, Flip, ...
  - MixUP, CutMix, Mosaic, Blur







Jetbot Software-Collision Avoidance 1

Split dataset into train and test sets

Next, we split the dataset into *training* and *test* sets. The test set will be used to verify the accuracy of the model we train.

```
train_dataset, test_dataset = torch.utils.data.random_split(dataset, [len(dataset) - 5
0, 50])
```

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

Jetbot Software-Collision Avoidance 1

Create data loaders to load data in batches

We'll create two `DataLoader` instances, which provide utilities for shuffling data, producing *batches* of images, and loading the samples in parallel with multiple workers.

```
train_loader = torch.utils.data.DataLoader(
    train_dataset,
    batch_size=8,
    shuffle=True,
    num_workers=0,
)

test_loader = torch.utils.data.DataLoader(
    test_dataset,
    batch_size=8,
    shuffle=True,
    num_workers=0,
)
```

Jetbot Software-Collision Avoidance 1

Define the neural network ¶

Now, we define the neural network we'll be training. The *torchvision* package provides a collection of pre-trained models that we can use.

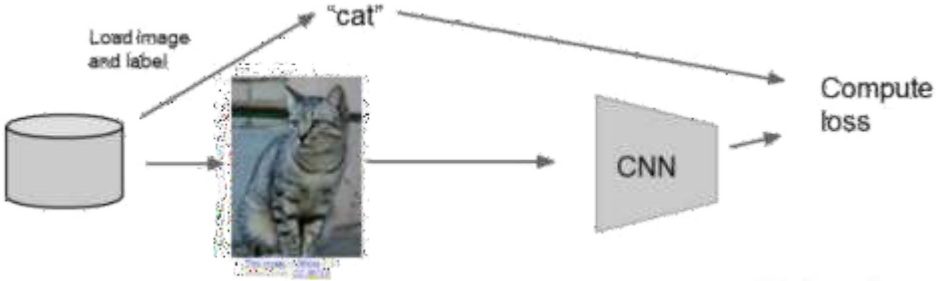
In a process called *transfer learning*, we can repurpose a pre-trained model (trained on millions of images) for a new task that has possibly much less data available.

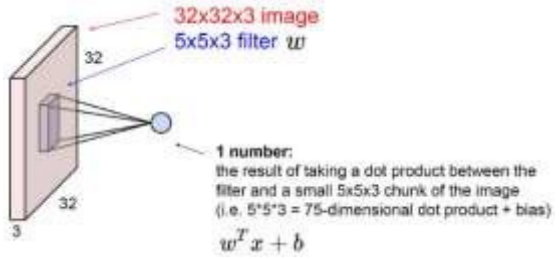
Important features that were learned in the original training of the pre-trained model are re-usable for the new task. We'll use the `resnet18` model.

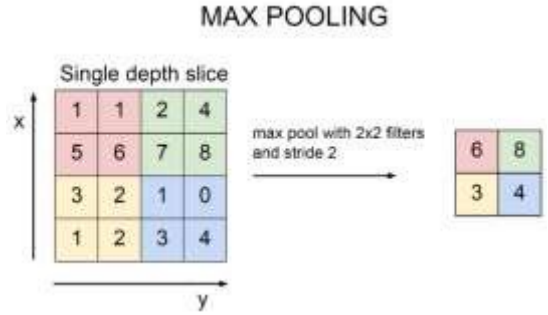
```
model = models.resnet18(pretrained=True)
```

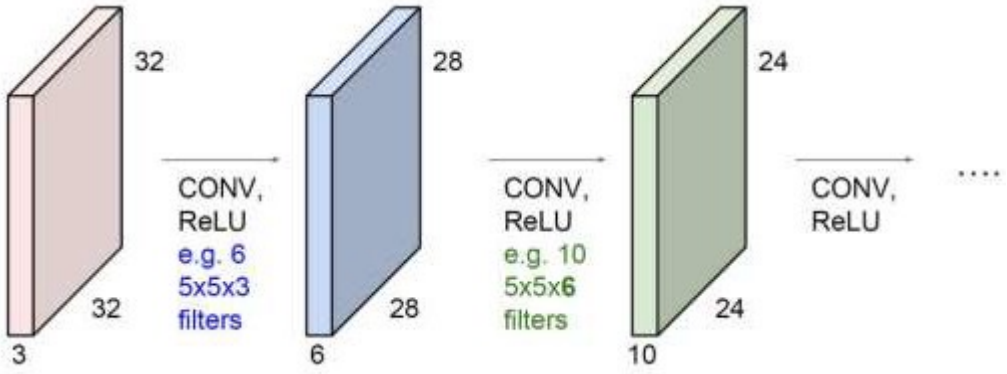
이미지 분류모델 학습 -3

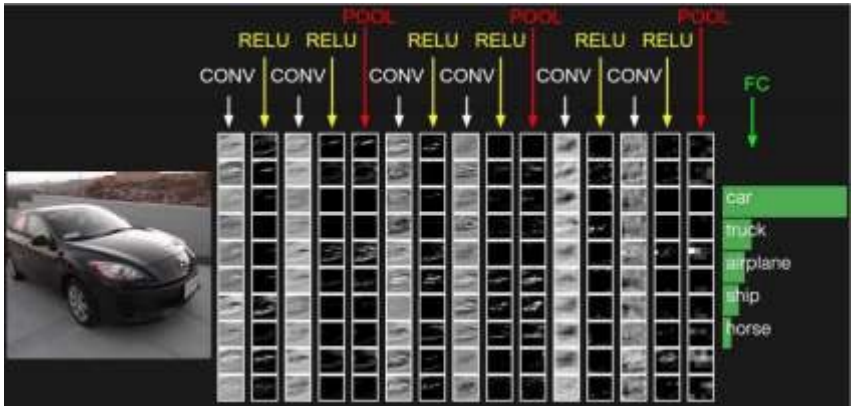
- 이미지 분류모델 – CNN





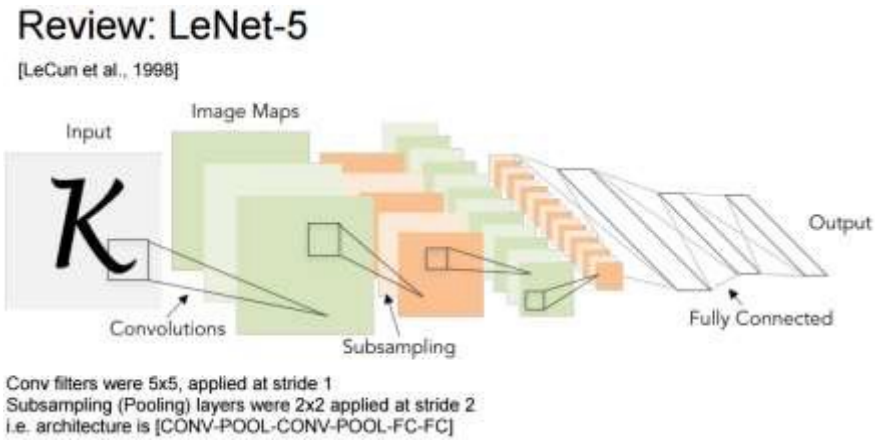


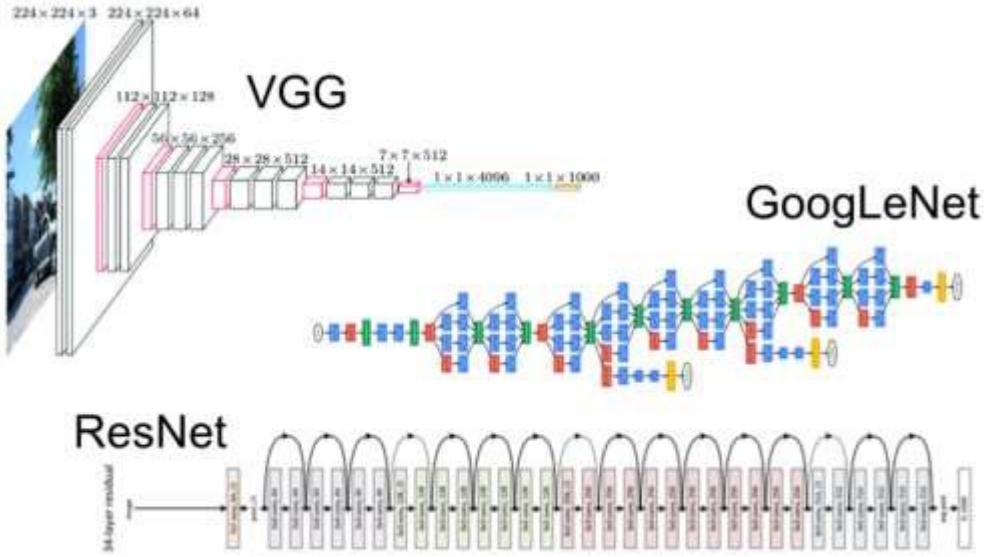


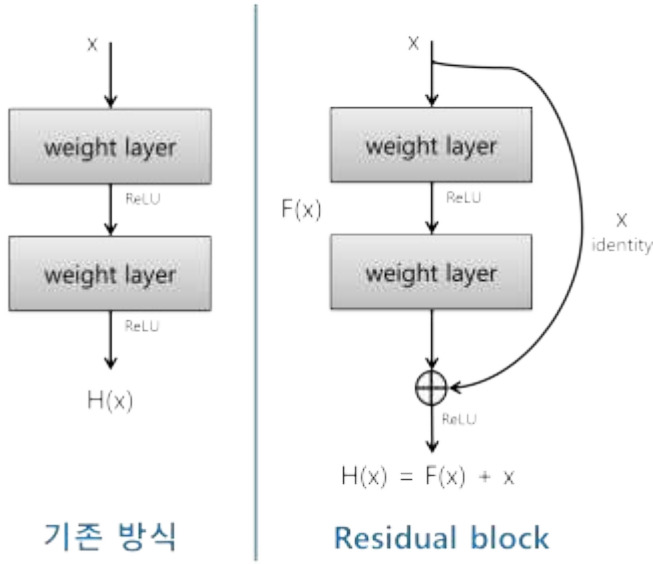


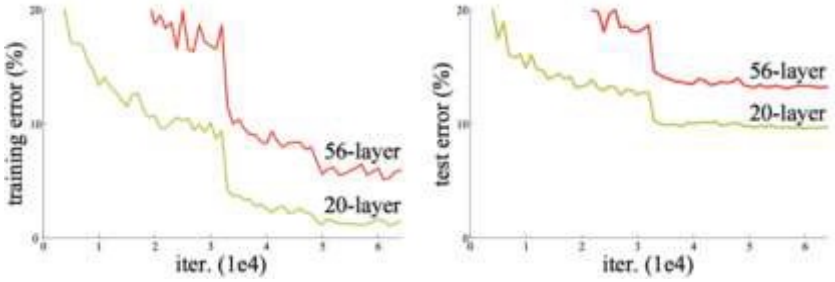
이미지 분류모델 학습 -3

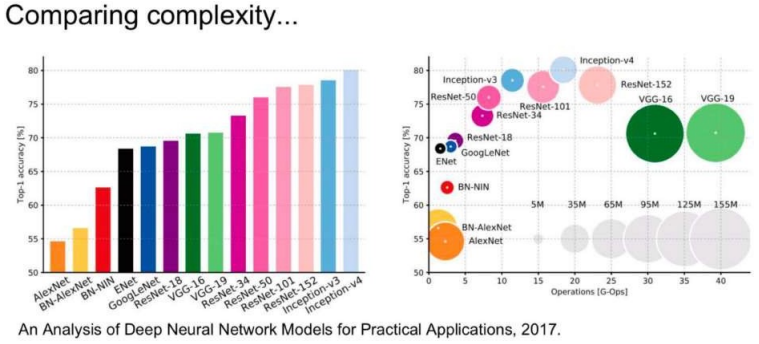
- 이미지 분류모델 – Residual Network







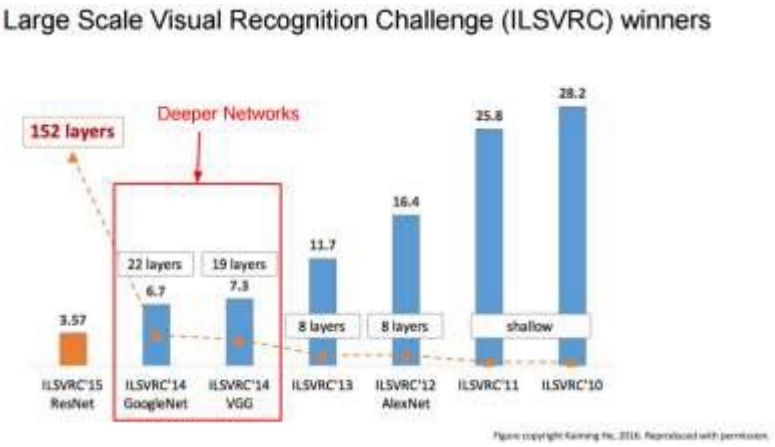




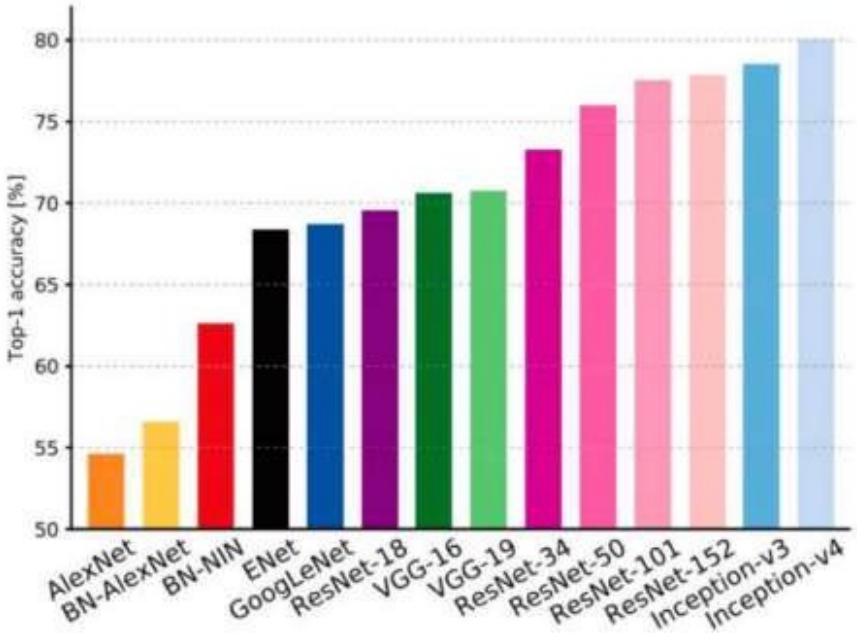
https://oi.readthedocs.io/en/latest/computer_vision/cnn/densenet.html
https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/
https://glassboxmedicine.com/2020/12/08/using-predefined-and-pretrained-cnns-in-pytorch-tutorial-with-code/

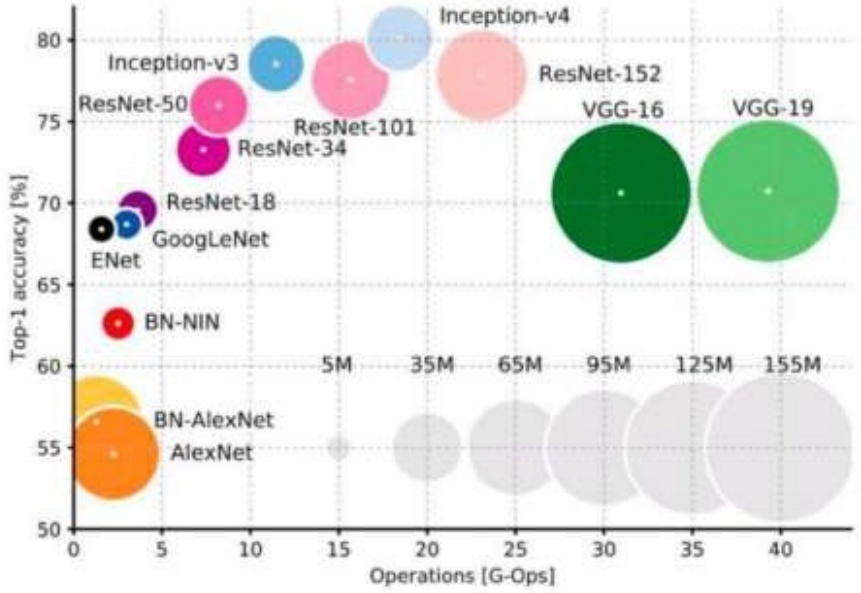
이미지 분류모델 학습 -3

- 이미지 분류모델 – Residual Network



Comparing complexity...



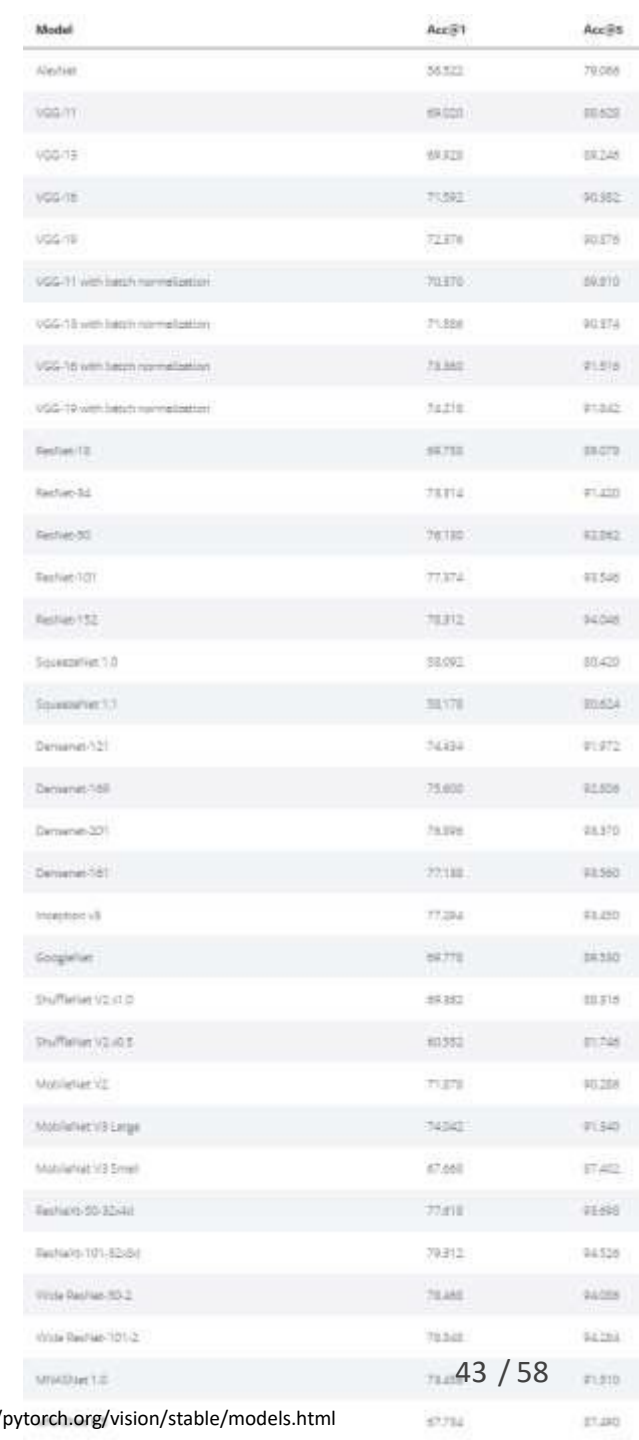


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

https://oi.readthedocs.io/en/latest/computer_vision/cnn/densenet.html
https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/
https://glassboxmedicine.com/2020/12/08/using-predefined-and-pretrained-cnns-in-pytorch-tutorial-with-code/

이미지 분류모델 학습 -4

- **다양한 이미지 분류모델**
  - AlexNet
  - VGG
  - ResNet
  - SqueezeNet
  - DenseNet
  - Inception v3
  - GoogLeNet
  - ShuffleNet v2
  - MobileNetV2
  - MobileNetV3
  - ResNeXt
  - Wide ResNet
  - MNASNet



https://pytorch.org/vision/stable/models.html

이미지 분류모델 학습 -5

- 다양한 이미지 분류모델

```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeezenet1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
googlenet = models.googlenet(pretrained=True)
shufflenet = models.shufflenet_v2_x1_0(pretrained=True)
mobilenet_v2 = models.mobilenet_v2(pretrained=True)
mobilenet_v3_large = models.mobilenet_v3_large(pretrained=True)
mobilenet_v3_small = models.mobilenet_v3_small(pretrained=True)
resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
mnasnet = models.mnasnet1_0(pretrained=True)
```

https://pytorch.org/vision/stable/models.html

이미지 분류모델 학습 -5

- 다양한 이미지 분류모델

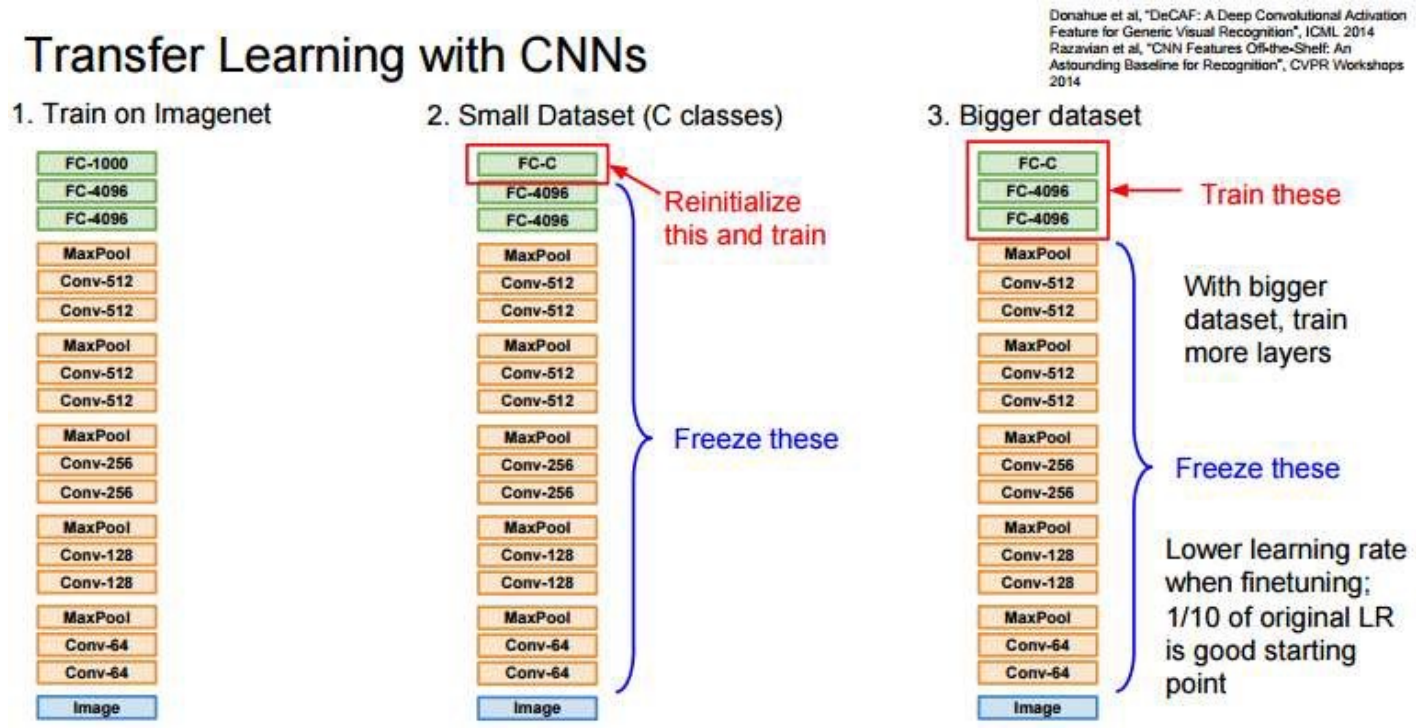
```
1  import torchvision.models as models
2  resnet18 = models.resnet18(pretrained=True)
3  alexnet
4  squeeze
5  vgg16 =
6  densene
7  incepti
8  googlen
9  shuffle                                              rue)
10 mobilen
11 mobilen                                              rained=True)
12 mobilenet_v3_small = models.mobilenet_v3_small(pretrained=True)
13 resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
14 wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
15 mnasnet = models.mnasnet1_0(pretrained=True)
```

Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

https://pytorch.org/vision/stable/models.html

이미지 분류모델 학습 -5

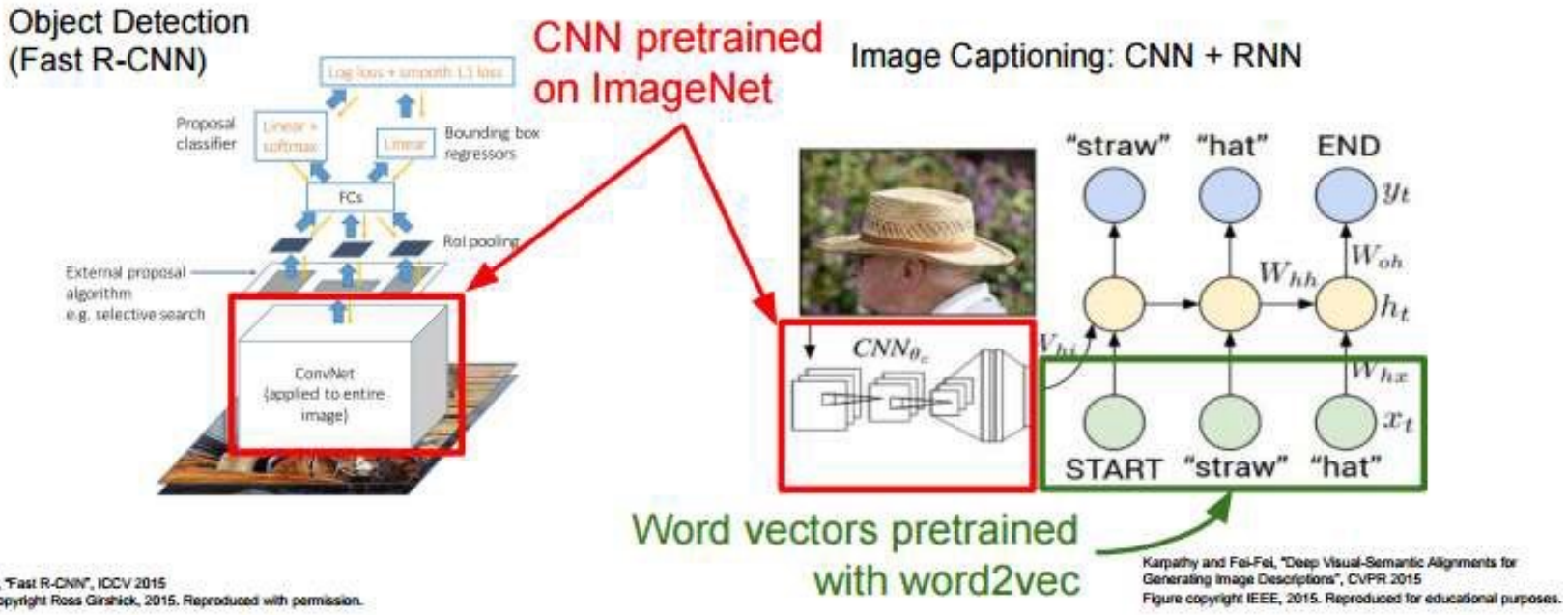


	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	You're in trouble... Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

https://pytorch.org/vision/stable/models.html

이미지 분류모델 학습 -5

Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)



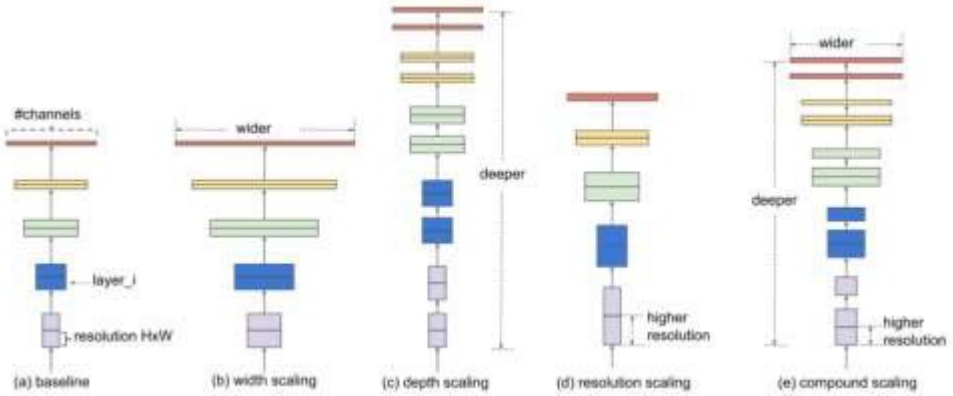
Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

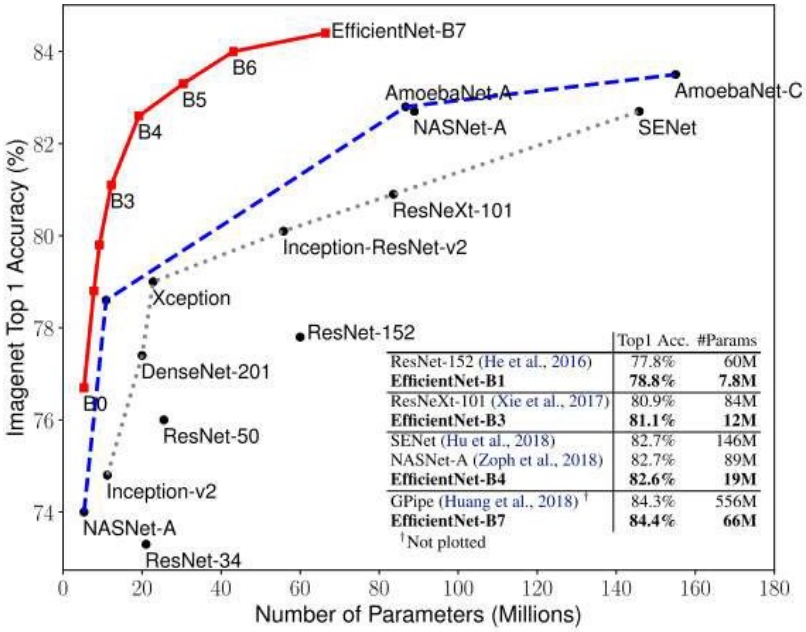
Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

https://pytorch.org/vision/stable/models.html

이미지 분류모델 학습 -6

- **SOTA 이미지 분류모델 : EfficientNet**
  - Find Small & Powerful model with AutoML
  - 참고자료 : https://keep-steady.tistory.com/35





https://arxiv.org/abs/1905.11946

Jetbot Software-Collision Avoidance 1

The resnet18 model was originally trained for a dataset that had 1000 class labels, but our dataset only has two class labels! We'll replace the final layer with a new, untrained layer that has only two outputs.

```
model.fc = torch.nn.Linear(512, 2)
```

Finally, we transfer our model for execution on the GPU

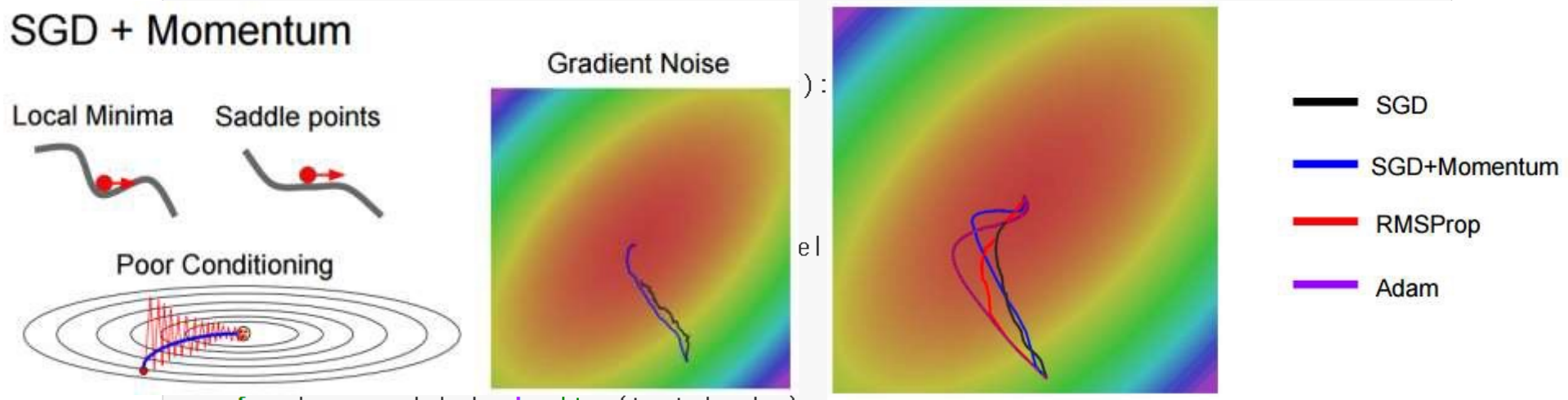
```
device = torch.device('cuda')
model = model.to(device)
```

Jetbot Software-Collision Avoidance 1

```
NUM_EPOCHS = 30
BEST_MODEL_PATH = 'best_model_resnet18.pth'
best_accuracy = 0.0

optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

for epoch in range(NUM_EPOCHS):

    for images, labels in iter(train_loader):
        images = images.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = F.cross_entropy(outputs, labels)
        loss.backward()
        optimizer.step()

    test_error_count = 0.0
    for images, labels in iter(test_loader):
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        test_error_count += float(torch.sum(torch.abs(labels - outputs.argmax(1))))

    test_accuracy = 1.0 - float(test_error_count) / float(len(test_dataset))
    print('%d: %f' % (epoch, test_accuracy))
    if test_accuracy > best_accuracy:
        torch.save(model.state_dict(), BEST_MODEL_PATH)
        best_accuracy = test_accuracy
```

Jetbot Software-Collision Avoidance 1

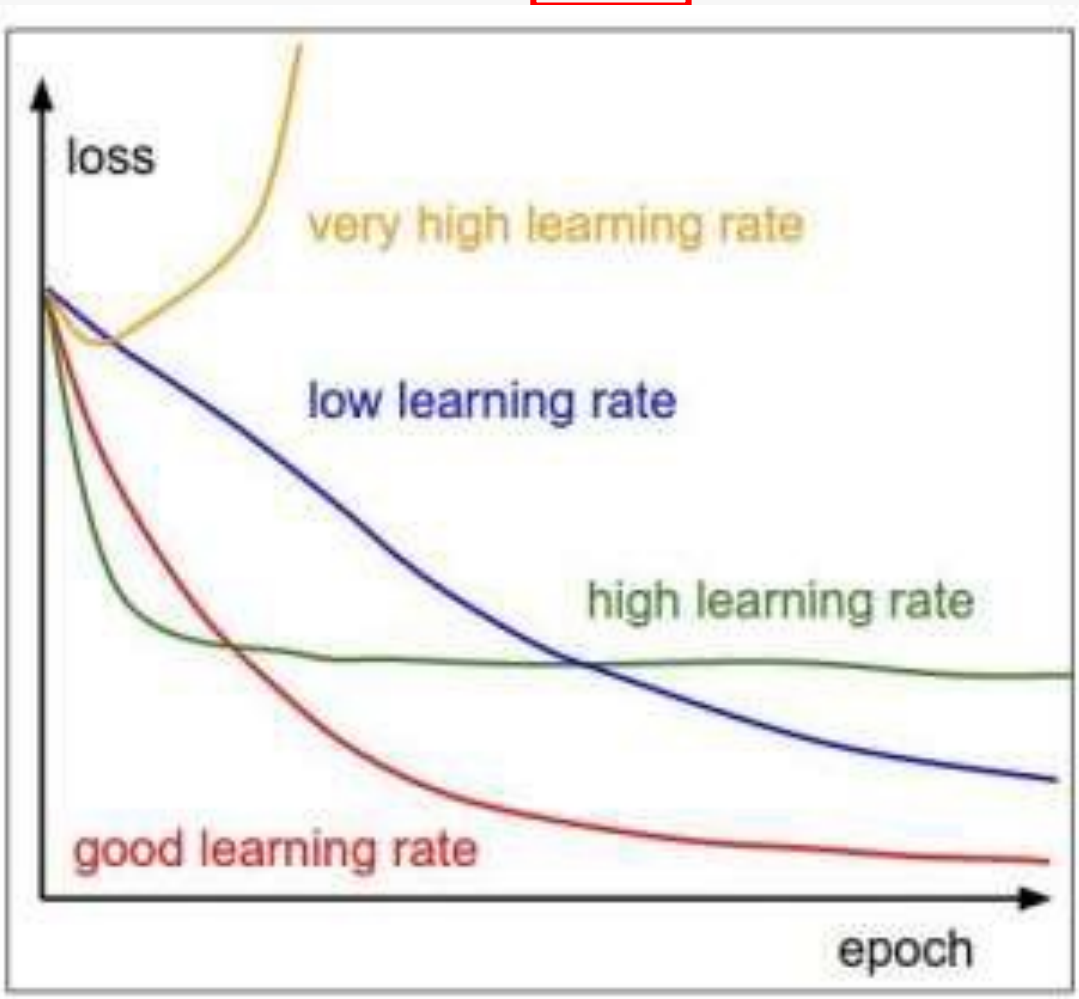
```
NUM_EPOCHS = 30
BEST_MODEL_PATH = 'best_model_resnet18.pth'
best_accuracy = 0.0

optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```



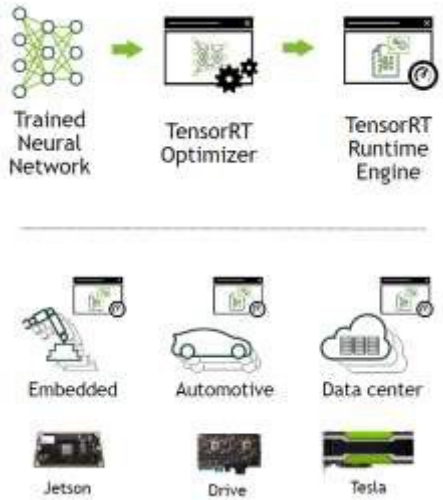

```
    for images, labels in iter(test_loader):
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        test_error_count += float(torch.sum(torch.abs(labels - outputs.argmax(1))))

    test_accuracy = 1.0 - float(test_error_count) / float(len(test_dataset))
    print('%d: %f' % (epoch, test_accuracy))
    if test_accuracy > best_accuracy:
        torch.save(model.state_dict(), BEST_MODEL_PATH)
        best_accuracy = test_accuracy
```

Jetbot Software-Collision Avoidance 1

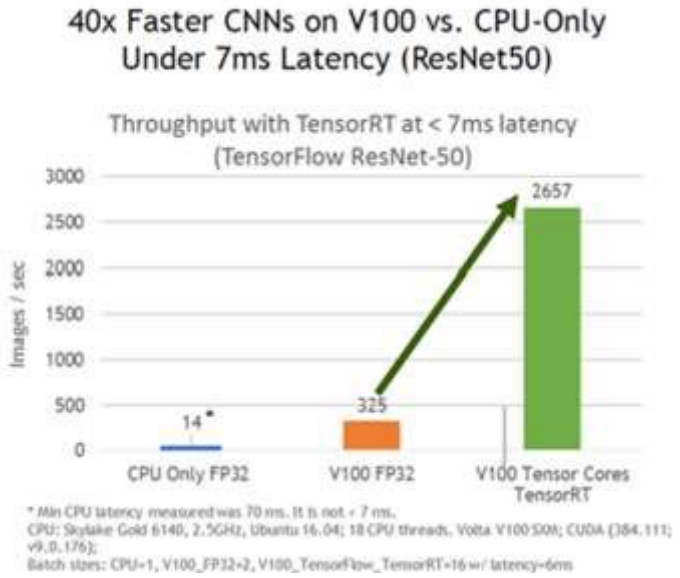
```
NUM_EPOCHS = 30
BEST_MODEL_PATH = 'best_model_resnet18.pth'
best_accuracy = 0.0

optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

for epoch i

    for ima
        ima
        lab
        opt
        out
        los
        los
        opt

    test_er
    for ima
        ima
        lab
        out
        tes

    test_ac
    print('
    if test
        tor
        bes
```



TensorRT



- **TensorRT**
  - 학습된 딥러닝 모델을 최적화
  - NVIDIA GPU 상에서의 추론 속도를 수배 ~ 수십배 까지 향상
  - 딥러닝 서비스를 개선하는데 도움을 줄 수 있는 모델 최적화 엔진
  - Caffe, Pytorch, TensorFlow 모델을 NVIDIA GPU 플랫폼(TESLA T4 , JETSON TX2, TESLA V100)에 아름답게 싣는 것
  - NVIDIA GPU 연산에 적합한 최적화 기법들을 이용하여 모델을 최적화하는 Optimizer 와 다양한 GPU에서 모델 연산을 수행하는 Runtime Engine 을 포함
    - 양자화 및 정밀도 캘리브레이션
    - 그래프 최적화
    - 커널 자동 튜닝
    - 동적 텐서 메모리 및 멀티 스트림 실행



https://eehoeskrap.tistory.com/414

Collision avoidance - TensorRT

```
import torch
import torchvision

model = torchvision.models.resnet18(pretrained=False)
model.fc = torch.nn.Linear(512, 2)
model = model.cuda().eval().half()
```

Next, load the trained weights from the best_model_resnet18.pth file that you uploaded

```
model.load_state_dict(torch.load('best_model_resnet18.pth'))
```

Currently, the model weights are located on the CPU memory execute the code below to transfer to the GPU device.

```
device = torch.device('cuda')
```

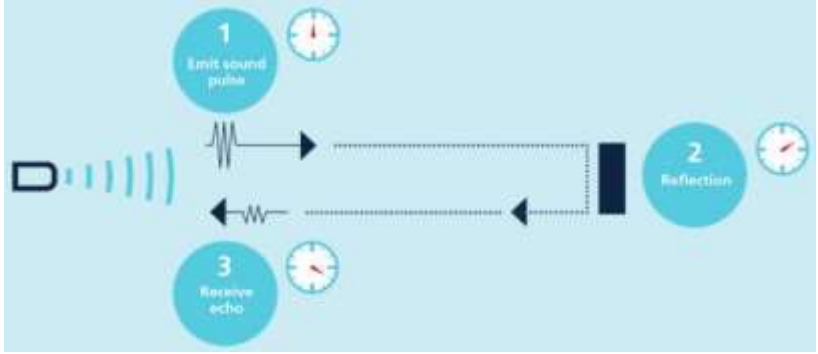
```
from torch2trt import torch2trt

data = torch.zeros((1, 3, 224, 224)).cuda().half()

model_trt = torch2trt(model, [data], fp16_mode=True)
```

Save the optimized model using the cell below

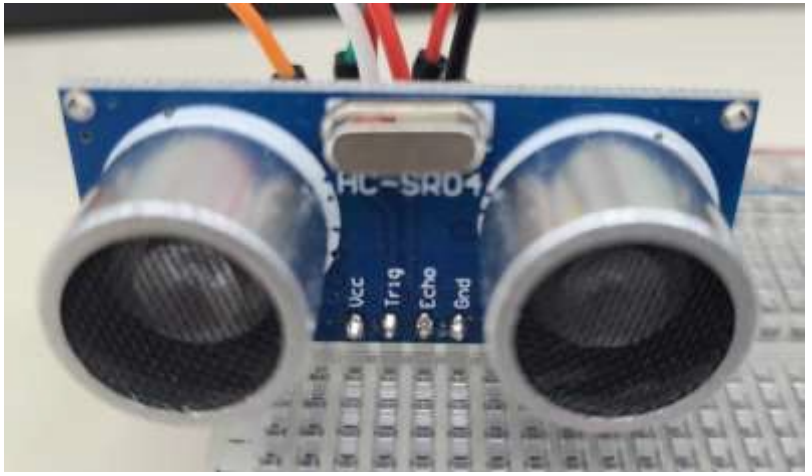
```
torch.save(model_trt.state_dict(), 'best_model_trt.pth')
```

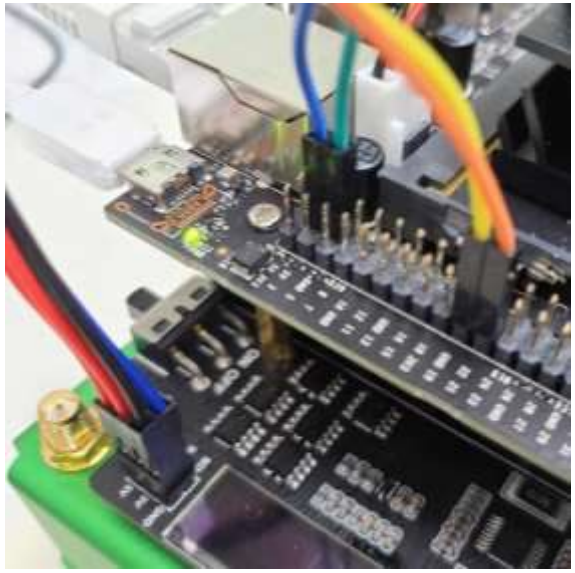
고생하셨어요~~

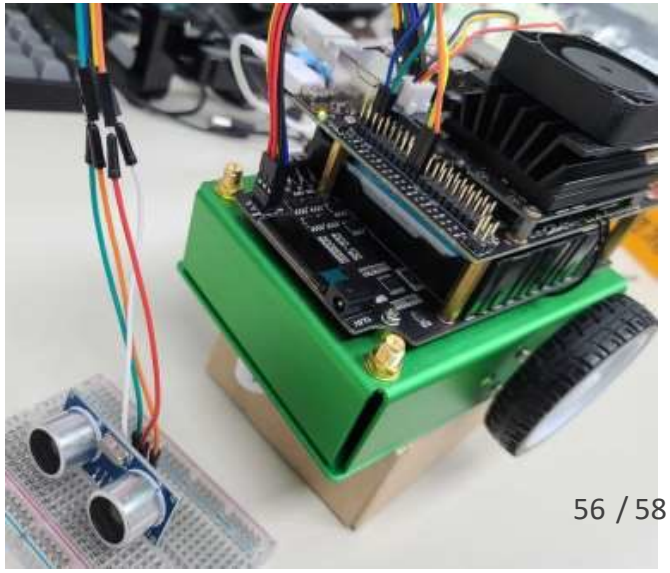
Practice Sonar sensor 1



- **Sonar sensor**
  - Sonar sensor 연결
  - 5V, trig(19), echo(21), GND







Practice Sonar sensor 2

- **Sonar sensor**
  - Sonar sensor 연결
  - jupyterlab 접속
  - 터미널을 연다
  - pip3 install Jetson.GPIO
  - python3
    - >> import RPi.GPIO as GPOI
  - python3 sonar.py