PyTorch

조 영 혁

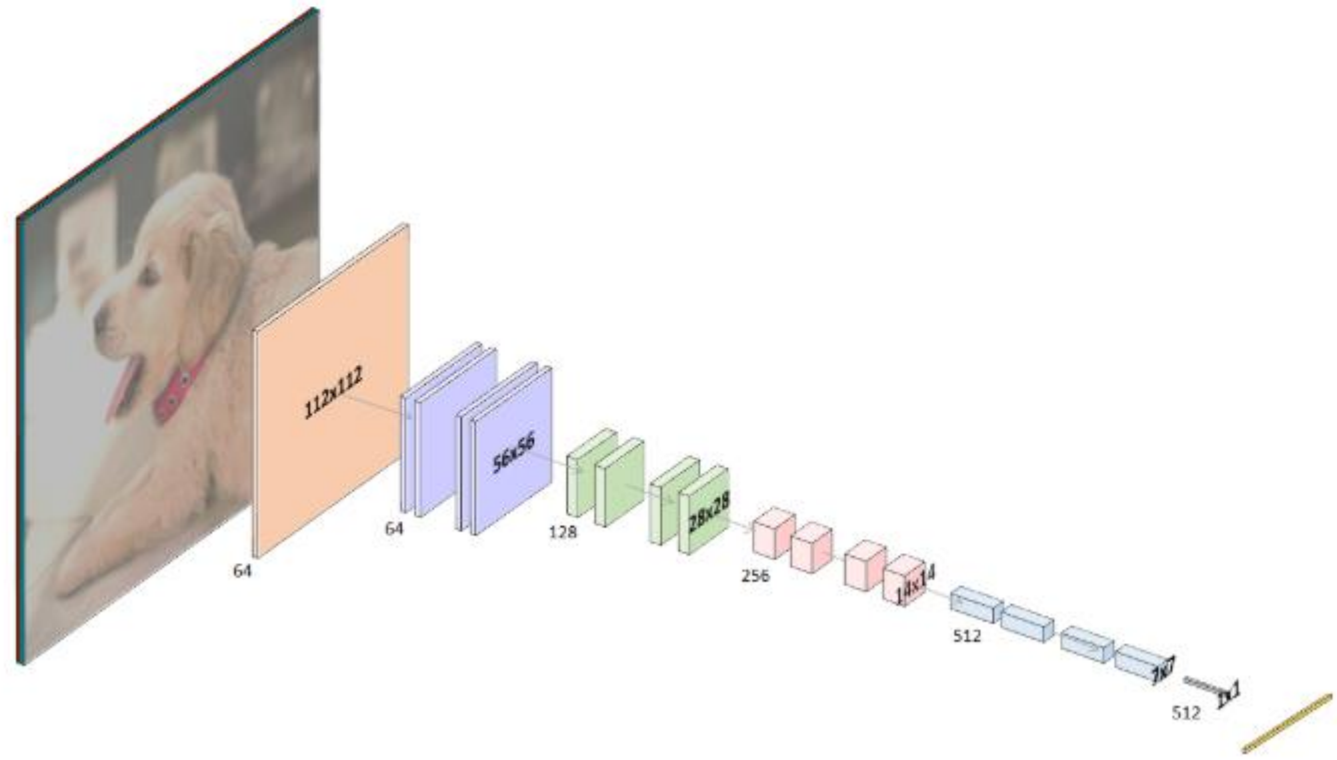
노다시스템

Transfer Learning with PyTorch

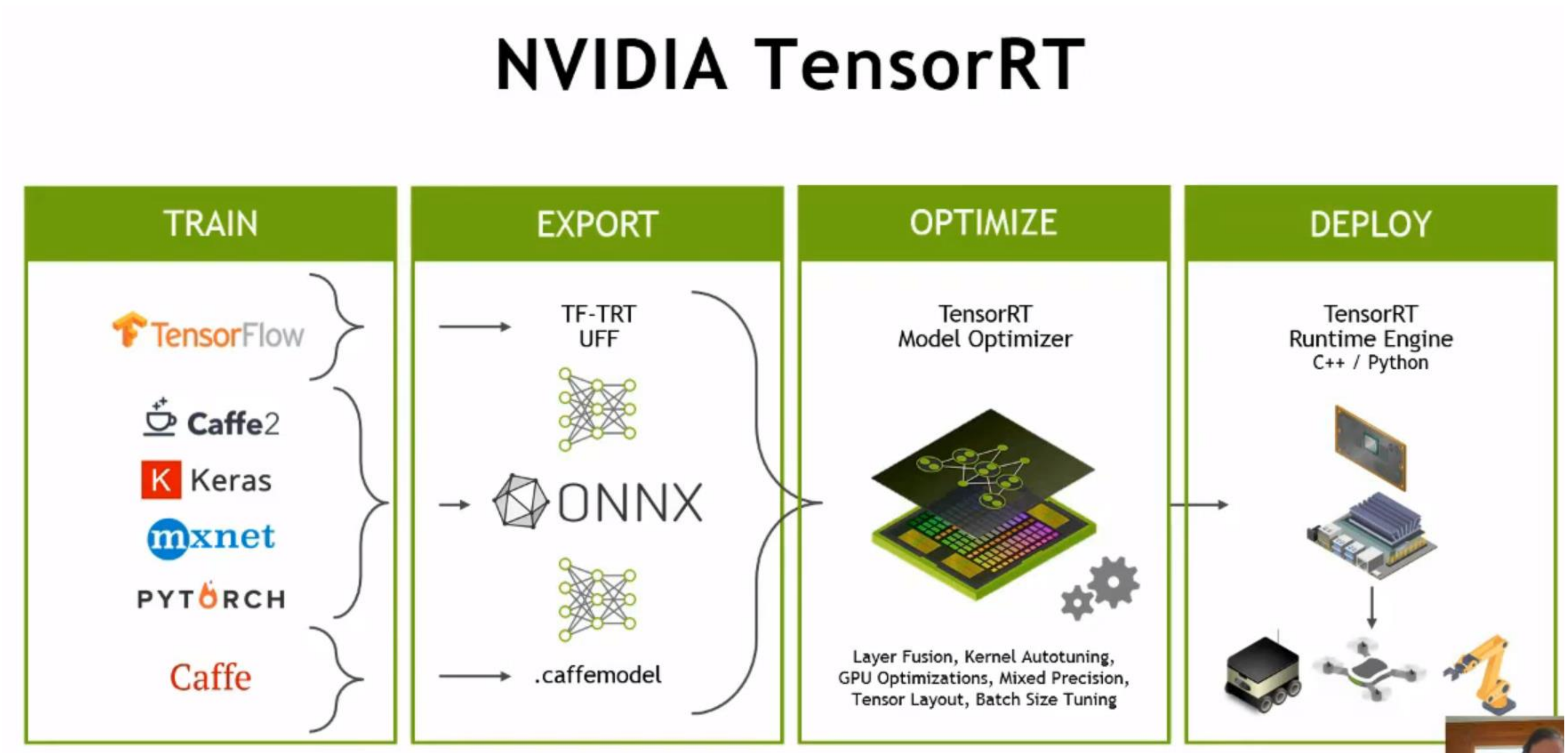
ResNet-18

- training is performed on aPC, Server, Cloud

- **PyTorch** is the machine learning framework



Jetson Nano

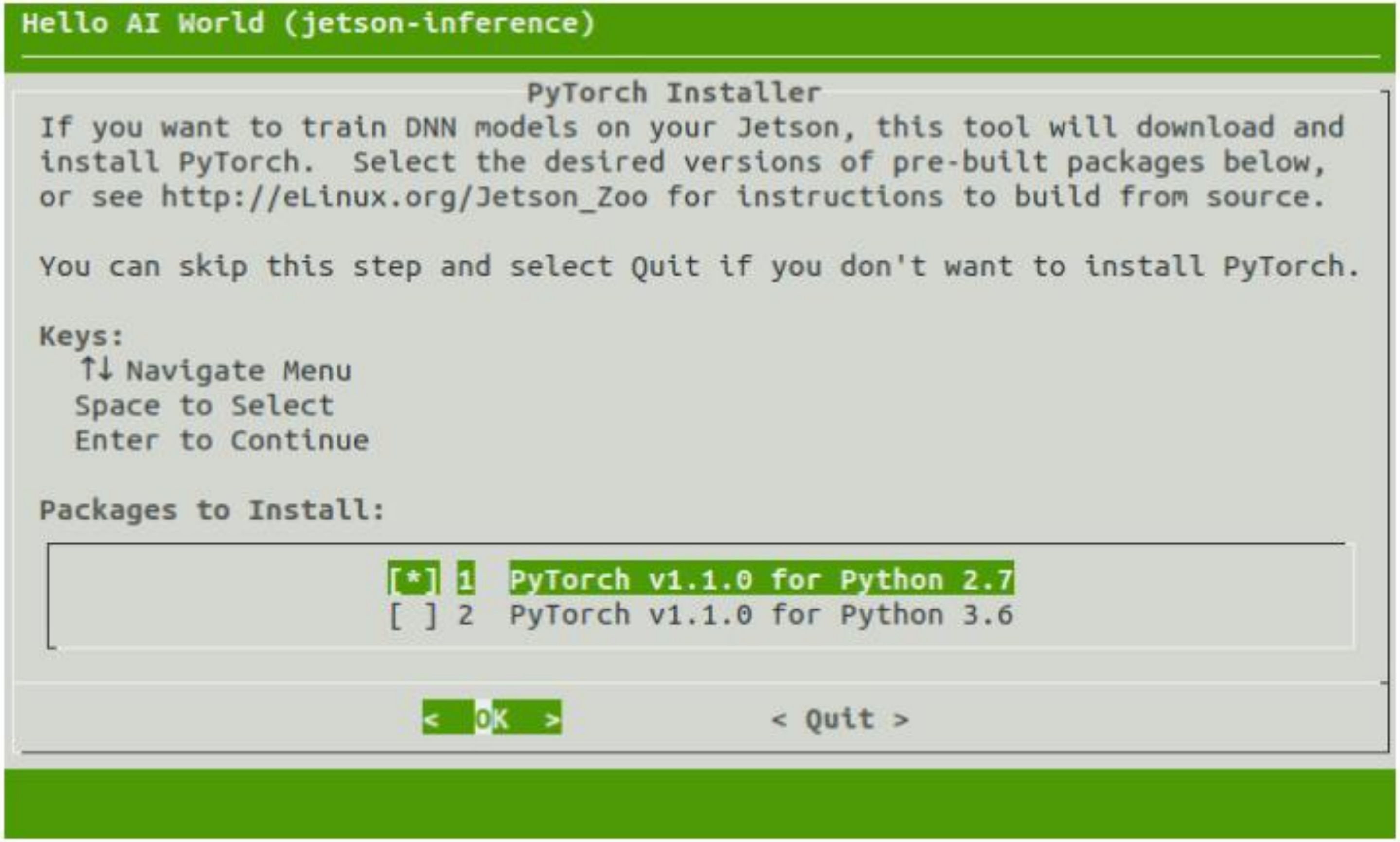


ONNX(open neural network exchange : 페이스북과 마이크로소프트가 만든 개방형 포맷으로 다른 프레임워크에서 모델을 교환할 수 있도록 만든것

PyTorch

Installing PyTorch

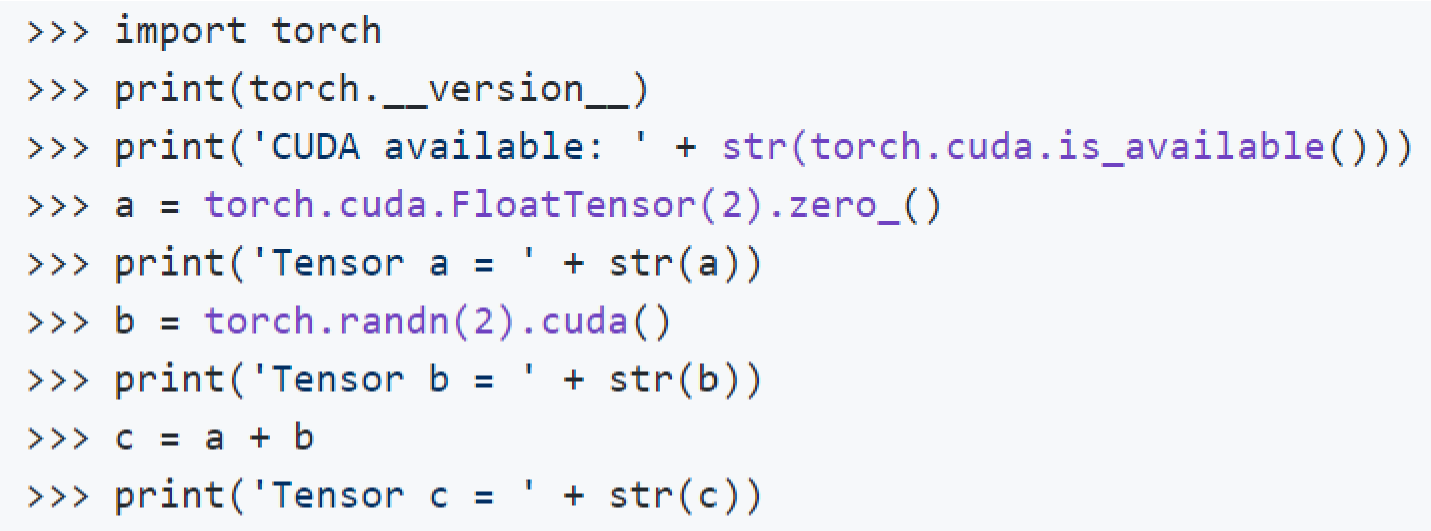
```
$ cd jetson-inference/build
$ ./install-pytorch.sh
```



```
Hello AI World (jetson-inference)

                              PyTorch Installer
 If you want to train DNN models on your Jetson, this tool will download and
 install PyTorch.  Select the desired versions of pre-built packages below,
 or see http://eLinux.org/Jetson_Zoo for instructions to build from source.

 You can skip this step and select Quit if you don't want to install PyTorch.

 Keys:
    ↑↓ Navigate Menu
   Space to Select
   Enter to Continue

 Packages to Install:

                  [*] 1   PyTorch v1.1.0 for Python 2.7
                  [ ] 2   PyTorch v1.1.0 for Python 3.6

                 <  OK  >                 < Quit >
```

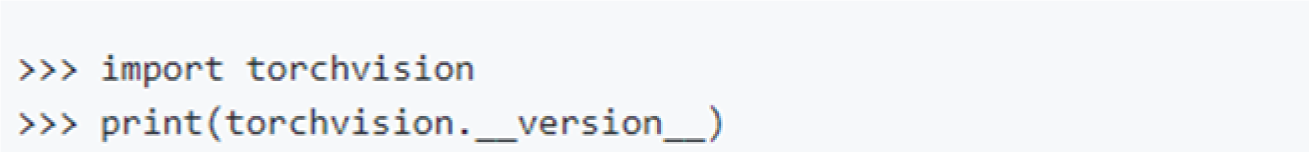
PyTorch

=> test that PyTorch was installed correctly and detects your GPU



```
>>> import torch
>>> print(torch.__version__)
>>> print('CUDA available: ' + str(torch.cuda.is_available()))
>>> a = torch.cuda.FloatTensor(2).zero_()
>>> print('Tensor a = ' + str(a))
>>> b = torch.randn(2).cuda()
>>> print('Tensor b = ' + str(b))
>>> c = a + b
>>> print('Tensor c = ' + str(c))
```

1.6.0



```
>>> import torchvision
>>> print(torchvision.__version__)
```

0.7.0

PyTorch

```
$ swapon –s          : To check the usage   메모리 공간 확인
```

Mounting Swap

```
$ sudo fallocate   -l   4G   /mnt/4GB.swap
$ sudo mkswap   /mnt/4GB.swap
$ sudo swapon   /mnt/4GB.swap
```

/etc/fstab에 아래 항목 추가

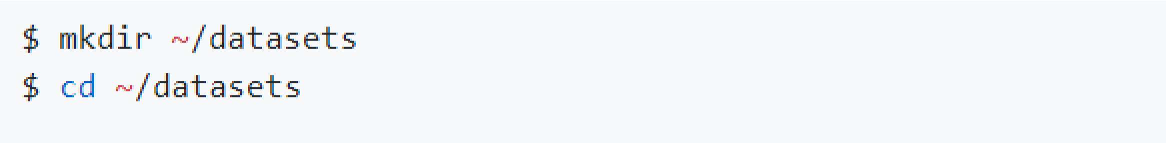
```
$ sudo chmod 777 fstab

/mnt/4GB.swap   none   swap   sw   0   0
```

```
$ swapon –s          : To check the usage
```

<< rebooting >>

PyTorch-Re-training on the Cat/Dog Dataset

Training Datasets



```
$ mkdir ~/datasets
$ cd ~/datasets
```

< jetson nano에서 직접 download , unzip >

```
$ wget https://nvidia.box.com/shared/static/o577zd8yp3lmxf5zhm38svrbrv45am3y.gz -O cat_dog.tar.gz
$ tar xvzf cat_dog.tar.gz
```

< google drive나 nvidia에서 download , unzip >

https://drive.google.com/file/d/16E3yFvVS2DouwgIl4TPFJvMlhGpnYWKF/view?usp=sharing

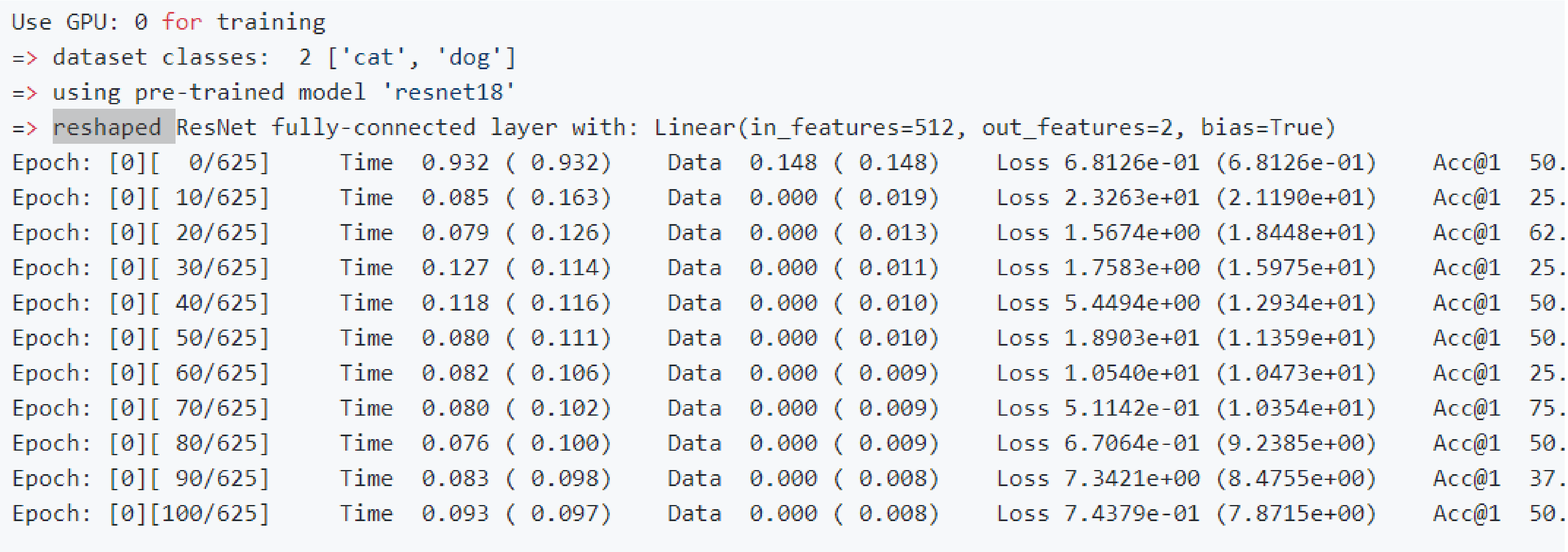
https://nvidia.box.com/s/o577zd8yp3lmxf5zhm38svrbrv45am3y

PyTorch-Re-training on the Cat/Dog Dataset

Re-training ResNet-18 Model

$ cd   jetson-inference/python/training/classification

$ python3   train.py --model-dir=cat_dog   ~/datasets/cat_dog



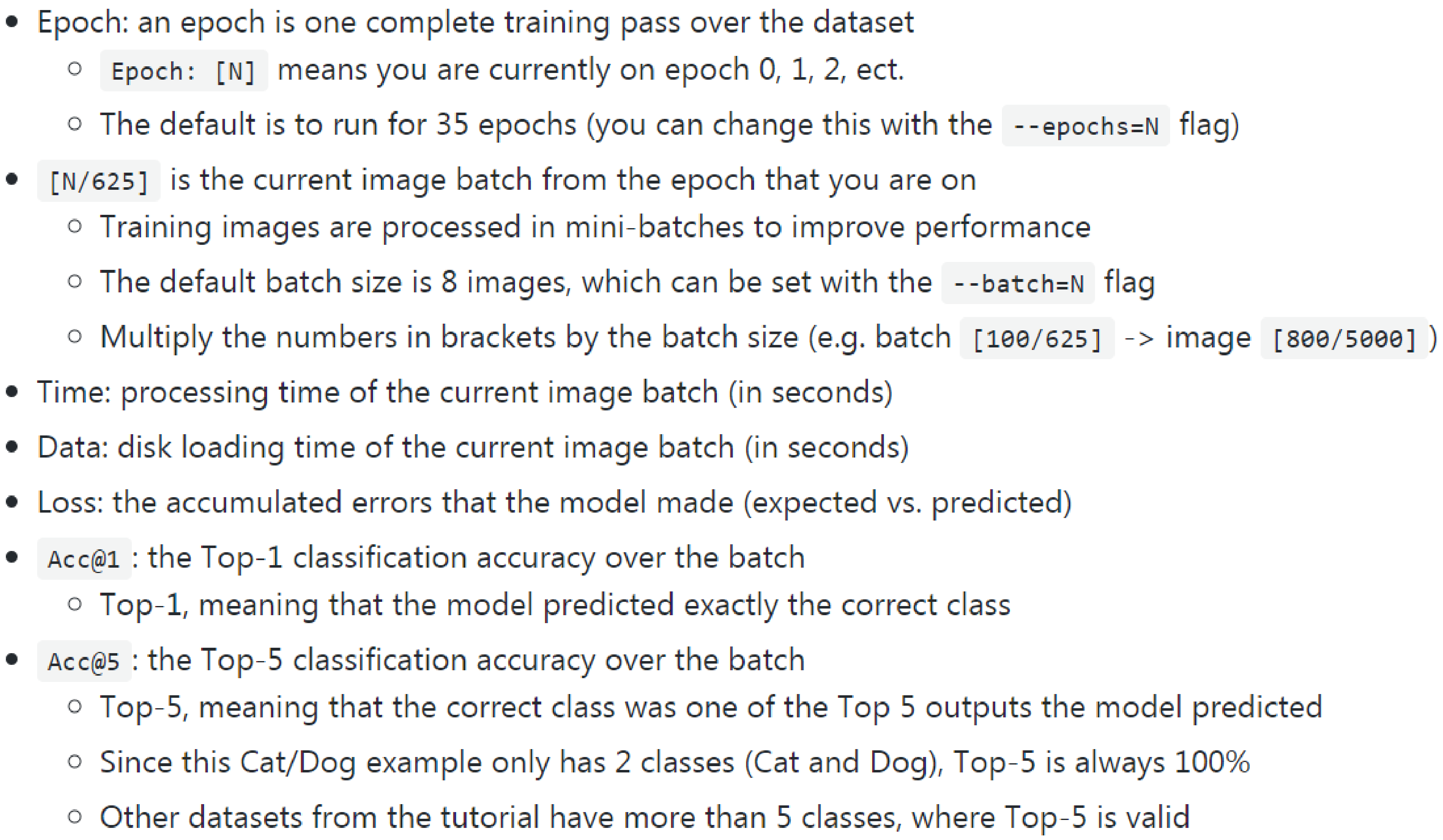
```
Use GPU: 0 for training
=> dataset classes:  2 ['cat', 'dog']
=> using pre-trained model 'resnet18'
=> reshaped ResNet fully-connected layer with: Linear(in_features=512, out_features=2, bias=True)
Epoch: [0][  0/625]    Time  0.932 ( 0.932)    Data  0.148 ( 0.148)    Loss 6.8126e-01 (6.8126e-01)    Acc@1  50.
Epoch: [0][ 10/625]    Time  0.085 ( 0.163)    Data  0.000 ( 0.019)    Loss 2.3263e+01 (2.1190e+01)    Acc@1  25.
Epoch: [0][ 20/625]    Time  0.079 ( 0.126)    Data  0.000 ( 0.013)    Loss 1.5674e+00 (1.8448e+01)    Acc@1  62.
Epoch: [0][ 30/625]    Time  0.127 ( 0.114)    Data  0.000 ( 0.011)    Loss 1.7583e+00 (1.5975e+01)    Acc@1  25.
Epoch: [0][ 40/625]    Time  0.118 ( 0.116)    Data  0.000 ( 0.010)    Loss 5.4494e+00 (1.2934e+01)    Acc@1  50.
Epoch: [0][ 50/625]    Time  0.080 ( 0.111)    Data  0.000 ( 0.010)    Loss 1.8903e+01 (1.1359e+01)    Acc@1  50.
Epoch: [0][ 60/625]    Time  0.082 ( 0.106)    Data  0.000 ( 0.009)    Loss 1.0540e+01 (1.0473e+01)    Acc@1  25.
Epoch: [0][ 70/625]    Time  0.080 ( 0.102)    Data  0.000 ( 0.009)    Loss 5.1142e-01 (1.0354e+01)    Acc@1  75.
Epoch: [0][ 80/625]    Time  0.076 ( 0.100)    Data  0.000 ( 0.009)    Loss 6.7064e-01 (9.2385e+00)    Acc@1  50.
Epoch: [0][ 90/625]    Time  0.083 ( 0.098)    Data  0.000 ( 0.008)    Loss 7.3421e+00 (8.4755e+00)    Acc@1  37.
Epoch: [0][100/625]    Time  0.093 ( 0.097)    Data  0.000 ( 0.008)    Loss 7.4379e-01 (7.8715e+00)    Acc@1  50.
```

Ctrl+C : stop

$ python3   train.py   --model-dir=cat_dog --resume ~/datasets/cat_dog : Continued

PyTorch-Re-training on the Cat/Dog Dataset

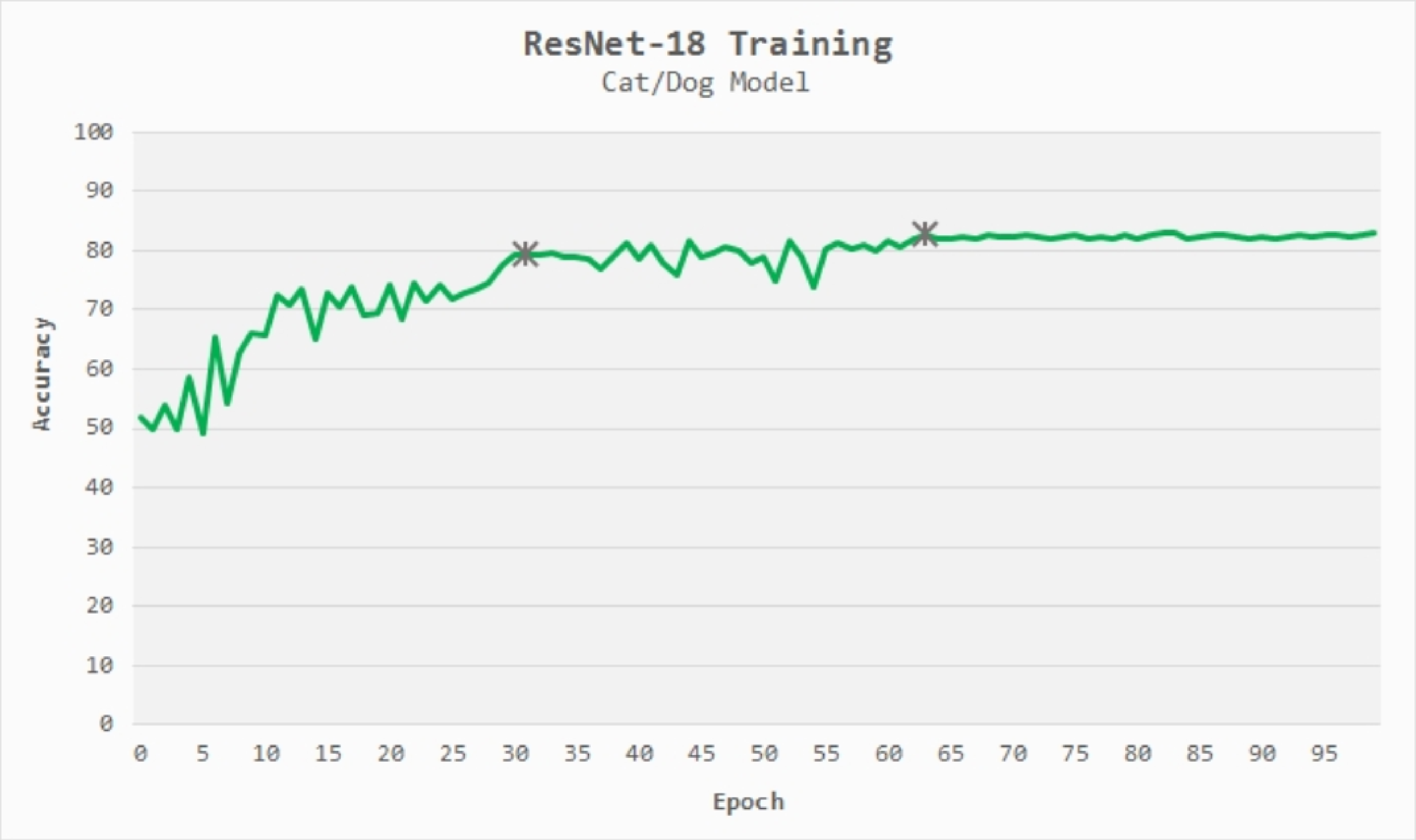
Re-training ResNet-18 Model



- Epoch: an epoch is one complete training pass over the dataset
  - `Epoch: [N]` means you are currently on epoch 0, 1, 2, ect.
  - The default is to run for 35 epochs (you can change this with the `--epochs=N` flag)
- `[N/625]` is the current image batch from the epoch that you are on
  - Training images are processed in mini-batches to improve performance
  - The default batch size is 8 images, which can be set with the `--batch=N` flag
  - Multiply the numbers in brackets by the batch size (e.g. batch `[100/625]` -> image `[800/5000]`)
- Time: processing time of the current image batch (in seconds)
- Data: disk loading time of the current image batch (in seconds)
- Loss: the accumulated errors that the model made (expected vs. predicted)
- `Acc@1` : the Top-1 classification accuracy over the batch
  - Top-1, meaning that the model predicted exactly the correct class
- `Acc@5` : the Top-5 classification accuracy over the batch
  - Top-5, meaning that the correct class was one of the Top 5 outputs the model predicted
  - Since this Cat/Dog example only has 2 classes (Cat and Dog), Top-5 is always 100%
  - Other datasets from the tutorial have more than 5 classes, where Top-5 is valid

PyTorch-Re-training on the Cat/Dog Dataset

On this dataset of 5000 images
- training ResNet-18 takes approximately ~7-8 minutes per epoch on Jetson Nano
- around 4 hours to train the model to 35 epochs and 80% classification accuracy
- At around epoch 30, the ResNet-18 model reaches 80% accuracy



PyTorch-Re-training on the Cat/Dog Dataset

Converting the Model to ONNX

../classification directory에
 chkpoint.pth.tar , model_best.pth.tar 파일이 존재하는가 체크

- TensorRT가 load 가능한 ONNX format으로 변환

```
$ python3    onnx_export.py   --model-dir=cat_dog
```

⇒resnet18.onnx 모델 생성
(jetson-inference/python/training/classification/cat_dog/)
(jetson-inference/python/training/classification/cat_dog_epoch_100/)

PyTorch-Re-training on the Cat/Dog Dataset

For 100 epochs

https://nvidia.box.com/s/zlvb4y43djygotpjn6azjhwu0r3j0yxc

아래 디렉토리로 다운받은 것을 복사하라
jetson-inference/python/training/classification/

PyTorch-Re-training on the Cat/Dog Dataset

Processing Images with TensorRT

```
$  cd ~/jetson-inference/python/training/classification/
```

에서 resnet18.onnx 파일 확인

데이터 셋 등록

```
$ DATASET=~/datasets/cat_dog
```

```
$ ./imagenet-console.py   --model=cat_dog/resnet18.onnx   --input_blob=input_0   --output_blob=output_0   --labels=$DATASET/labels.txt $DATASET/test/cat/01.jpg   cat.jpg
```

```
$ ./imagenet-console.py --model=cat_dog/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=$DATASET/labels.txt $DATASET/test/dog/01.jpg dog.jpg
```

PyTorch-Re-training on the Cat/Dog Dataset

Epoch 변경후 학습

```
$ cd   jetson-inference/python/training/classification
$ python   train.py --epochs=50  --model-dir=cat_dog   ~/datasets/cat_dog
```

```
$ python    onnx_export.py   --model-dir=cat_dog
```

```
$ ./imagenet-console.py   --model=cat_dog/resnet18.onnx   --
input_blob=input_0   --output_blob=output_0   --labels=$DATASET/labels.txt
$DATASET/test/cat/01.jpg   cat.jpg
```

PyTorch-Re-training on the Cat/Dog Dataset

다른 network로 학습

```
$ cd   jetson-inference/python/training/classification
$ python   train.py –arch=alexnet  --model-dir=cat_dog   ~/datasets/cat_dog


$ python    onnx_export.py   --model-dir=cat_dog


$ imagenet-console.py   --model=cat_dog/resnet18.onnx   --
input_blob=input_0   --output_blob=output_0   --labels=$DATASET/labels.txt
$DATASET/test/cat/01.jpg   cat.jpg
```

PyTorch-Re-training on the Cat/Dog Dataset

Detection을 학습

```
$ cd   jetson-inference/python/training/detection
$ python   train.py --model-dir=cat_dog   ~/datasets/cat_dog
```

```
$ python    onnx_export.py   --model-dir=cat_dog
```

```
$ detnet-console.py   --model=cat_dog/resnet18.onnx   --input_blob=input_0
--output_blob=output_0   --labels=$DATASET/labels.txt
$DATASET/test/cat/01.jpg   cat.jpg
```

PyTorch-Re-training on the Cat/Dog Dataset

segmentation 학습

```
$ cd   jetson-inference/python/training/segmentation
$ python   train.py --model-dir=cat_dog   ~/datasets/cat_dog    :fcn_resnet18
```

```
$ python    onnx_export.py   --model-dir=cat_dog
```

```
$ segnet-console.py   --model=cat_dog/resnet18.onnx   --
input_blob=input_0   --output_blob=output_0   --labels=$DATASET/labels.txt
$DATASET/test/cat/01.jpg   cat.jpg
```

PyTorch-Re-training on the Cat/Dog Dataset

Running the Live Camera Program

```
$ imagenet-camera.py   --model=cat_dog/resnet18.onnx
  --input_blob=input_0    --output_blob=output_0
  --labels=$DATASET/labels.txt
```

⇒resnet18.onnx 모델 생성

(jetson-inference/python/training/classification/cat_dog/)

(jetson-inference/python/training/classification/cat_dog_epoch_100/)

PyTorch-Re-training on the PlantCLEF Dataset

- Plant data set is a 1.5GB subset that includes 10,475 training images, 1,155 validation images, and 30 test images across 20 classes of plants and trees.

```
$ cd  ~/datasets

$ wget  https://nvidia.box.com/shared/static/vbsywpw5iqy7r38j78xs0ctalg7jrg79.gz  -O
PlantCLEF_Subset.tar.gz

$ tar xvzf PlantCLEF_Subset.tar.gz
```

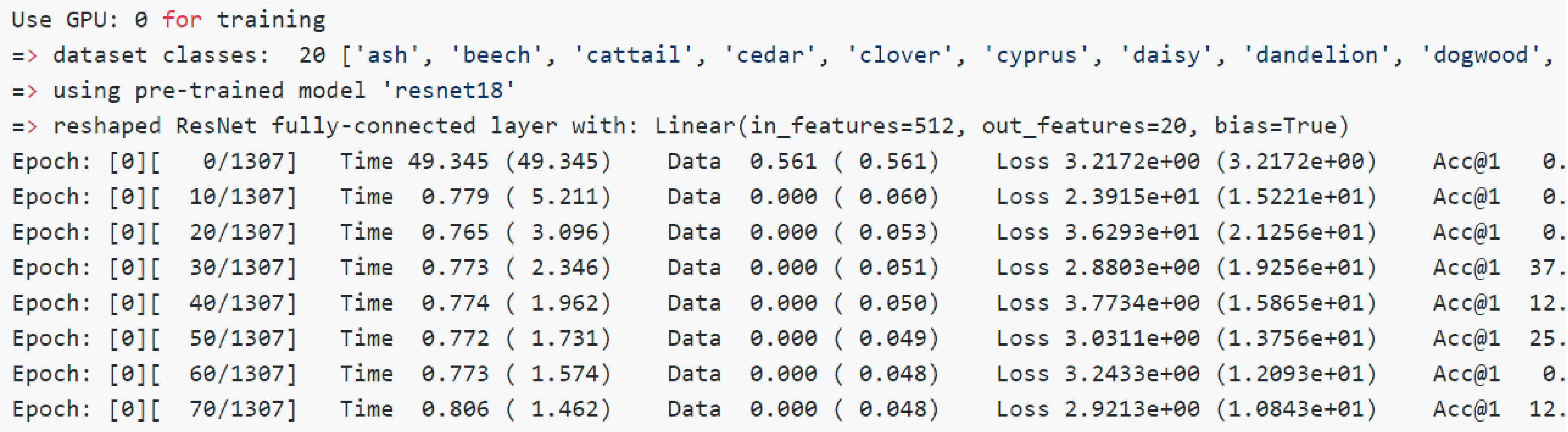
Mirrors of the dataset are available here:
https://drive.google.com/file/d/14pUv-ZLHtRR-zCYjznr78mytFcnuR_1D/view?usp=sharing
https://nvidia.box.com/s/vbsywpw5iqy7r38j78xs0ctalg7jrg79

PyTorch-Re-training on the PlantCLEF Dataset

Re-training ResNet-18 Model

$ cd   jetson-inference/python/training/classification

$ python   train.py --model-dir=plants ~/datasets/PlantCLEF_Subset



```
Use GPU: 0 for training
=> dataset classes:  20 ['ash', 'beech', 'cattail', 'cedar', 'clover', 'cyprus', 'daisy', 'dandelion', 'dogwood',
=> using pre-trained model 'resnet18'
=> reshaped ResNet fully-connected layer with: Linear(in_features=512, out_features=20, bias=True)
Epoch: [0][   0/1307]   Time 49.345 (49.345)    Data  0.561 ( 0.561)    Loss 3.2172e+00 (3.2172e+00)    Acc@1   0.
Epoch: [0][  10/1307]   Time  0.779 ( 5.211)    Data  0.000 ( 0.060)    Loss 2.3915e+01 (1.5221e+01)    Acc@1   0.
Epoch: [0][  20/1307]   Time  0.765 ( 3.096)    Data  0.000 ( 0.053)    Loss 3.6293e+01 (2.1256e+01)    Acc@1   0.
Epoch: [0][  30/1307]   Time  0.773 ( 2.346)    Data  0.000 ( 0.051)    Loss 2.8803e+00 (1.9256e+01)    Acc@1  37.
Epoch: [0][  40/1307]   Time  0.774 ( 1.962)    Data  0.000 ( 0.050)    Loss 3.7734e+00 (1.5865e+01)    Acc@1  12.
Epoch: [0][  50/1307]   Time  0.772 ( 1.731)    Data  0.000 ( 0.049)    Loss 3.0311e+00 (1.3756e+01)    Acc@1  25.
Epoch: [0][  60/1307]   Time  0.773 ( 1.574)    Data  0.000 ( 0.048)    Loss 3.2433e+00 (1.2093e+01)    Acc@1   0.
Epoch: [0][  70/1307]   Time  0.806 ( 1.462)    Data  0.000 ( 0.048)    Loss 2.9213e+00 (1.0843e+01)    Acc@1  12.
```

this completed model that was trained for a full 100 epochs

https://nvidia.box.com/s/dslt9b0hqq7u71o6mzvy07w0onn0tw66

PyTorch-Re-training on the PlantCLEF Dataset

Converting the Model to ONNX

Chkpoint.pth.tar / model_best.pth.tar

- TensorRT가 load 가능한 ONNX format으로 변환

```
$ python    onnx_export.py   --model-dir=plants
```

⇒resnet18.onnx 모델 생성

(jetson-inference/python/training/classification/plants/)

PyTorch-Re-training on the PlantCLEF Dataset

Processing Images with TensorRT

```
$  cd ~/jetson-inference/python/training/classification/
```

Path 등록

```
$ DATASET=~/datasets/PlantCLEF_Subset

$ imagenet-console  --model=plants/resnet18.onnx  --input_blob=input_0  --output_blob=output_0  --labels=$DATASET/labels.txt   $DATASET/test/cattail.jpg cattail.jpg

$  imagenet-console --model=plants/resnet18.onnx --input_blob=input_0  --output_blob=output_0 --labels=$DATASET/labels.txt $DATASET/test/elm.jpg elm.jpg
```

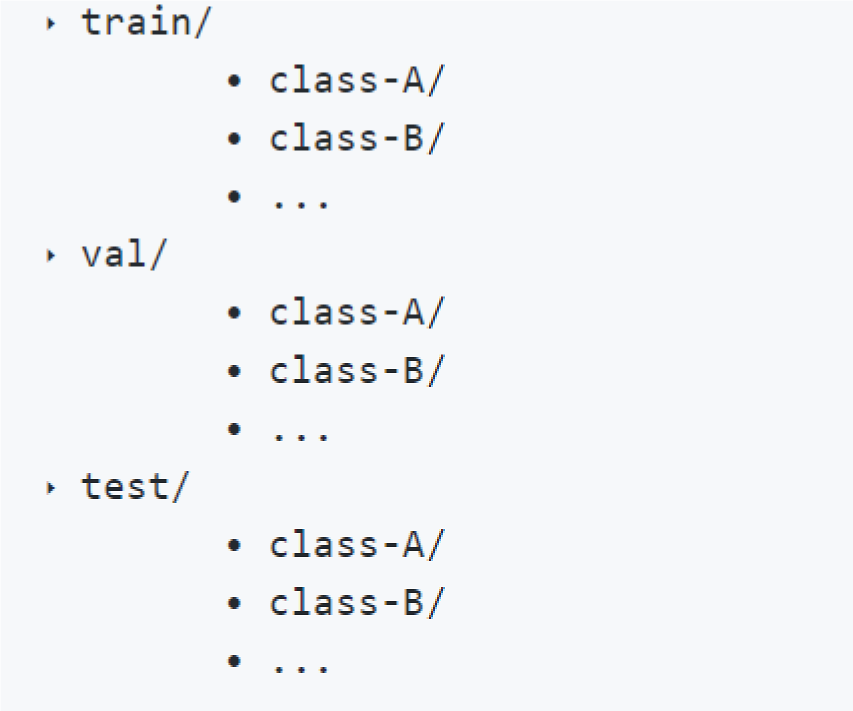
PyTorch-Re-training on the PlantCLEF Dataset

Running the Live Camera Program

```
$ imagenet-camera.py  --model=plants/resnet18.onnx  --input_blob=input_0  --output_blob=output_0  --labels=$DATASET/labels.txt
```

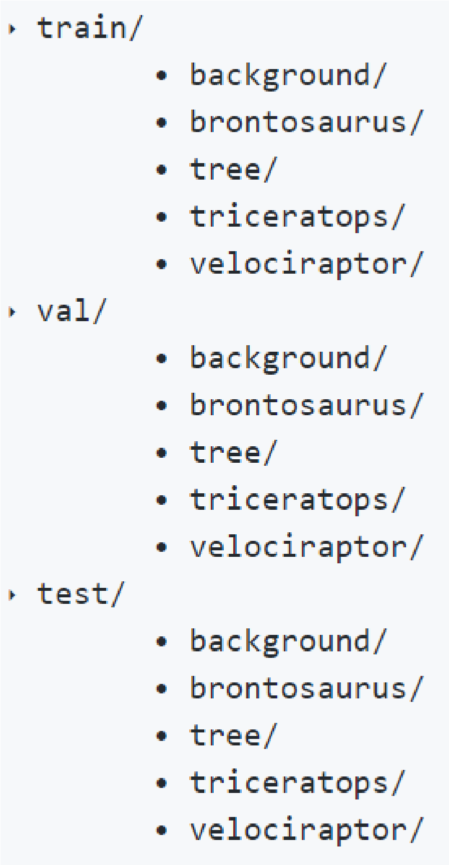
PyTorch-Collecting your own Datasets

1. Make directory of dataset



- ▸ train/
  - class-A/
  - class-B/
  - ...
- ▸ val/
  - class-A/
  - class-B/
  - ...
- ▸ test/
  - class-A/
  - class-B/
  - ...

2. camera-capture에 의하여 image capture



- ▸ train/
  - background/
  - brontosaurus/
  - tree/
  - triceratops/
  - velociraptor/
- ▸ val/
  - background/
  - brontosaurus/
  - tree/
  - triceratops/
  - velociraptor/
- ▸ test/
  - background/
  - brontosaurus/
  - tree/
  - triceratops/
  - velociraptor/

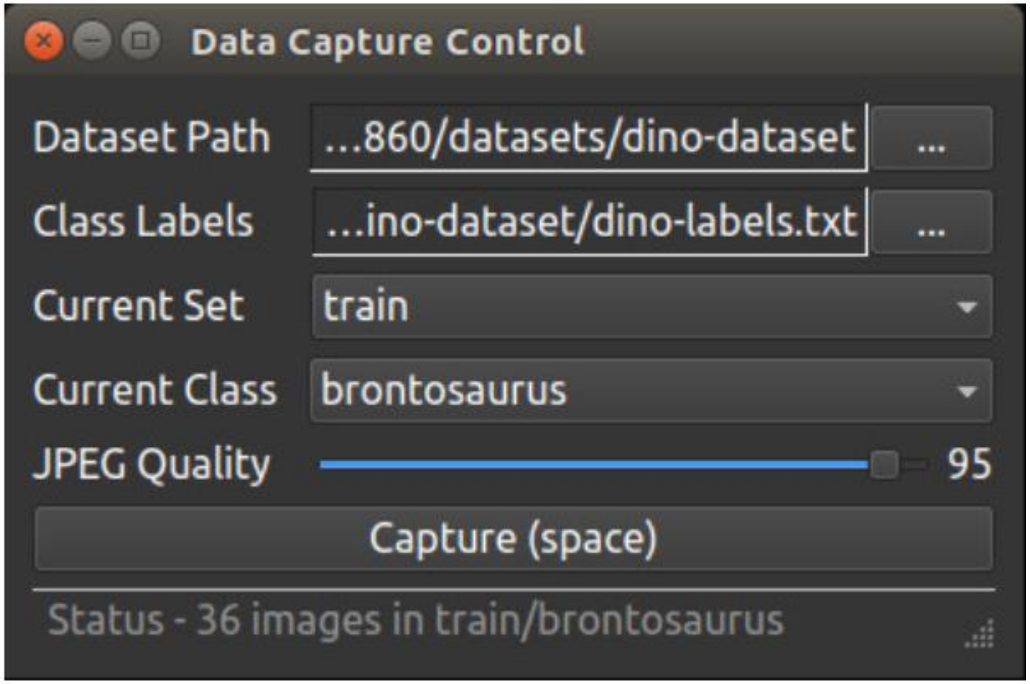
PyTorch-Collecting your own Datasets

Creating the Label File

< 100 training images >

Collecting Data

```
$ camera-capture                              # using default MIPI CSI camera (1280x720)
$ camera-capture --camera=/dev/video0         # using V4L2 camera /dev/video0 (1280x720)
$ camera-capture --width=640 --height=480 # using default MIPI CSI camera (640x480)
```



Dataset Path: ...860/datasets/dino-dataset

Class Labels: ...ino-dataset/dino-labels.txt

Current Set: train

Current Class: brontosaurus

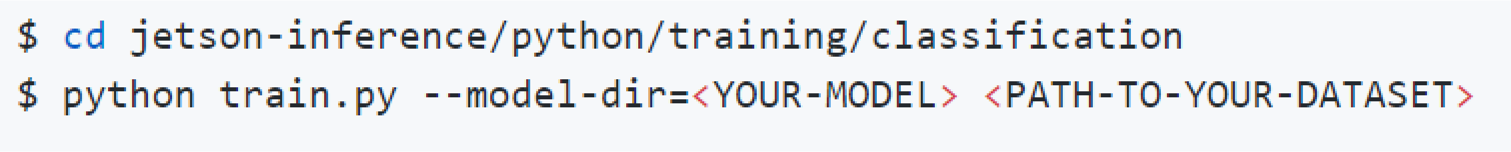
JPEG Quality: 95

Status - 36 images in train/brontosaurus

-It's recommended to collect **at least 100 training images per class(train)** be fore attempting training.
- A rule of thumb for the **validation set(val)** is that it should be roughly 10-20% the size of the training set

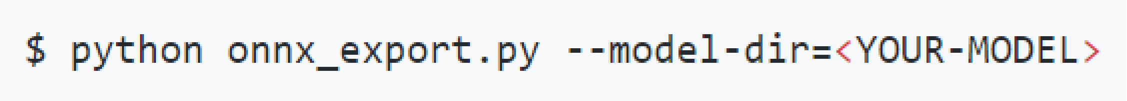
PyTorch-Collecting your own Datasets

Training your Model



```
$ cd jetson-inference/python/training/classification
$ python train.py --model-dir=<YOUR-MODEL> <PATH-TO-YOUR-DATASET>
```

Converting the Model to ONNX



```
$ python onnx_export.py --model-dir=<YOUR-MODEL>
```

imagenet-camera.py --model=<YOUR-MODEL>/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=$DATASET/labels.txt

THANK YOU

Suggestions Questions