# アジェンダ

- 物体検出プログラムの中身を見る
  - Pythonスクリプトの編集
- 独自の検出器を構築する
  - 独自の学習データ構築を学ぶ
- グループのテーマを考える
  - アイデア出しを開始

# ナビゲーションのためのUnixコマンド

- pwd（現在の作業ディレクトリ）– ここはどこ？
- cd（ディレクトリの変更）– 指定されたディレクトリに移動
- ls （リスト）– 現在のディレクトリに何があるか見る

迷子になった？

- cd と入力すれば，自動的にホーム（デフォルト）に帰る

- ..（親ディレクトリ）–「cd ..」1つ前のディレクトリに戻る

# Jetsonを用いた推論

jetson hello ai で検索


## Deploying Deep Learning

Welcome to our instructional guide for inference and realtime vision DNN library for **NVIDIA Jetson** devices. This project uses **TensorRT** to run optimized networks on GPUs from C++ or Python, and PyTorch for training models.

Supported DNN vision primitives include `imageNet` for image classification, `detectNet` for object detection, `segNet` for semantic segmentation, `poseNet` for pose estimation, and `actionNet` for action recognition. Examples are provided for streaming from live camera feeds, making webapps with WebRTC, and support for ROS/ROS2.

https://github.com/dusty-nv/jetson-inference

# Dockerを起動（1/2）

## Hello AI World

Hello AI World can be run completely onboard your Jetson, including live inferencing with TensorRT and transfer learning with PyTorch. For installation instructions, see System Setup. It's then recommended to start with the Inference section to familiarize yourself with the concepts, before diving into Training your own models.

### System Setup

- Setting up Jetson with JetPack
- Running the Docker Container
- Building the Project from Source

クリック

# Dockerを起動（2/2）

## Launching the Container

Due to various mounts and devices needed to run the container, it's recommended to use the `docker/run.sh` script to run the container:

```
$ git clone --recursive --depth=1 https://github.com/dusty-nv/jetson-inference
$ cd jetson-inference
$ docker/run.sh
```

ターミナル（Terminal）を起動してコマンドを入力

# 物体検出を実行（1/2）

Inference

- Image Classification
  - Using the ImageNet Program on Jetson
  - Coding Your Own Image Recognition Program (Python)
  - Coding Your Own Image Recognition Program (C++)
  - Running the Live Camera Recognition Demo
  - Multi-Label Classification for Image Tagging
- Object Detection
  - Detecting Objects from Images
  - Running the Live Camera Detection Demo
  - Coding Your Own Object Detection Program
  - Using TAO Detection Models
  - Object Tracking on Video

# 物体検出を実行（2/2）

Python

```
$ ./detectnet.py csi://0                  # MIPI CSI camera
$ ./detectnet.py /dev/video0              # V4L2 camera
$ ./detectnet.py /dev/video0 output.mp4   # save to video file
```

1. cdコマンドを入力して，detectnet.py が存在する jetson-inference/python/examples/ に移動
2. Dockerが起動しているターミナル（Terminal）上でコマンドを入力

lsコマンドを入力して「detectnet.py」がフォルダに存在しないと実行できない

# 物体検出のPythonスクリプト作成（1/2）

**Inference**

- Image Classification
  - Using the ImageNet Program on Jetson
  - Coding Your Own Image Recognition Program (Python)
  - Coding Your Own Image Recognition Program (C++)
  - Running the Live Camera Recognition Demo
  - Multi-Label Classification for Image Tagging
- Object Detection
  - Detecting Objects from Images
  - Running the Live Camera Detection Demo
  - Coding Your Own Object Detection Program
  - Using TAO Detection Models
  - Object Tracking on Video

# 物体検出のPythonスクリプト作成（2/2）

1. jetson-inference/python/examples/my-detection.py を jetson-inference/data/ にコピー
2. Visual studio codeで jetson-inference/data/my-detection.py を開く

```python
from jetson_inference import detectNet
from jetson_utils import videoSource, videoOutput

net = detectNet("ssd-mobilenet-v2", threshold=0.5)
camera = videoSource("csi://0")      # '/dev/video0' for V4L2
display = videoOutput("display://0") # 'my_video.mp4' for file

while display.IsStreaming():
    img = camera.Capture()

    if img is None: # capture timeout
        continue

    detections = net.Detect(img)

    display.Render(img)
    display.SetStatus("Object Detection | Network {:.0f} FPS".format(net.GetNetworkFPS()))
```

Importing Modules

Loading the Detection Model

Opening the Camera Stream

Display Loop

Camera Capture

Detecting Objects

Rendering

camera = videoSource(“/dev/video”)

print(detections)

# アジェンダ

- 物体検出プログラムの中身を見る
  - 第2回の復習
  - Pythonコードを編集
- 独自の検出器を構築する
  - 独自の学習データを構築
- グループのテーマを考える
  - アイデア出しを開始

# 独自の学習データを構築

## Training

- Transfer Learning with PyTorch
- Classification/Recognition (ResNet-18)
  - Re-training on the Cat/Dog Dataset
  - Re-training on the PlantCLEF Dataset
  - Collecting your own Classification Datasets
- Object Detection (SSD-Mobilenet)
  - Re-training SSD-Mobilenet
  - Collecting your own Detection Datasets

# 独自のデータセットを構築(1/3)

1. GUIのエクスプローラーで jetson-inference/python/training/detection/ssd/data に移動
2. エクスプローラー内を右クリックして「ターミナル（端末）で開く」をクリック
3. 開いたターミナルで touch labels.txt を入力
4. GUIのエクスプローラーの中に labels.txt が作成されたことを確認
5. ダブルクリックでlabels.txt を開き適当なクラス名を英語で入力して保存・閉じる

## Creating the Label File

Under `jetson-inference/python/training/detection/ssd/data`, create an empty directory for storing your dataset and a text file that will define the class labels (usually called `labels.txt`). The label file contains one class label per line, for example:

```
Water
Nalgene
Coke
Diet Coke
Ginger ale
```

# 独自のデータセットを構築(2/3)

6. カメラキャプチャツールを起動

Dockerが起動していてるターミナル上で入力すること

## Launching the Tool

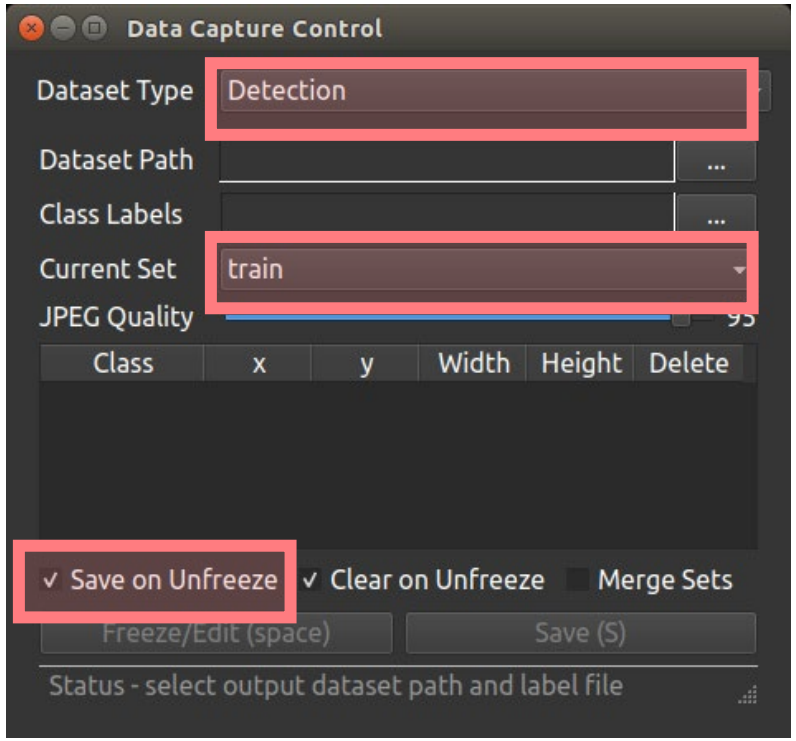
The `camera-capture` tool accepts the same input URI's on the command line that are found on the Camera Streaming and Multimedia page.

Below are some example commands for launching the tool:

```
$ camera-capture csi://0       # using default MIPI CSI camera
$ camera-capture /dev/video0   # using V4L2 camera /dev/video0
```

# 独自のデータセットを構築(3/3)

7. GUIのエクスプローラーで jetson-inference/python/training/detection/ssd/data に移動
8. My_Data （仮名）フォルダを作成
9. データ収集，データアノテーション




チェックを外す

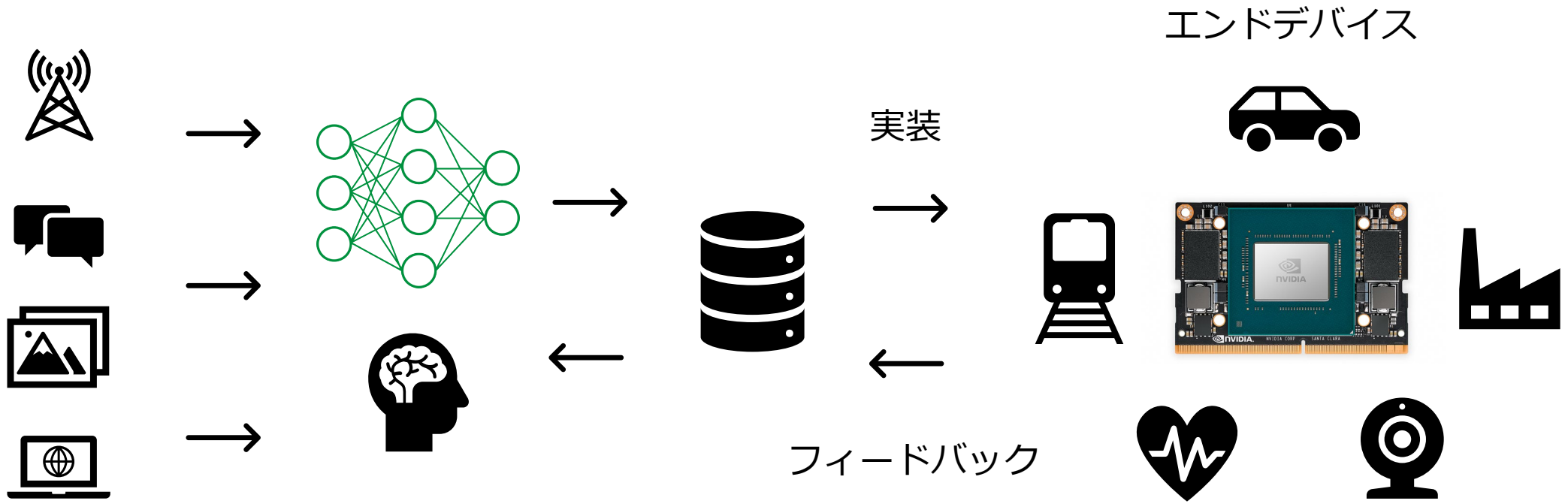
Dataset Path

jetson-inference/python/training/detection/ssd/data/My_Data

Class Labels

jetson-inference/python/training/detection/ssd/data/labels.txt

# 学習はサーバーで実行


エンドデバイス
実装
フィードバック
データ
学習
学習済モデル
推論
クラウド・サーバー（大規模演算）
エッジ（リアルタイム処理）

# アジェンダ

- 物体検出プログラムの中身を見る
  - 第2回の復習
  - Pythonコードを編集
- 独自の検出器を構築する
  - 独自の学習データを構築
- グループのテーマを考える
  - アイデア出しを開始

# 物体検出の実践的な課題例

独自テーマを歓迎

- 遠隔地の混雑判定（2021年度）
- 空席判定（2021，2023年度年度）
- 文字認識とその応用（2021-2022年度）
- 校内ランドマーク検出と地図マッピング（2023年度）
- 骨格検出による手のジェスチャー分類
- 歩きスマホの検出

**日常に潜む課題に“気づき”を得て，その実践的な解決を通じて，画像認識とデータ分析の基礎を身につける**