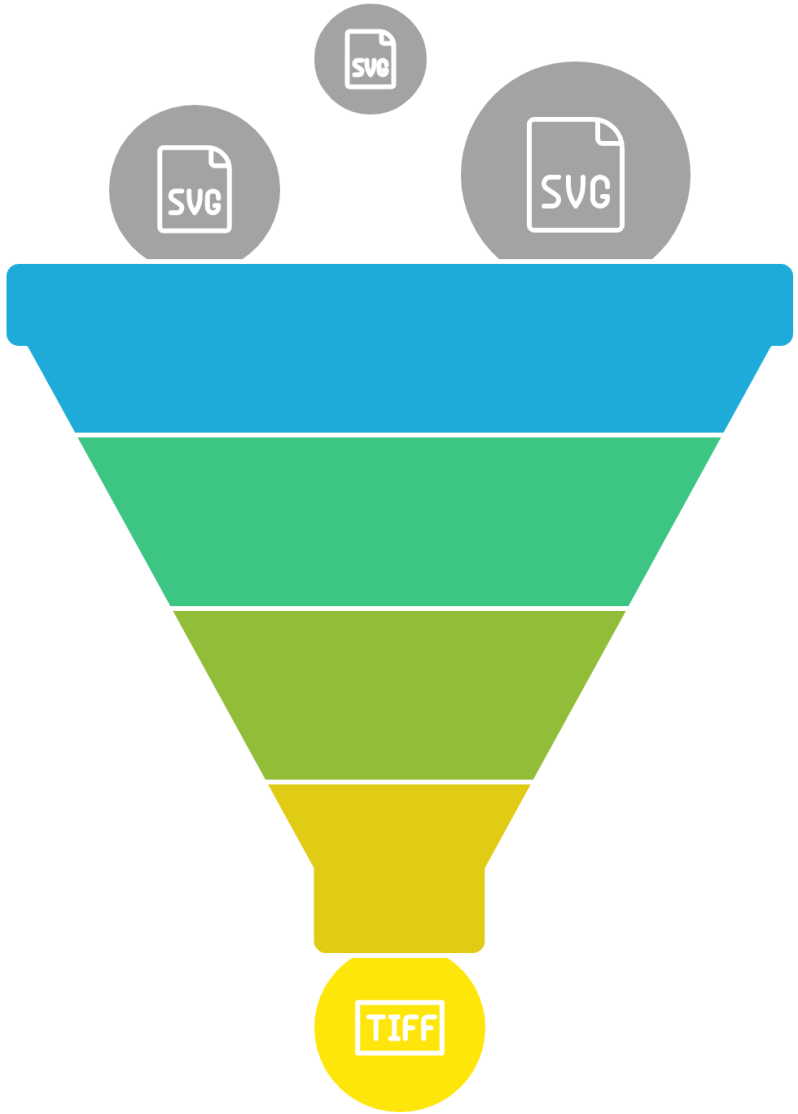
Capstone Project

Edge computing device programming for AI projects

Lecture 2: Image classification

Dr. Wilton Fok & Carol Chen

Image classification



Input Processing

Convert images to numerical data

Feature Extraction

Identify patterns in images

Feature Selection

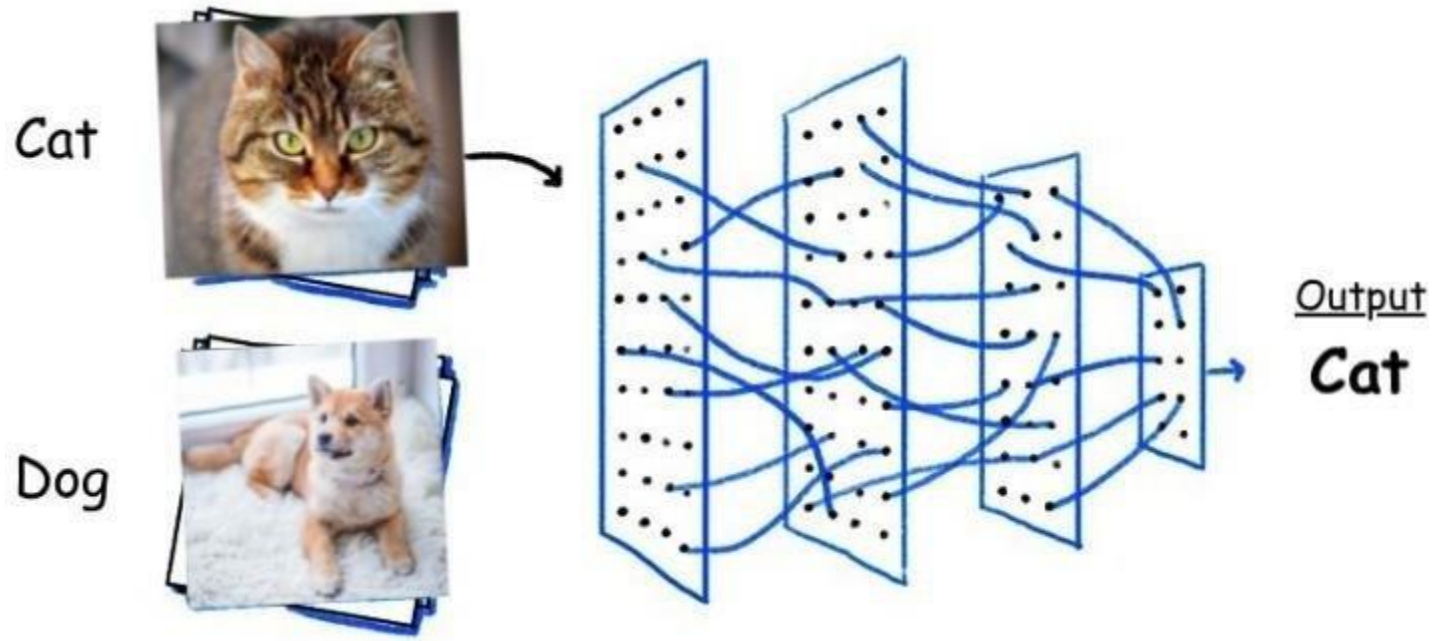
Reduce dimensionality and summarize features

Classification

Determine image class probabilities

Mission

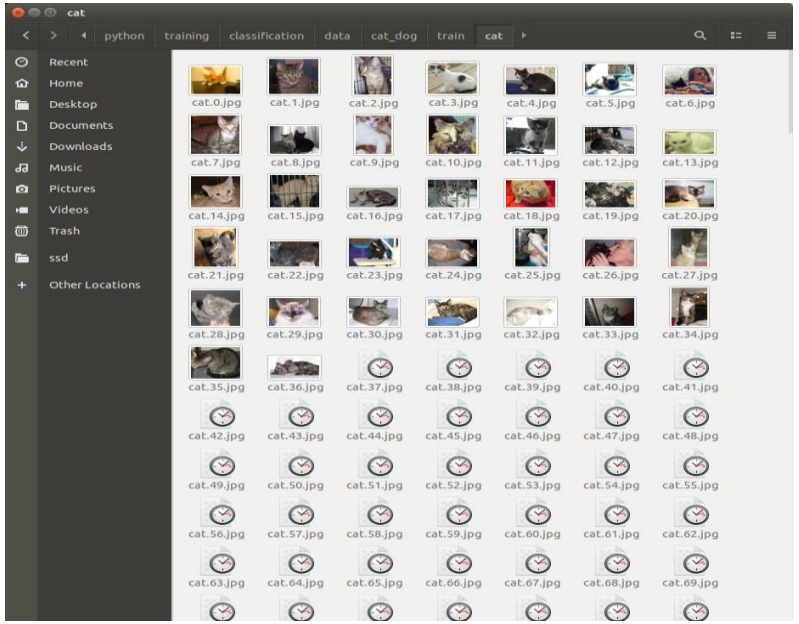
- Train a catdog recognition model using these images, and use it to classify cat and dog.



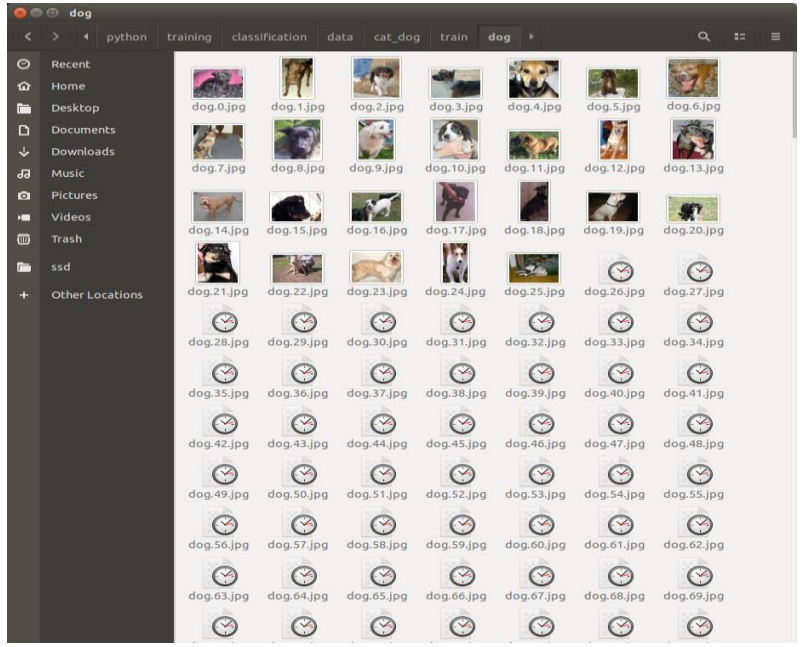
Task1: Train the built-in model --- catdog

- Download the catdog dataset frome onedrive cat_dog (https://connecthkuhk-my.sharepoint.com/:f:/g/personal/u3597499_connect_hku_hk/EtShD5TRW_JOo4Ad_AcBw4kBvvB_8vsBWQ2uNMANLJ014g?e=b2tsWg)
- and put it in ***/home/nvidia/jetson-inference/python/training/classification/data or /home/jetson-inference/python/training/classification/data. .***

cat



dog



Step 1: Train model

- To train the model, we need to open a terminal, and enter:

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

** ***cd ~/jetson-inference/python/training/classification***
Change the directory into ~/jetson-inference/python/training/classification

$ python3 train.py --model-dir=models/cat_dog data/cat_dog --batch-size=1 --workers=1 ***--epochs=2***

***** ***python3 train.py data/cat_dog --model-dir=models/cat_dog --batch-size=1 --workers=1 --epochs=1***
Use the software python3 to run the file "train.py" with the following arguments "*data/cat_dog --model-dir=models/cat_dog --batch-size=1 --workers=1 --epochs=2*"

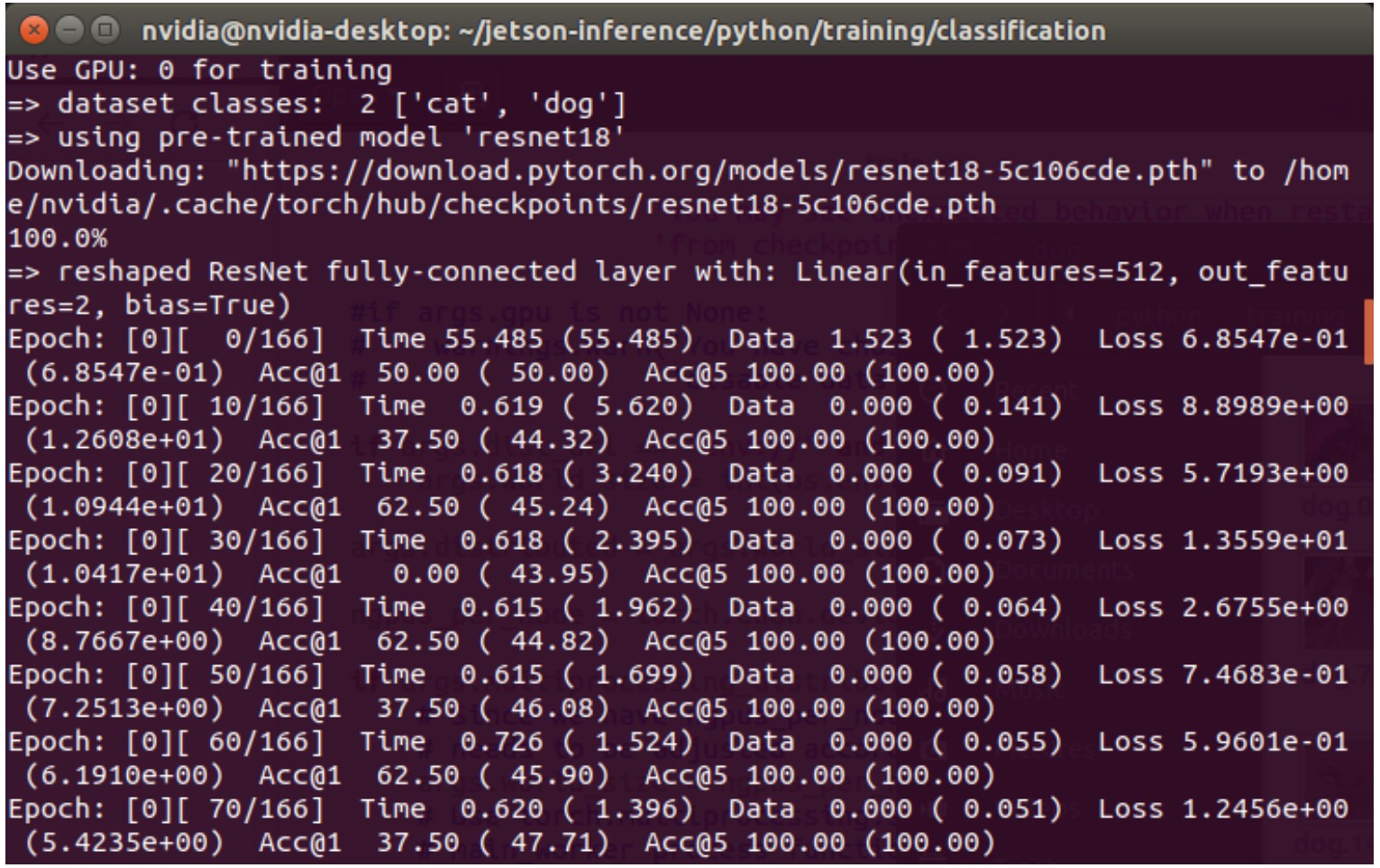
--batch-size=4 --workers=1 --epochs=36

- Batch size is a term used in machine learning and refers to the number of training examples utilized in one iteration.
- Workers : Create multiple threads and load data in advance. num_workers=0 means ONLY the main process will load batches (that can be a bottleneck). num_workers=1 means ONLY one worker (just not the main process) will load data, but it will still be slow.
- An epoch elapses when an entire dataset is passed forward and backward through the neural network exactly one time.

Data set size = Iteration * Batch size （1 Epoch）
Iteration = （Data set size / Batch size） * Epoch

Step 1: Train model

- During training, the terminal will show lots of information.



Acc@1 means that only the true label is the class with **the highest** predicted probability, the prediction is correct.
Acc@5 means that as long as the true label is one of **the five predicted classes with the highest probability**, the prediction is correct.

Example:

If a model predicts probabilities for **an image of a Siamese cat**:

Predicted results:

- Persian Cat: 0.3
- **Siamese Cat: 0.25**
- Tabby Cat: 0.2
- Dog: 0.15
- Bird: 0.1

==========

- Acc@1 would count this as incorrect (Persian Cat ≠ Siamese Cat)
- Acc@5 would count this as correct (Siamese Cat is in top 5)

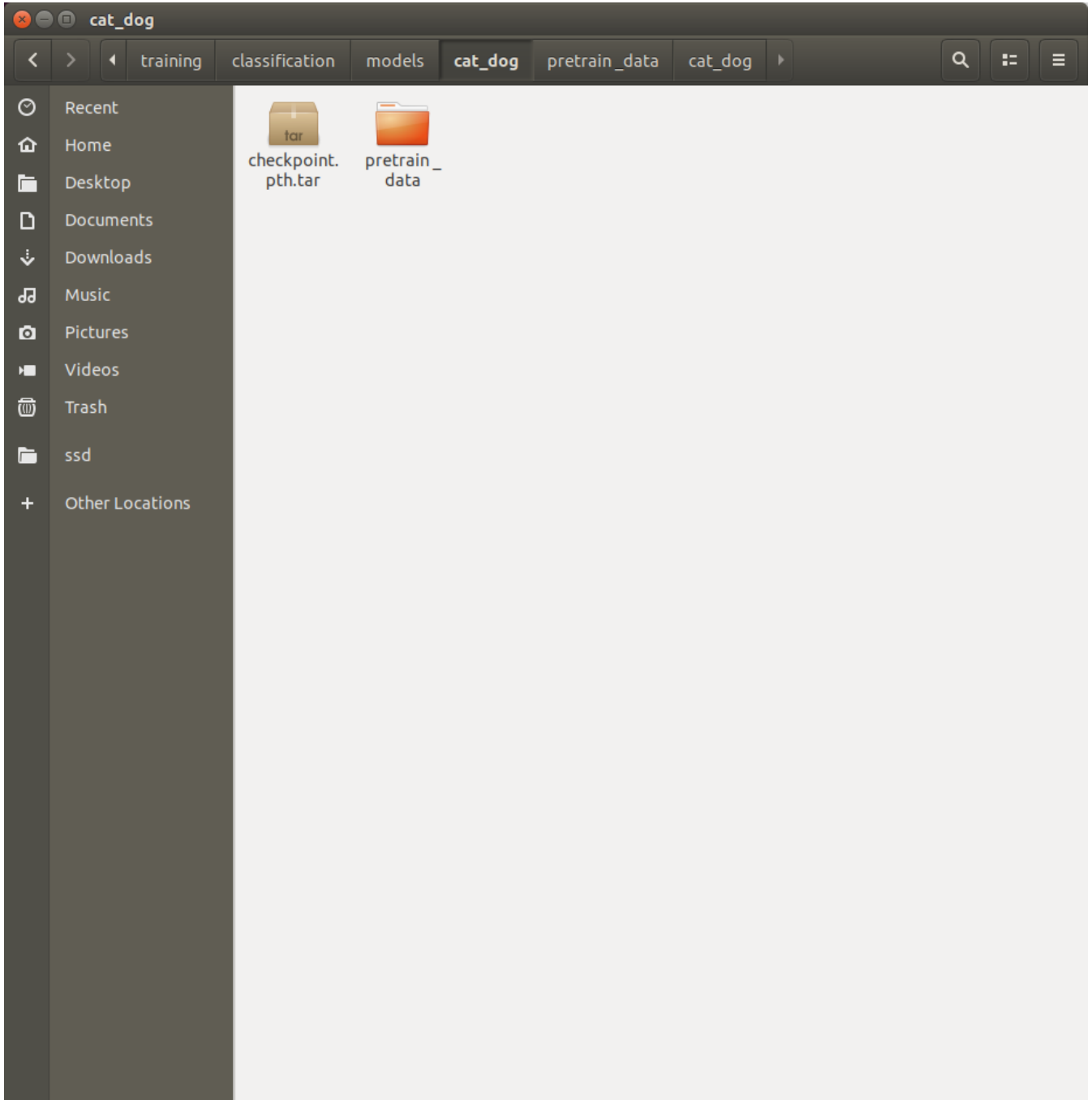
How can we get the loss value?

Where is the loss definition?

Step 2: Convert model

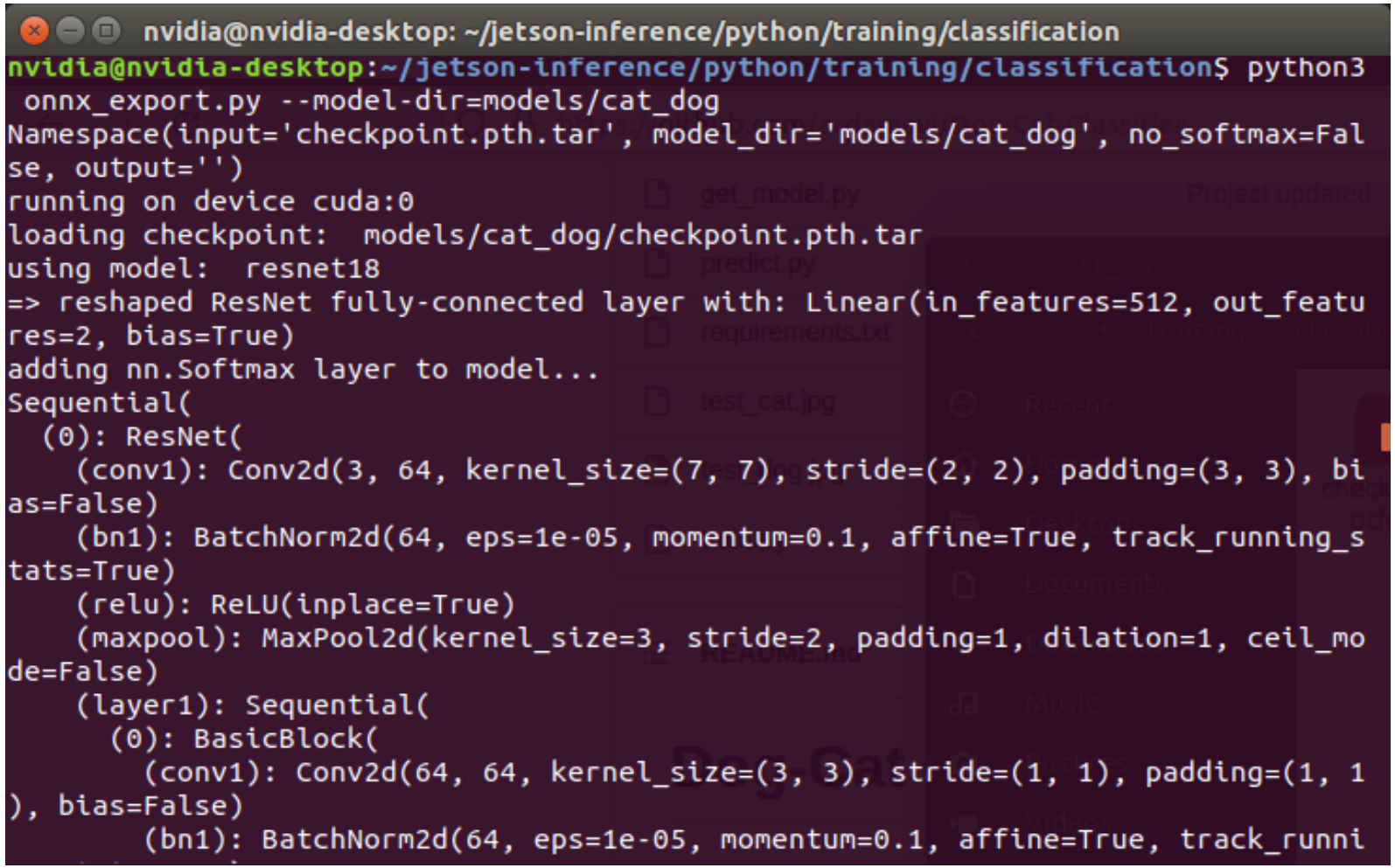
After training, the model folder will be:

In ***/home/jetson-inference/python/training/classification/models/cat_dog/***

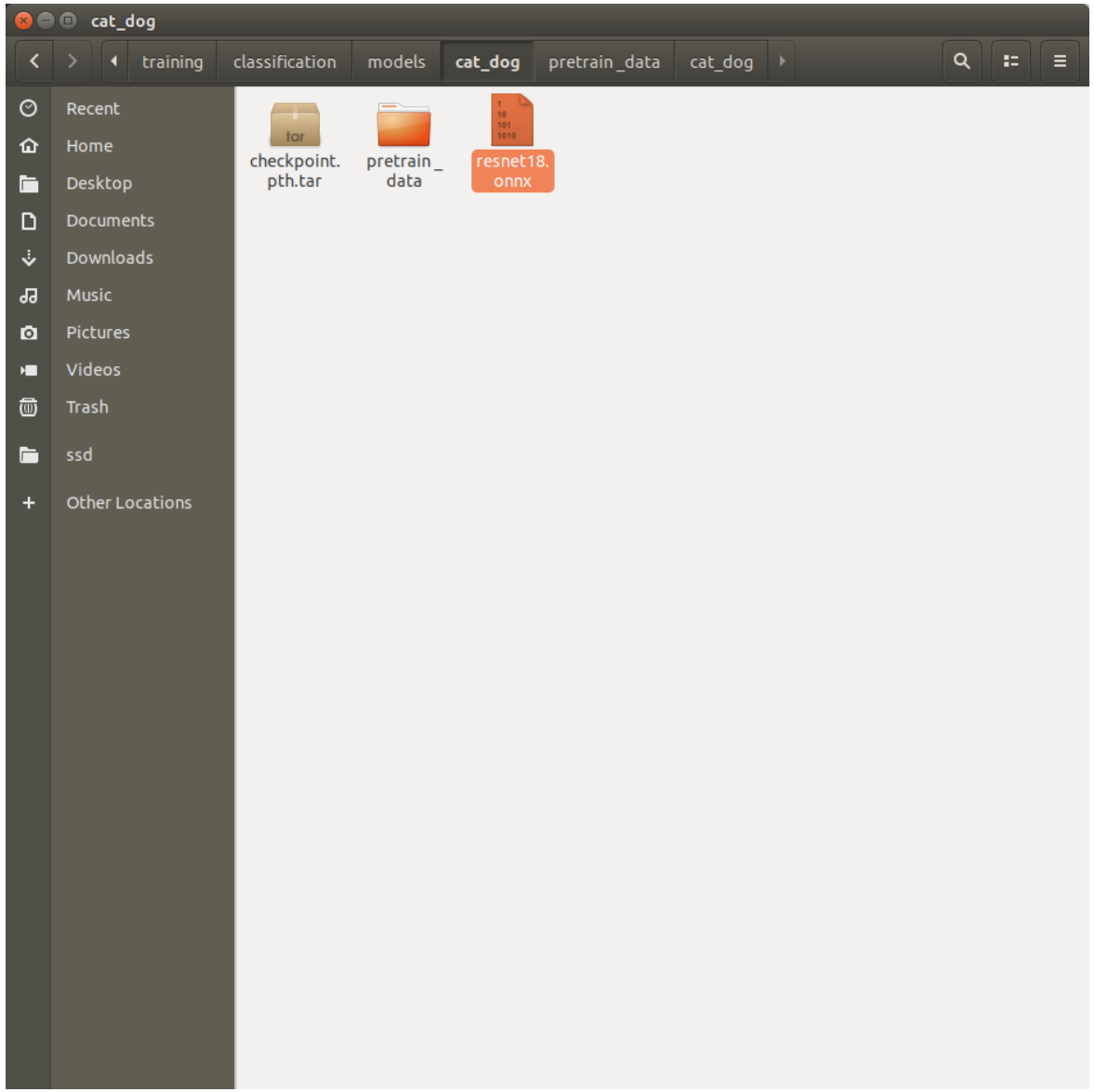


Then, we need to convert the model using:

*$ python3 onnx_export.py --model-dir=models/cat_dog --input=*checkpoint.pth.tar



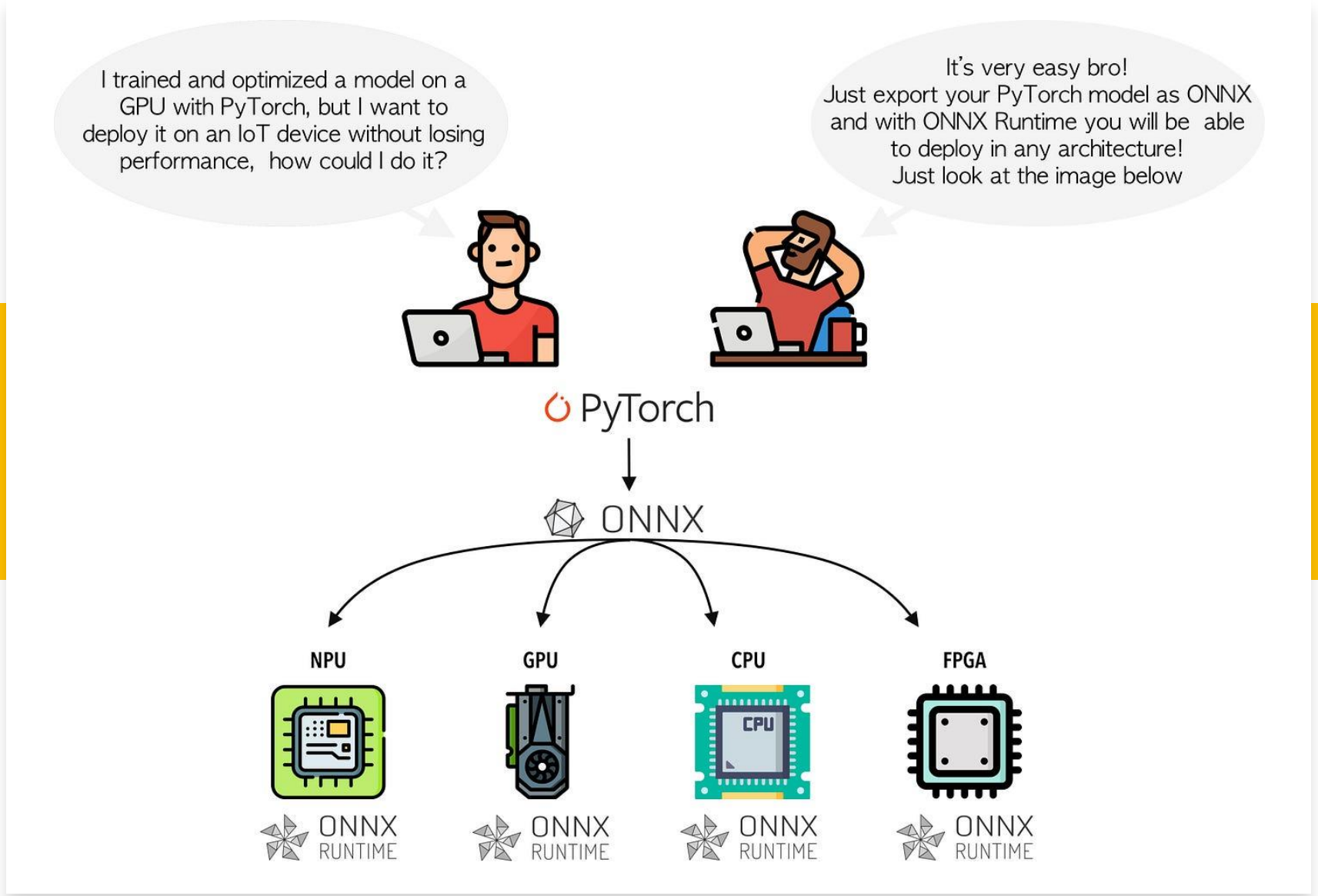
When you find a file with .onnx in the folder, it succeeds!



I trained and optimized a model on a GPU with PyTorch, but I want to deploy it on an IoT device without losing performance, how could I do it?

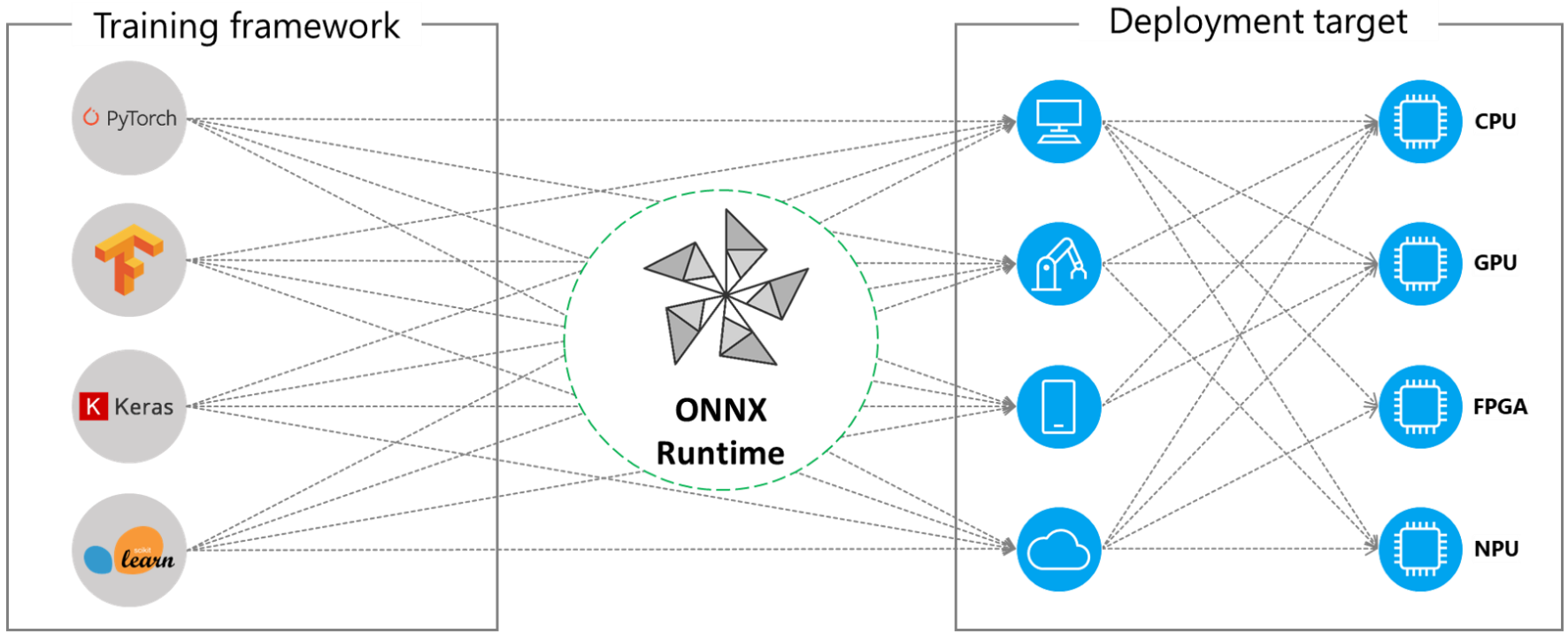
One training and multiple deployment?

Use ONNX


I trained and optimized a model on a GPU with PyTorch, but I want to deploy it on an IoT device without losing performance, how could I do it?
It's very easy bro!
Just export your PyTorch model as ONNX and with ONNX Runtime you will be able to deploy in any architecture!
Just look at the image below
PyTorch
ONNX
NPU
GPU
CPU
FPGA
CPU
ONNX RUNTIME
ONNX RUNTIME
ONNX RUNTIME
ONNX RUNTIME

Open Neural Network Exchange (ONNX)

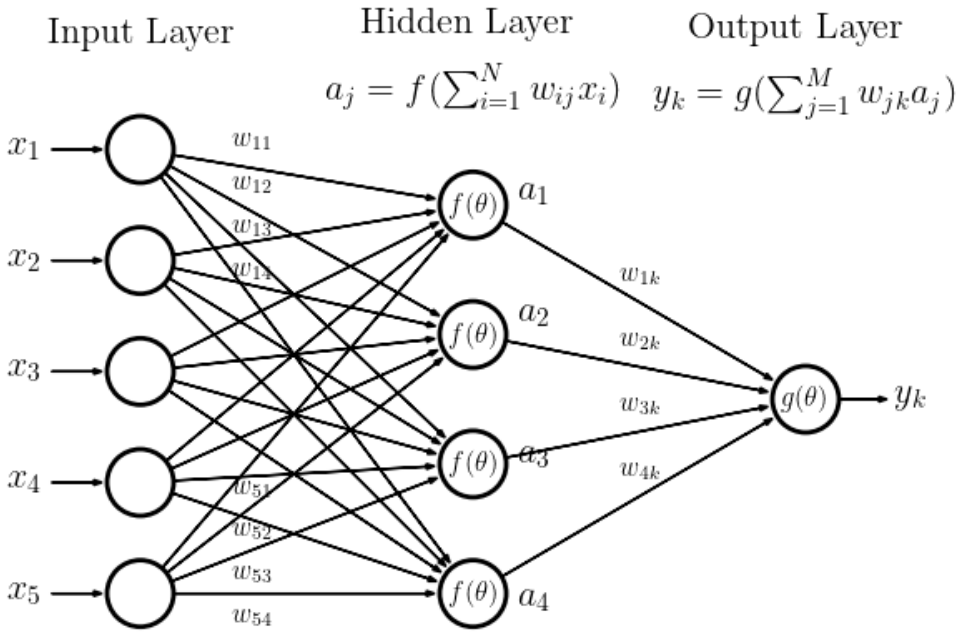
- An open format proposed by Microsoft and Facebook to represent deep learning models.
- Openness means that ONNX defines a set of standard formats that are independent of the environment and platform to enhance the interactivity of various AI models.
- i.e. no matter which training framework you use to train the model (such as TensorFlow/Pytorch/OneFlow/Paddle), after training, you can uniformly convert the models of these frameworks into a unified format such as ONNX for storage.



ONNX file

- ONNX file not only stores the weights of the neural network model, but also stores the structural information of the model, the input and output of each layer in the network, and some other auxiliary information.

  - weights of the neural network model
  - structural information of the model
  - input and output of each layer in network
  - and some other auxiliary information.



Step 3: Inference model with image

- Now you get a model runnable! In the next step, we need to use it to make prediction on different images.
- Take a cat for example:

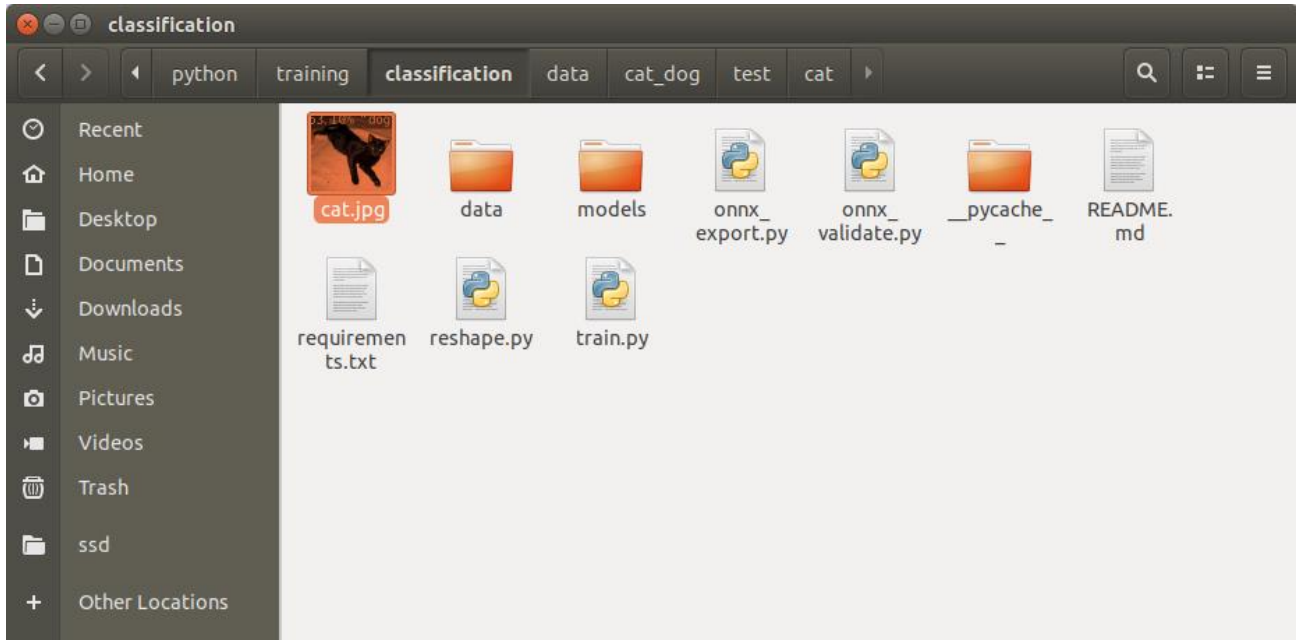
$ cd ~/jetson-inference/python/training/classification/

$ DATASET=data/cat_dog

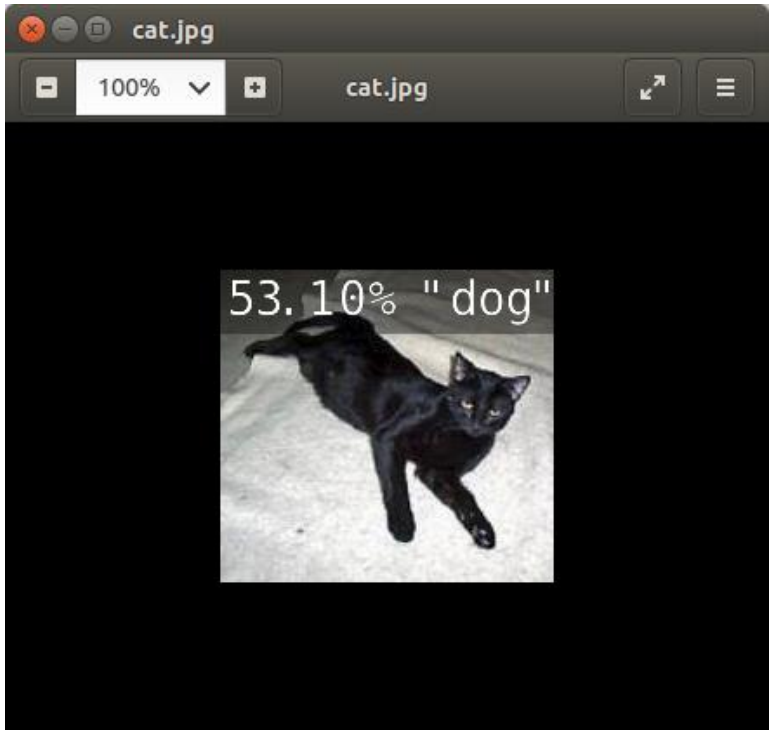
*$ /home/**nvidia**/jetson-inference/build/aarch64/bin/imagenet --model=models/cat_dog/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=$DATASET/labels.txt $DATASET/test/cat/0001.jpg cat.jpg*

Step 3: Inference model with image

- We can see the model prediction using your trained model!



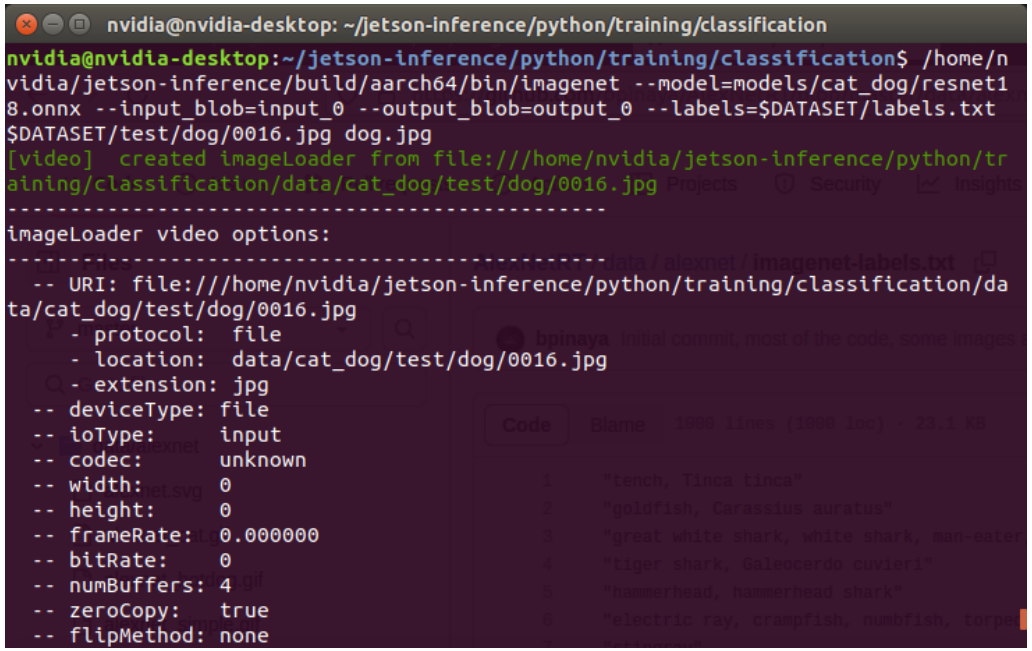
Original image

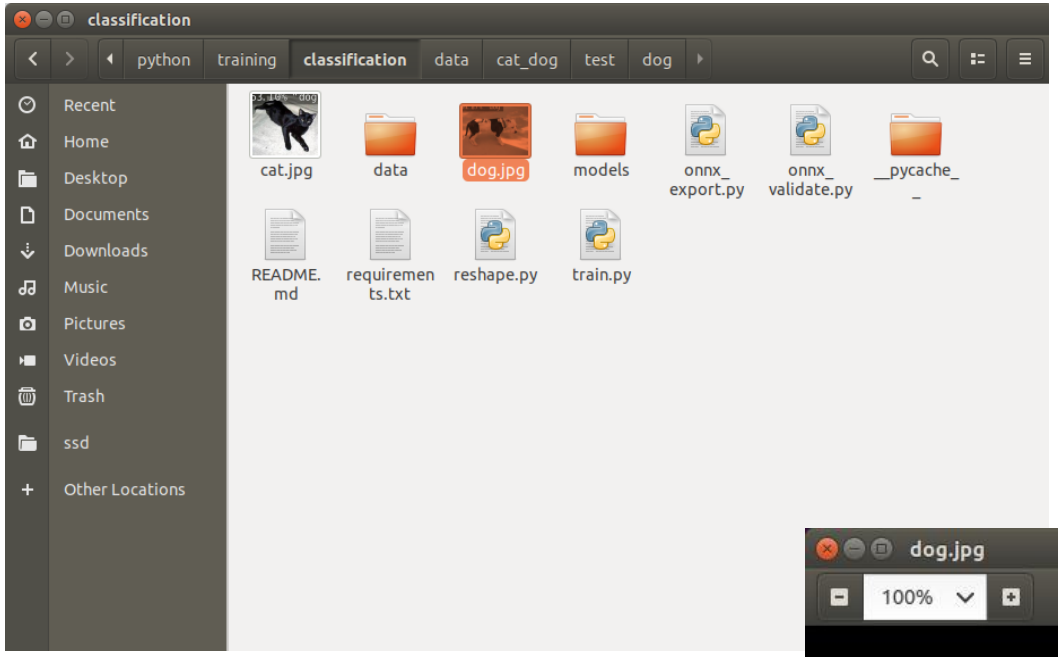


The prediction made by the model

Step 3: Inference model with image

- How about recognizing a dog?



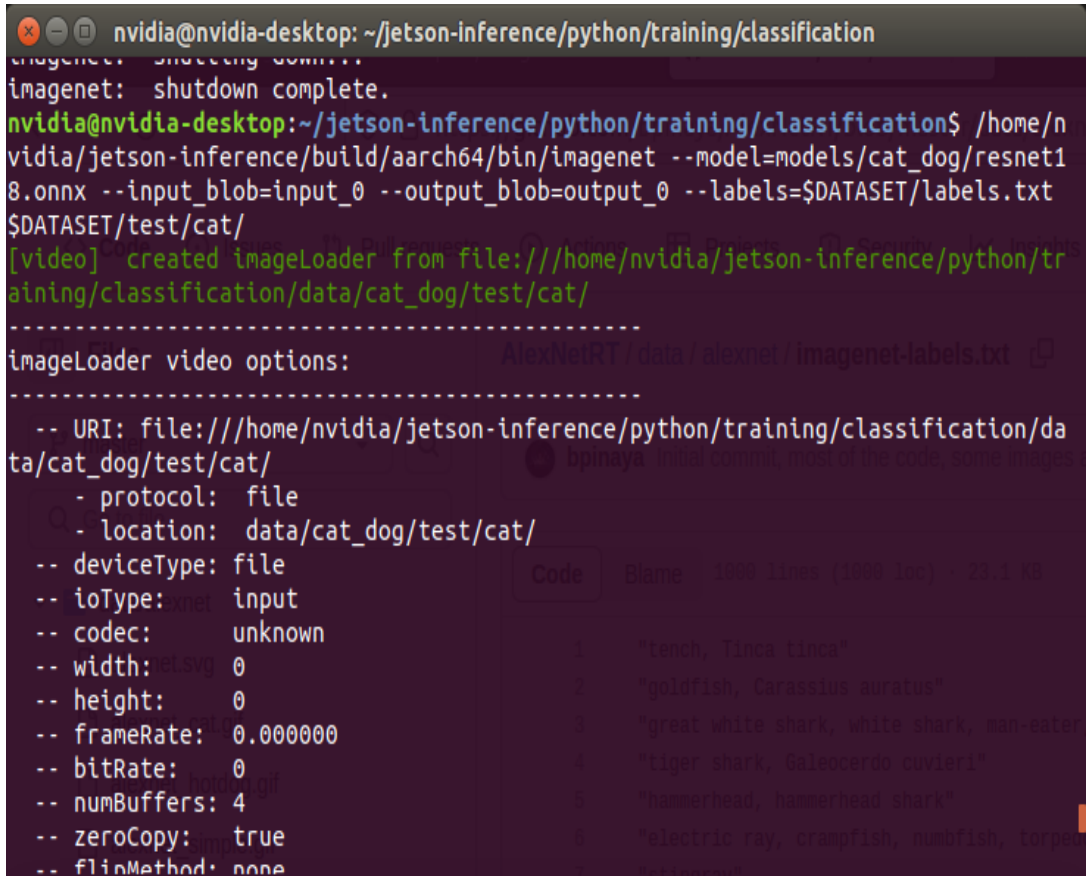


$ cd ~/jetson-inference/python/training/classification

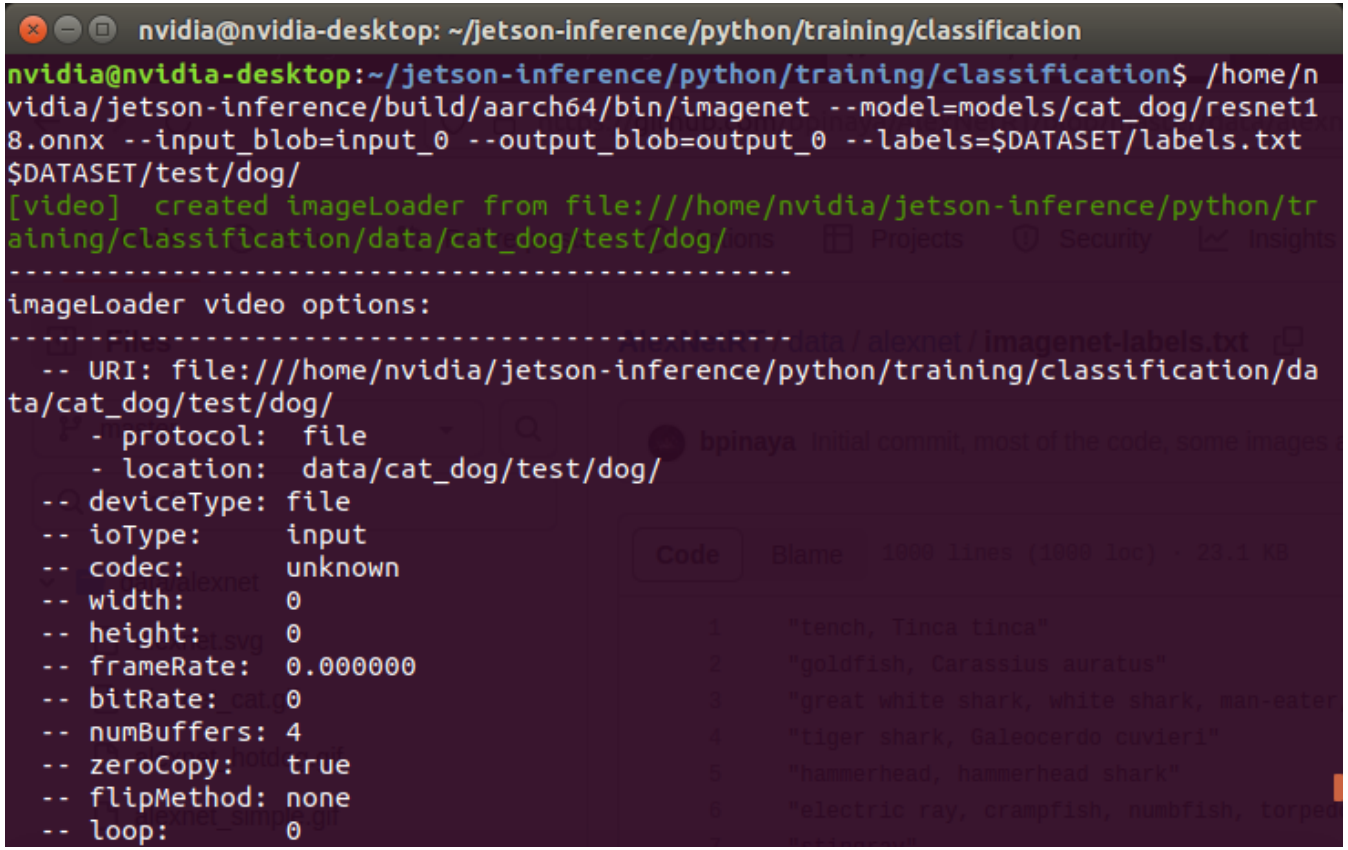
$ /home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model=models/cat_dog/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=$DATASET/labels.txt $DATASET/test/dog/0016.jpg dog.jpg

Step 3: Inference model with image

To tired to enter commands one by one? Try directly run a whole dataset.



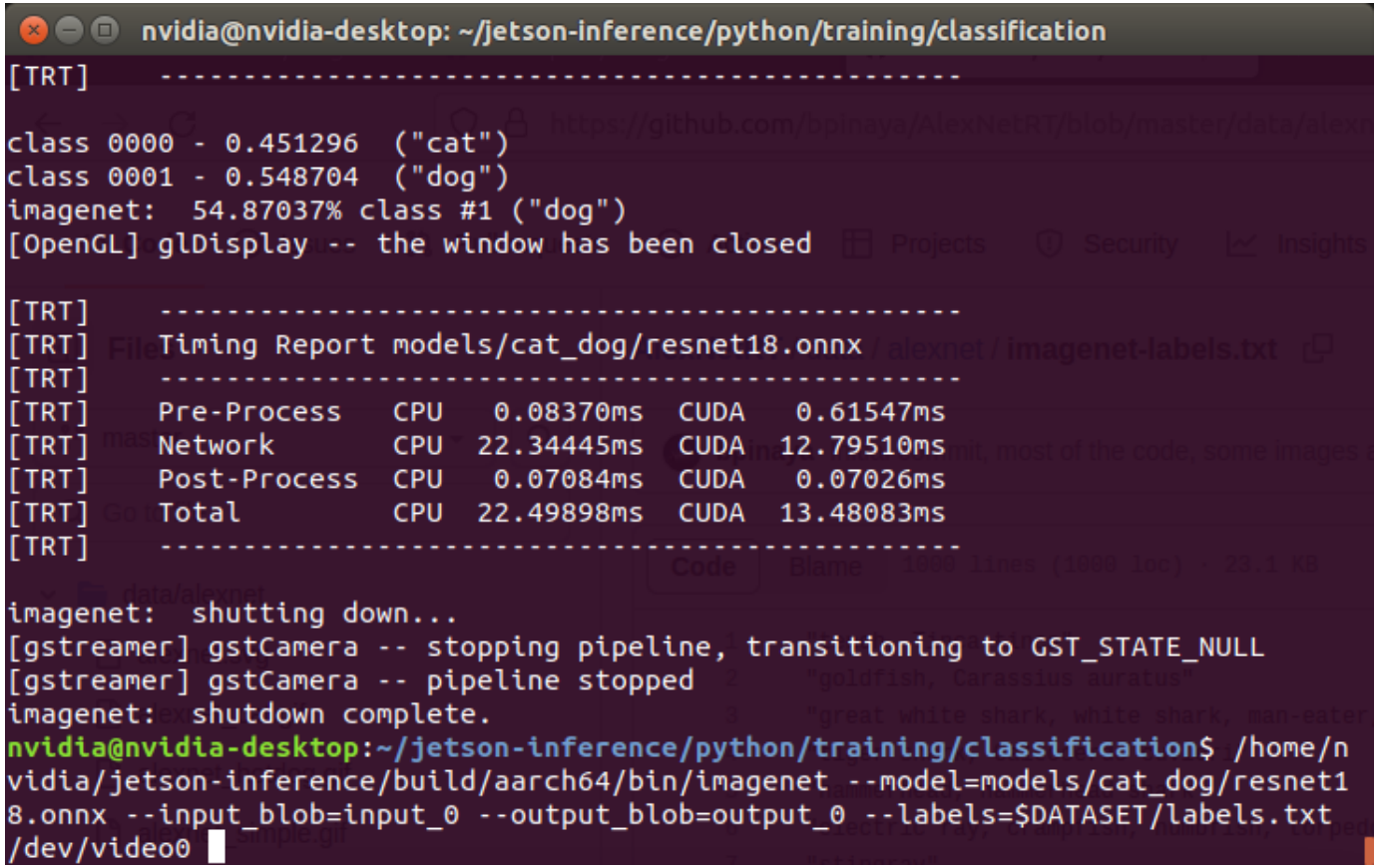
$ /home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model=models/cat_dog/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=$DATASET/labels.txt $DATASET/test/dog/



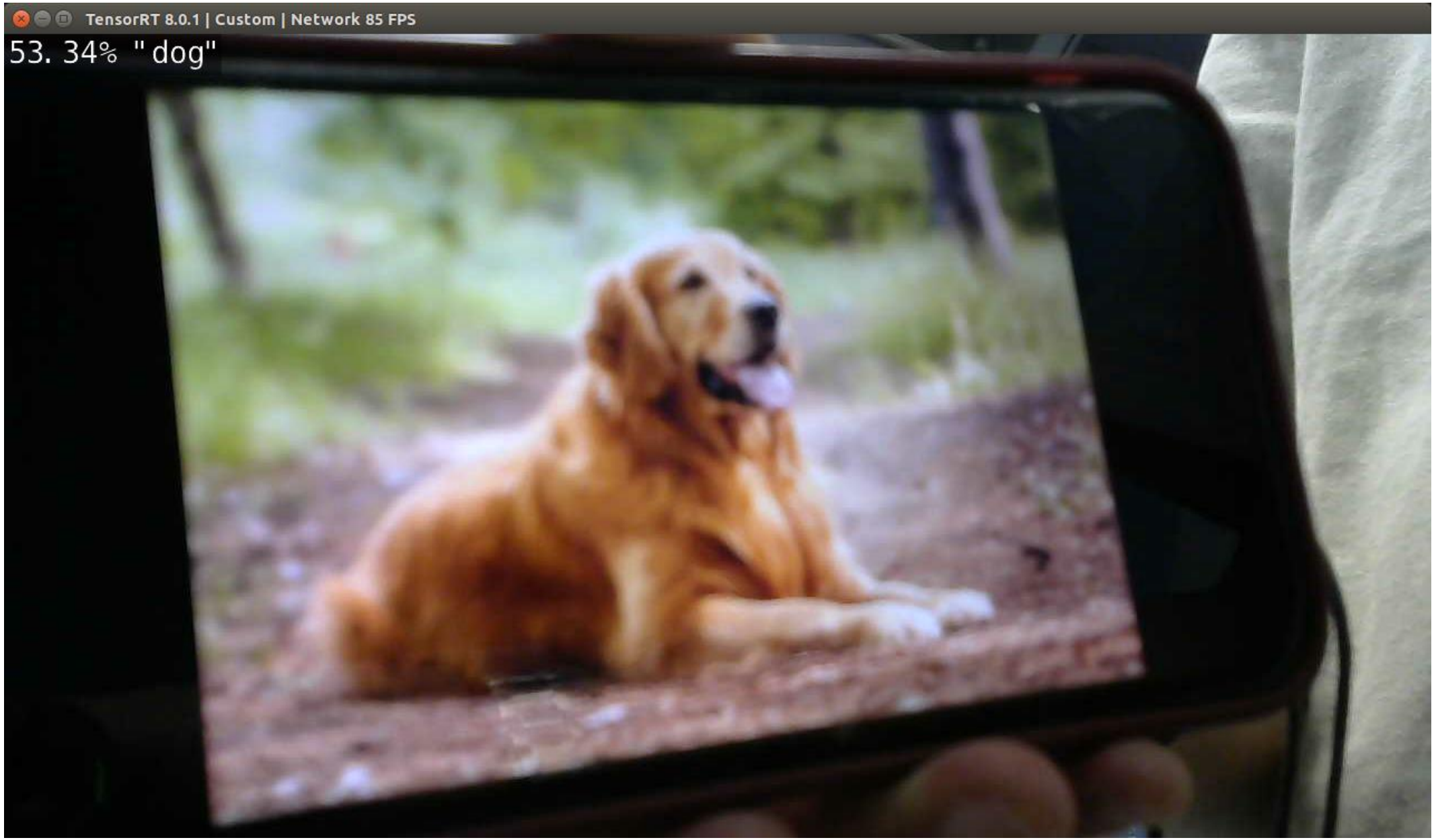
$ /home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model=models/cat_dog/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=$DATASET/labels.txt $DATASET/test/cat/

Step 4: Inference with video

- Apart from image, you can also use this model through webcam.

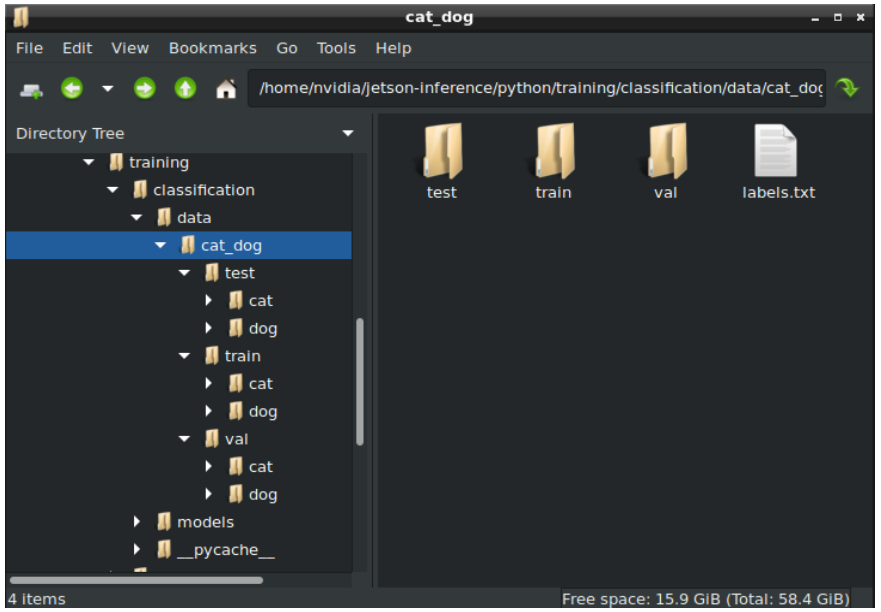


$ /home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model=models/cat_dog/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=$DATASET/labels.txt /dev/video0

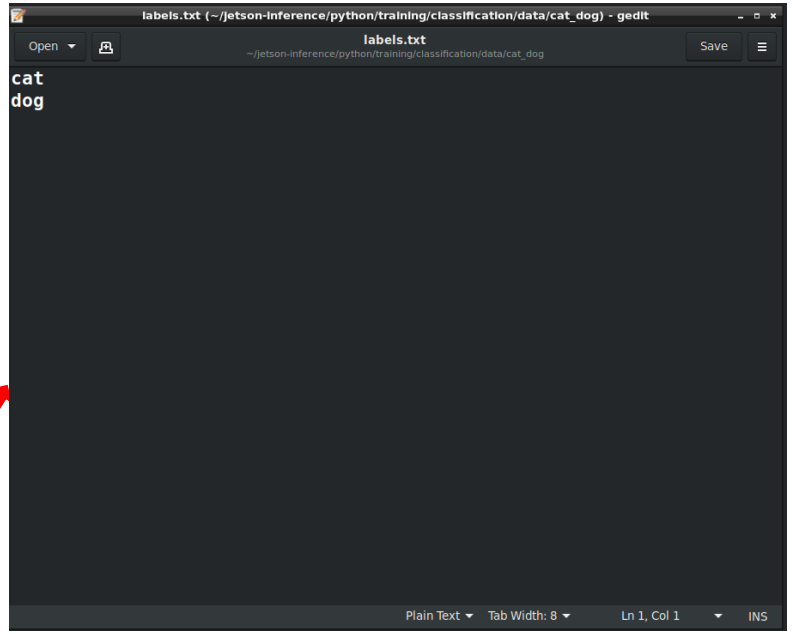

TensorRT 8.0.1 | Custom | Network 85 FPS
53. 34% "dog"

Task 2: Train our own model

- Now you are familiar with how to train classification model.
- Before training, we should collect the dataset, and organize them similar to the catdog dataset.



--data/catdog
------train
----------cat
----------dog
------test
----------cat
----------dog
------validation
----------cat
----------dog
------labels.txt



Training with your own dataset

- A group of dataset
- Two important parameter should be defined

(1) Where to store your data (***YourDataPath***)

(2) Where to store your model (***YourModelPath***)

- Following the similar steps to finish model training, conversion and inference.

Dataset collection

Now, you can download the images you want to classify from google.

Remember the image between train, validation and test is around 7:2:1

After finishing the dataset, download check_dataset.py and run it to verify whether the dataset structure is valid.

Train our own model: Training

- After getting the dataset, similarly, we need to open a terminal, and:

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

*$ python3 train.py --model-dir=**YourModelPath** **YourDataPath** --batch-size=1 --workers=1 **--epochs=1***

****PREVIOUS EXAMPLE:*
$ python3 train.py --model-dir=models/cat_dog data/cat_dog --batch-size=4 --workers=1 --epochs=36

Please keep the path of ***YourModelPath*** and ***YourDataPath*** in mind! They are necessary in the following

Train our own model: Converting and Inference

Convert

*$ python3 onnx_export.py --model-dir=**YourModelPath***

Inference with image

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

*$ DATASET=**YourDataPath***

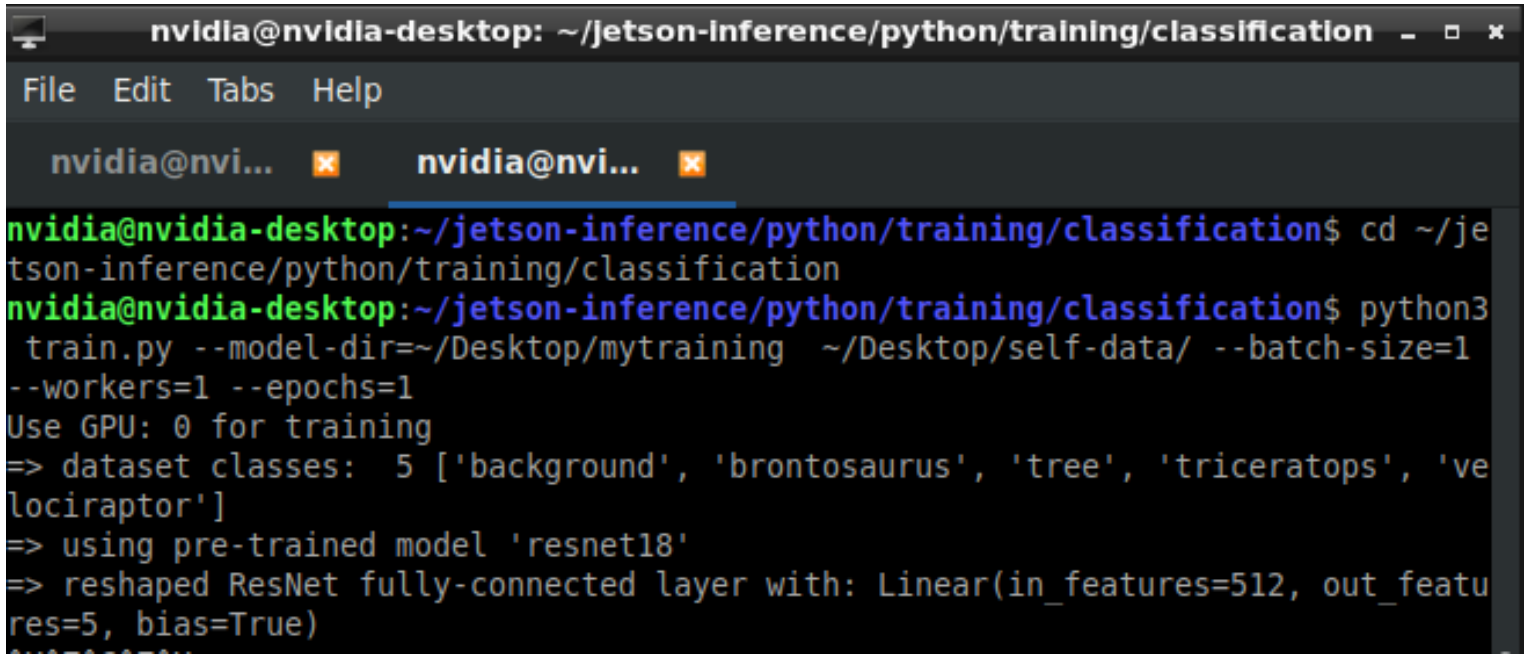
*$ /home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model=**YourModelPath**/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=$DATASET/labels.txt* ***TheImageYouWantToProcess***

****Example:**

$ /home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model=models/cat_dog/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=$DATASET/labels.txt $DATASET/test/dog/0016.jpg dog.jpg

- Training with your own data

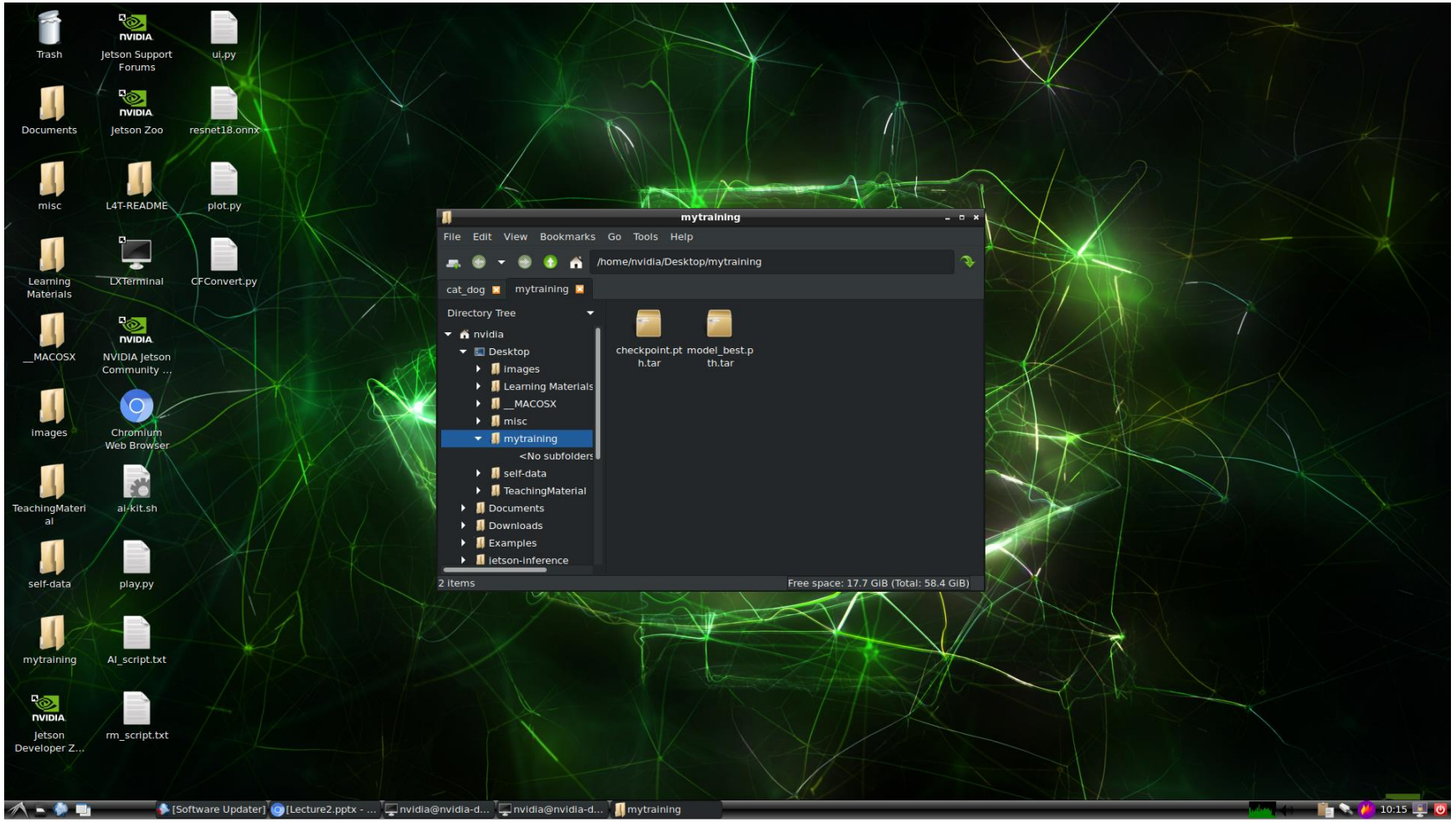
After you've captured your own dataset, it's time to train your own model



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

$ python3 train.py --model-dir=~/Desktop/mytraining ~/Desktop/self-data/ --batch-size=1 --workers=1 –epoch=1

- Training with your own data

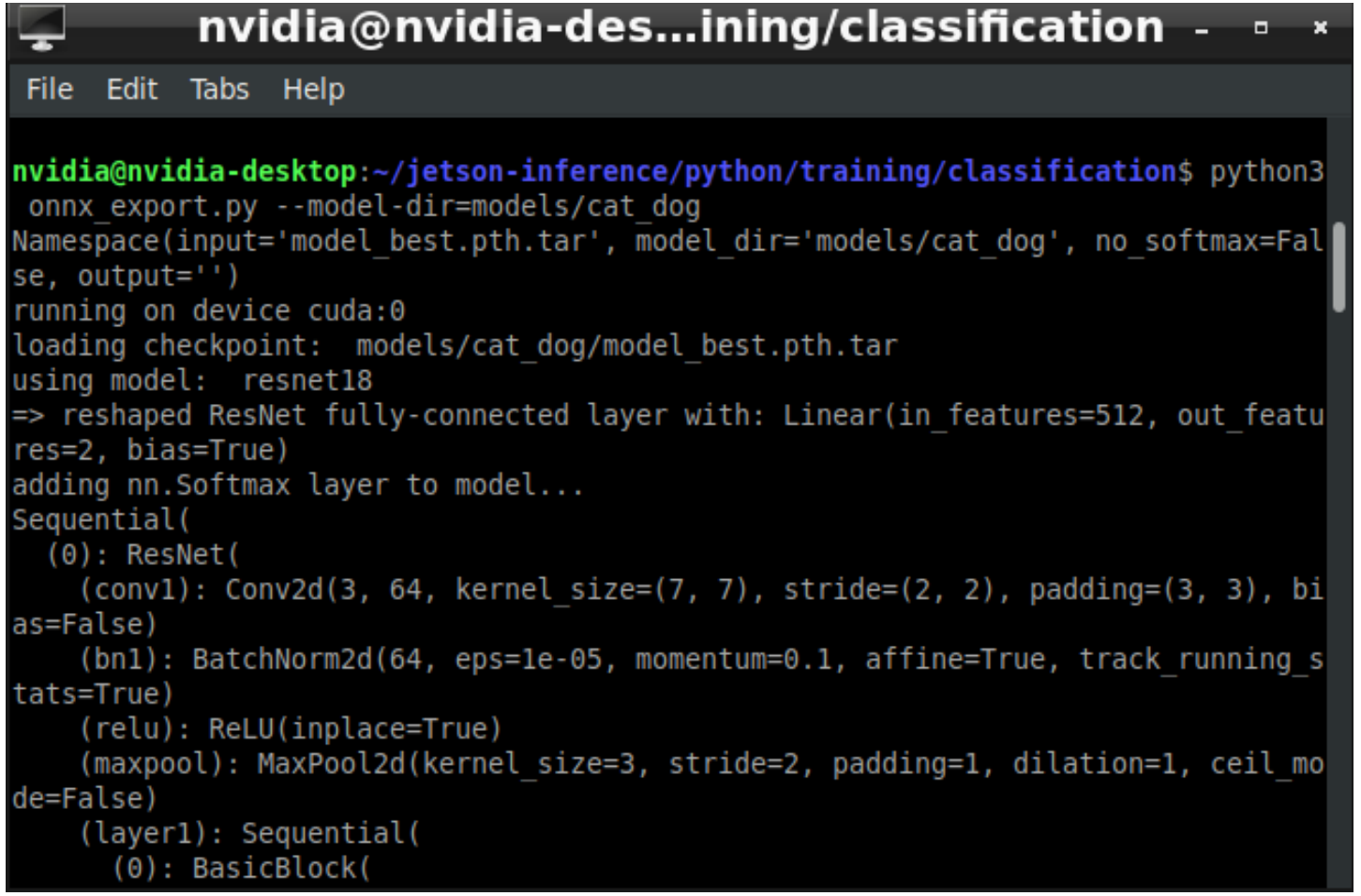


After finishing your training, you would find this two files in "mytraining"

- Converting the Model to ONNX

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

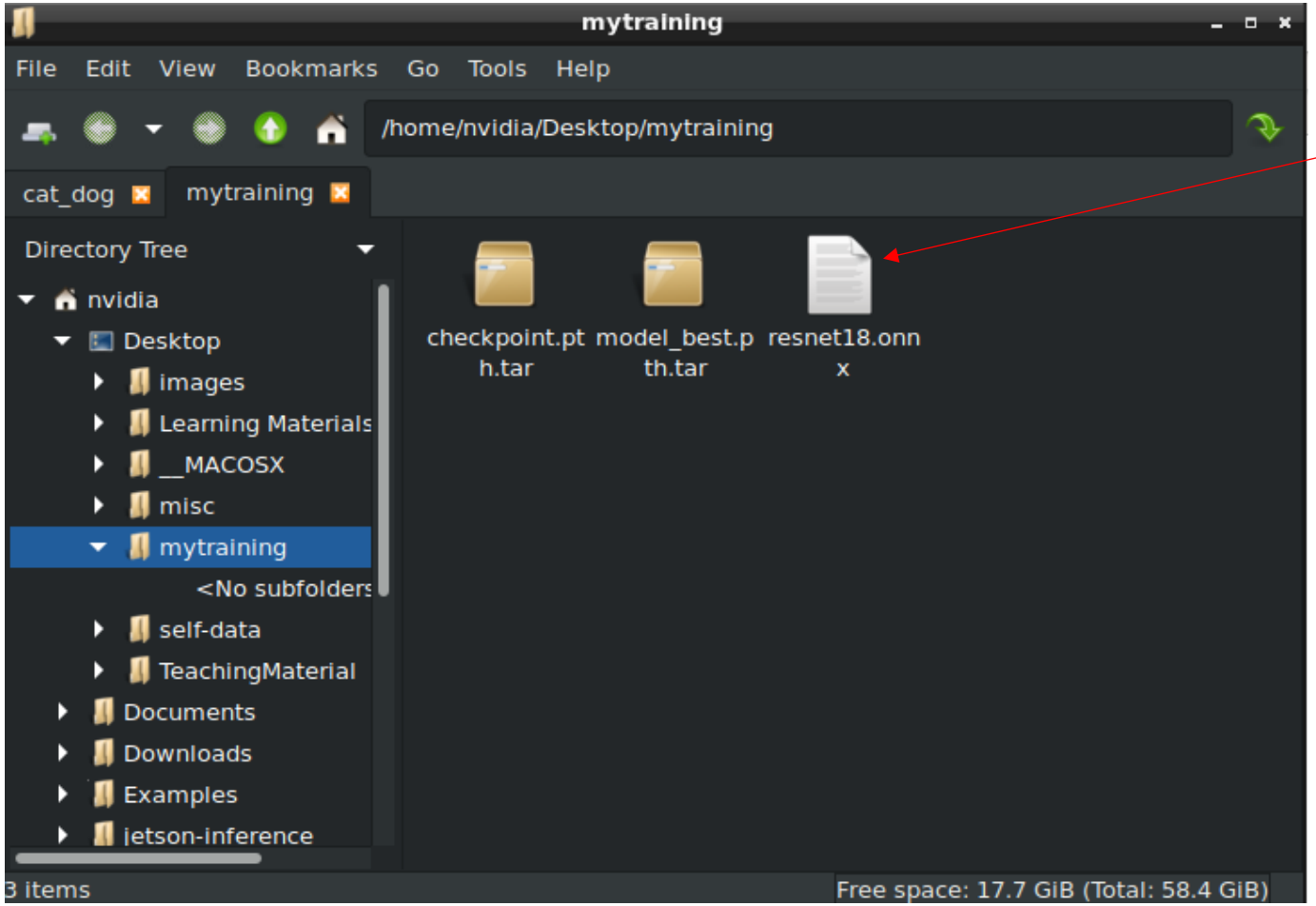
$ python3 onnx_export.py --model-dir=~/Desktop/mytraining



- Converting the Model to ONNX

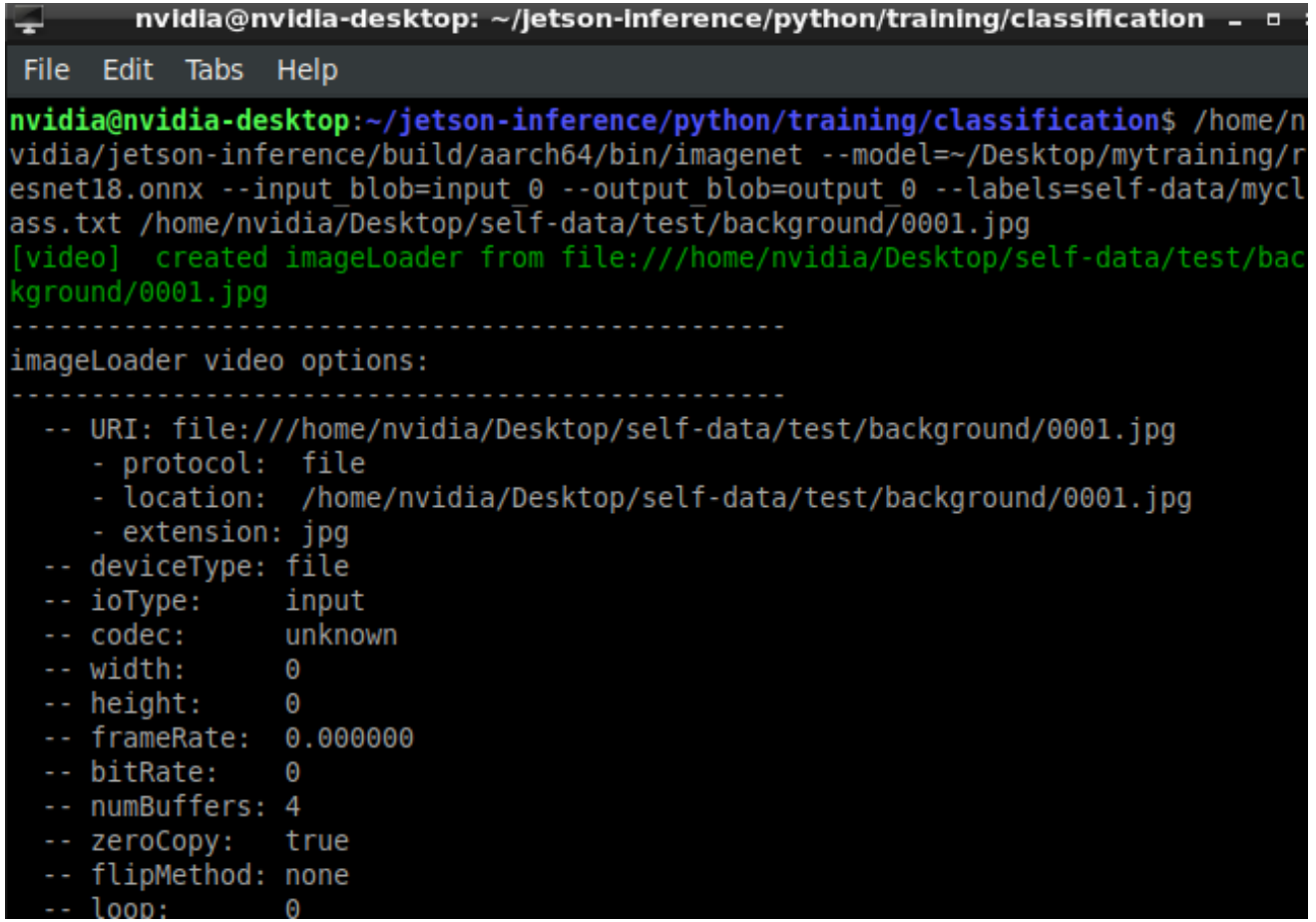
This will create a model called resnet18.onnx under:

~/Desktop/mytraining



- Processing Images with TensorRT

$ /home/nvidia/jetson-inference/build/aarch64/bin/imagenet --model=~/Desktop/mytraining/resnet18.onnx --input_blob=input_0 --output_blob=output_0 --labels=self-data/myclass.txt ***yourOwenPicturePath***



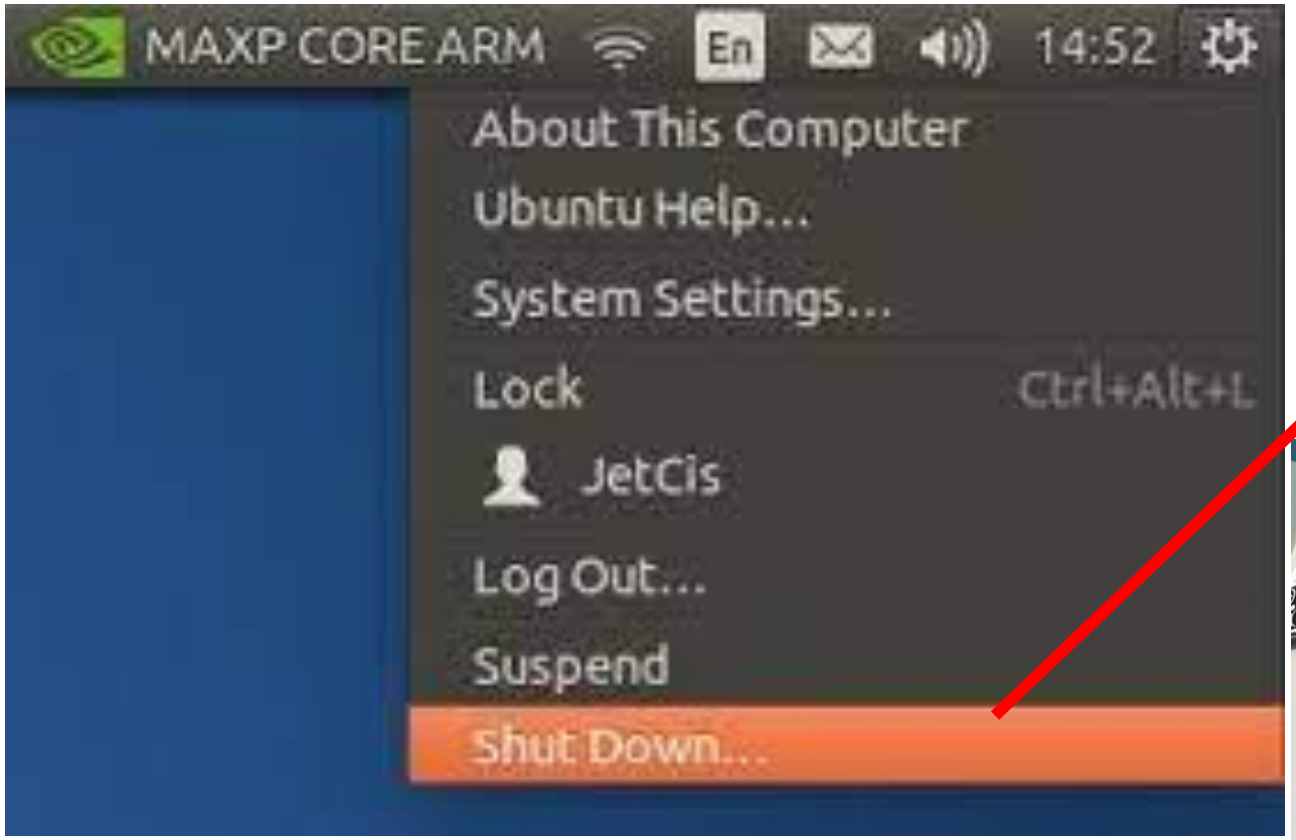
Assignment 2

Train you own model

Collect your own dataset for more than three classes (new class can not same with your teammate&can not be cat/dag)/ Train a new model/ Output the prediction image

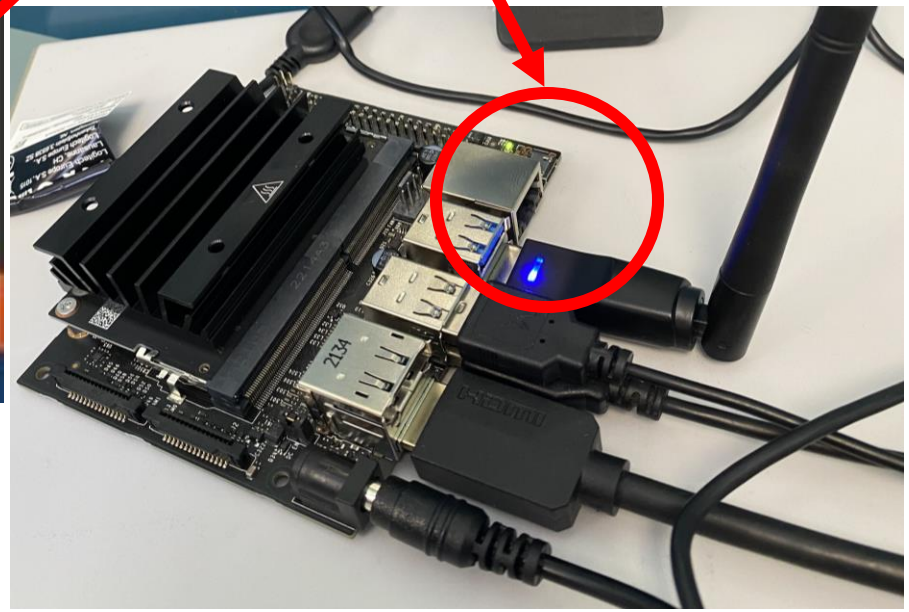
Upload the dataset (OneDrive link), result (tested image with result) and onnx model to the moodle

Deadline: 24/02/2025 **11:59pm**



Click on the **Shut Down button**, shut down the device like computer

After the **light turning off**, **plug out** the power supply



Pack it well for next time

Pack it well for next time