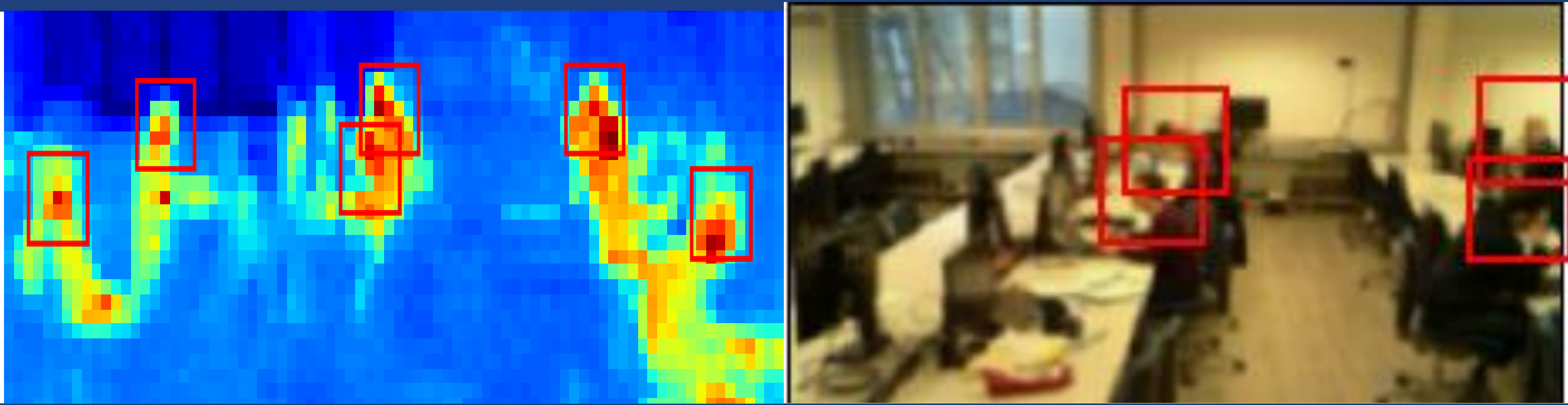
ETH zürich



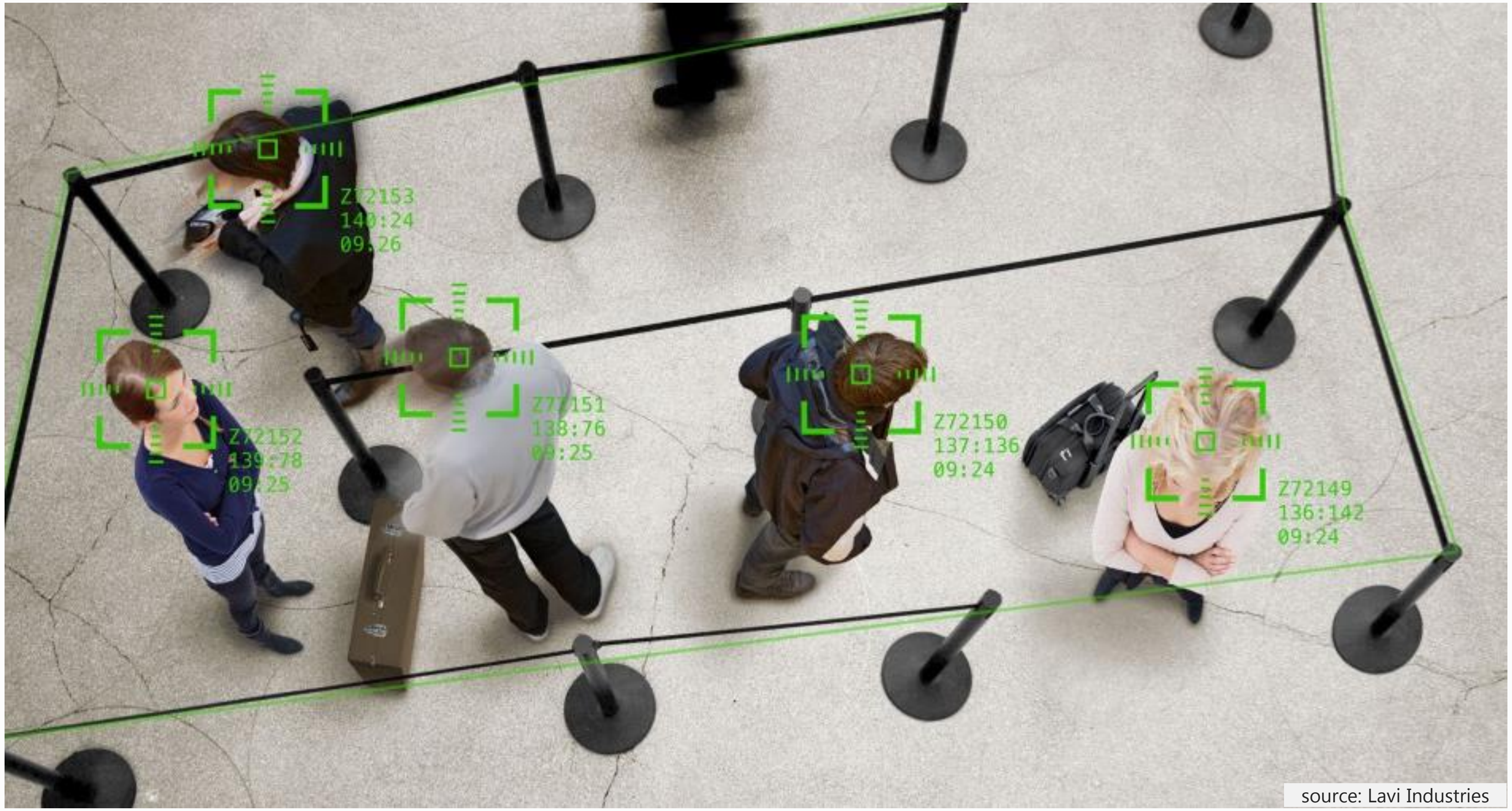
Thermal Image-Based CNN's for Ultra-Low Power People Recognition

Andres Gomez   Francesco Conti   Luca Benini

ETH Zurich, University of Bologna

Low-Power Embedded Systems workshop @ CF'18, Ischia, Italy – 9 May 2018

People Recognition



source: Lavi Industries

Smart buildings:

- Occupancy estimation
- Queue management
- HVAC systems

Energy autonomy:

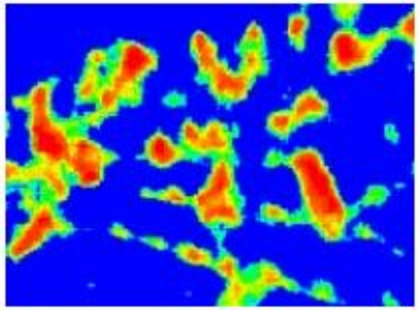
- Low maintenance sensors
- Leave in the field

Embedded people recognition: examples

[F. Conti et al. 2014]

high resolution
visible imaging

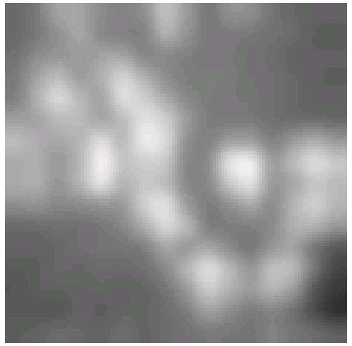
CNN



head detection or
density estimation

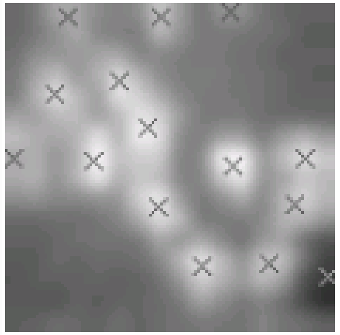
[M. Berger et al. 2010]

(top view)



low resolution
thermal imaging

blob detection



count estimation

Ultra-low-power people recognition

From the computer vision side:

- Many dependencies (perspective, lighting conditions, scenario/background)
- Proper datasets (ground truth)
- Privacy concerns

From the implementation side:

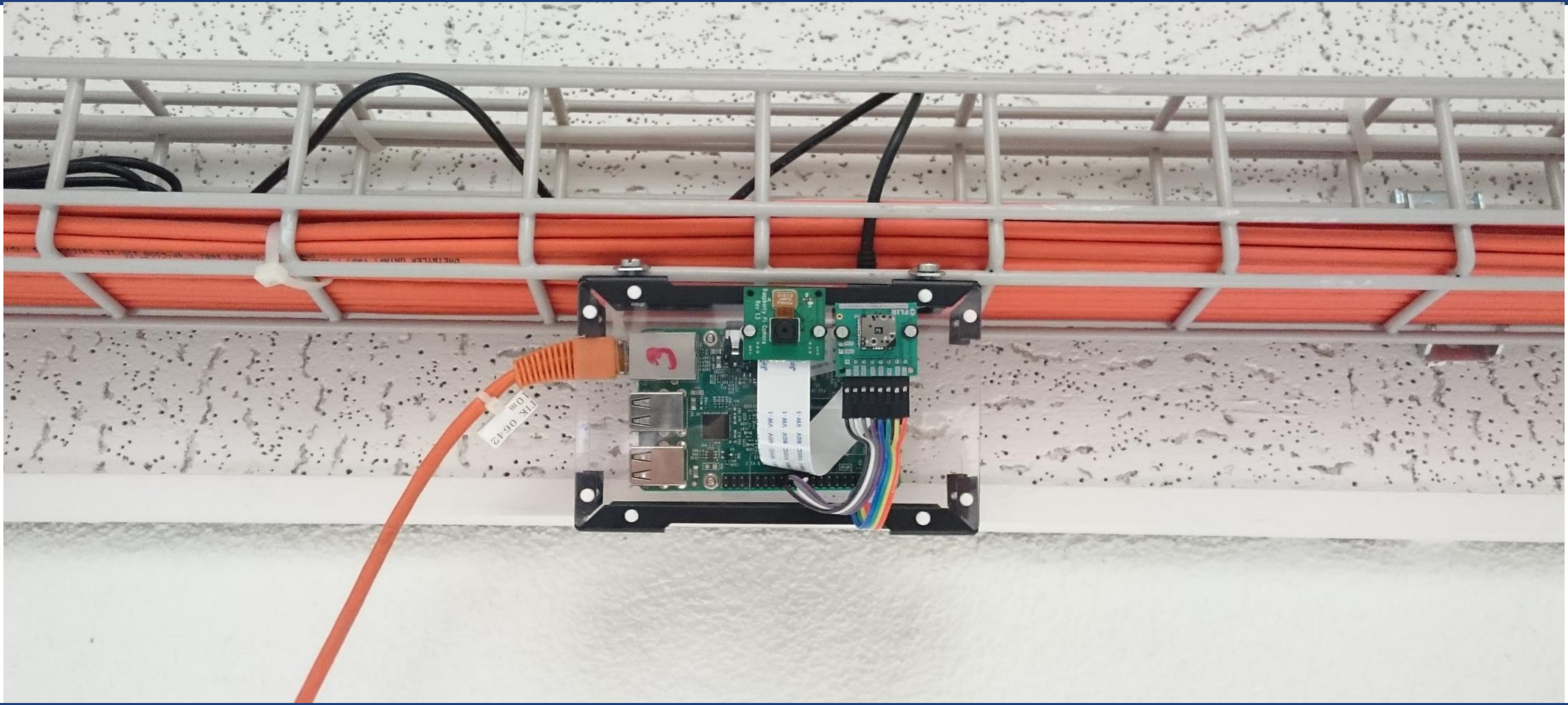
- Limited memory
- Limited processing power

Research question

Can we achieve people counting functionality on a resource-constrained embedded system?

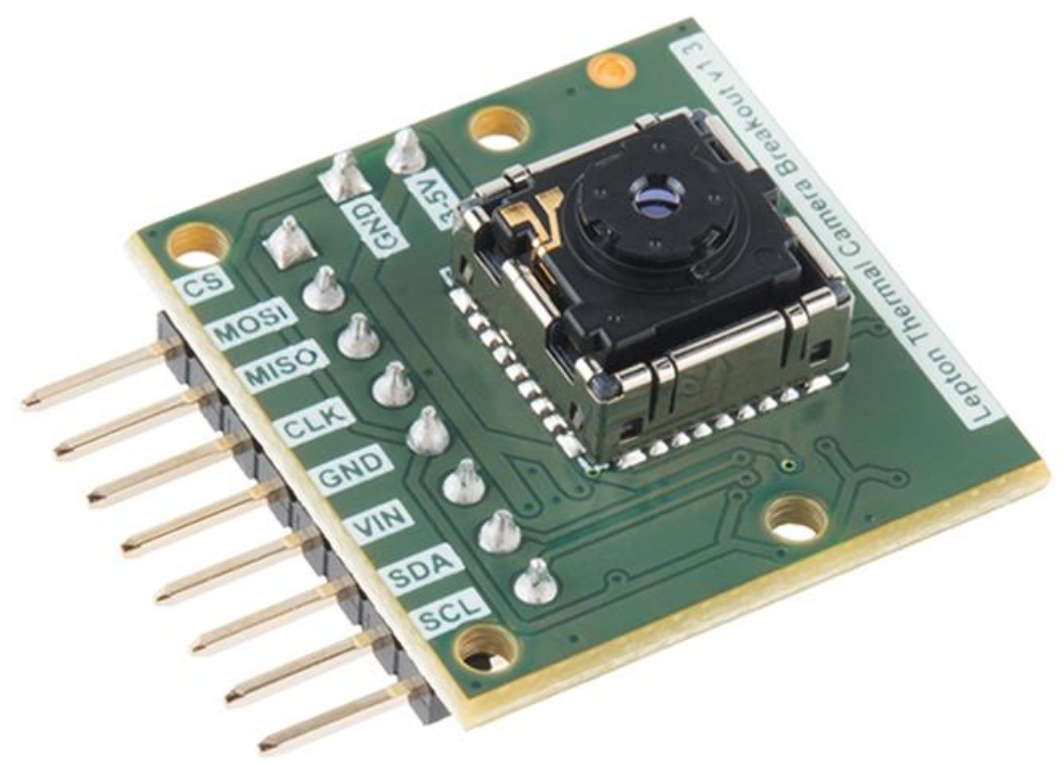
Contributions

1. Collected a dataset of 3000+ manually tagged thermal and visible images
2. Developed an algorithm to count the number of people with sliding windows and NMS
3. Compared head counting and detection errors on both thermal and visible images
4. Provided a implementation on the low-power LP54110 platform



Dataset Acquisition and Pre-Processing

Image Capturing Hardware: Thermal



- **FLIR Lepton** Thermal Camera
- Long-wave infrared: 8 – 14 µm
- Thermal information isolates warm objects from background
- Low resolution (80x60 pixel) compared to classic computer vision

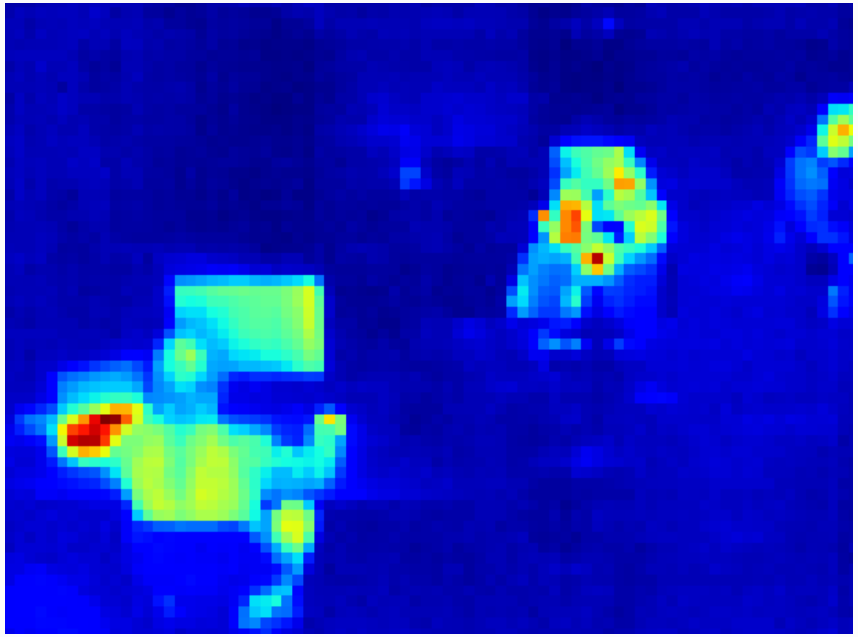
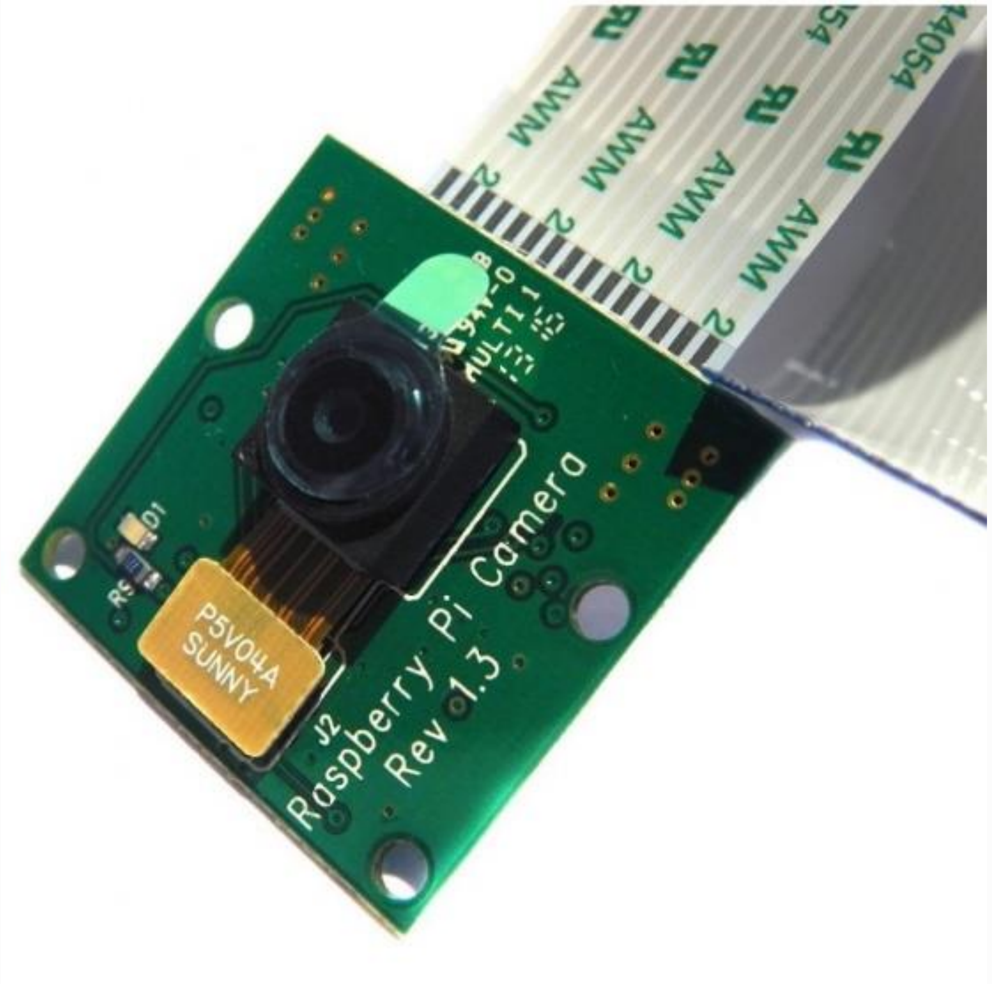


Image Capturing Hardware: Visual

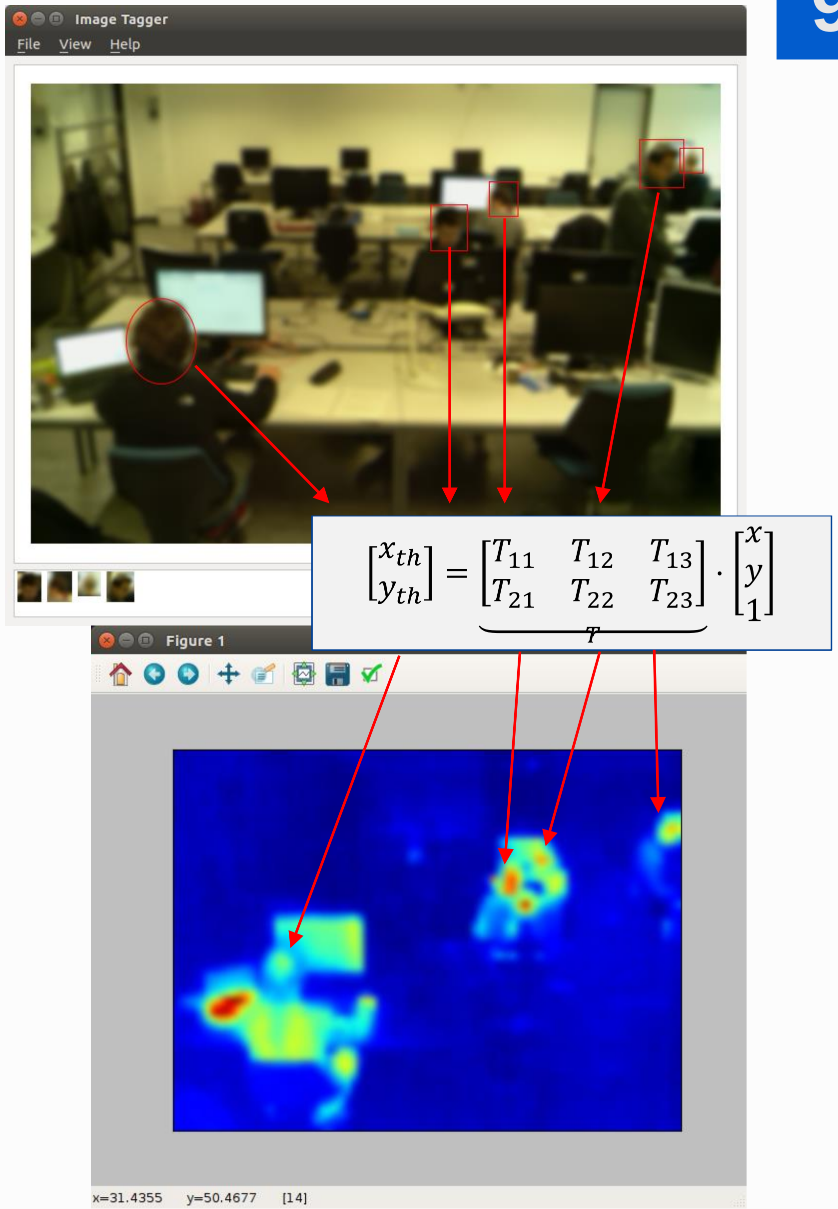


- **Raspberry Pi** Camera
- Images recorded at 720x480p
- Artificially blurred (privacy)
- Useful for reference/cross check

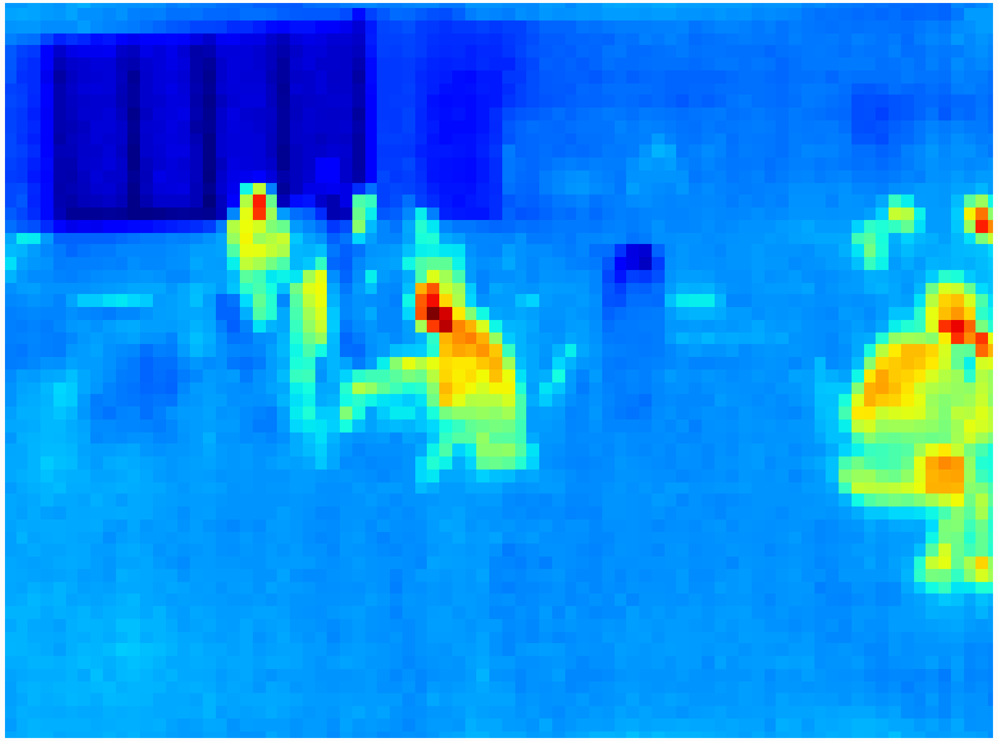
Dataset collection

- We deployed **Raspberry Pi** boards equipped with the two cameras in several ETH classrooms

- The *full-image dataset* collected consists of **~3000 images** in thermal and visual version (70% training, 15% validation, 15% test)
- All images have been tagged manually based on the visual version, using an empirical transformation to derive the thermal tags



Visual vs Thermal



	Visible Image	Thermal Image
Privacy	low	high
Resolution	high	low
Cost	low	high
Accuracy	?	?

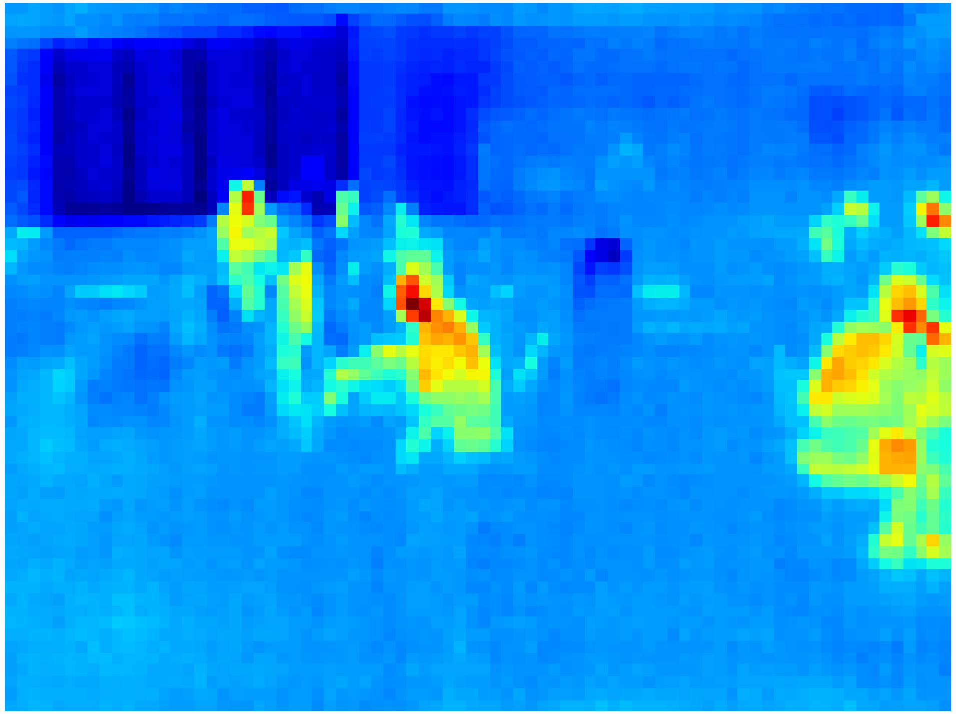
ETH zürich

People Counting Algorithm

Prototype in Python Framework Keras

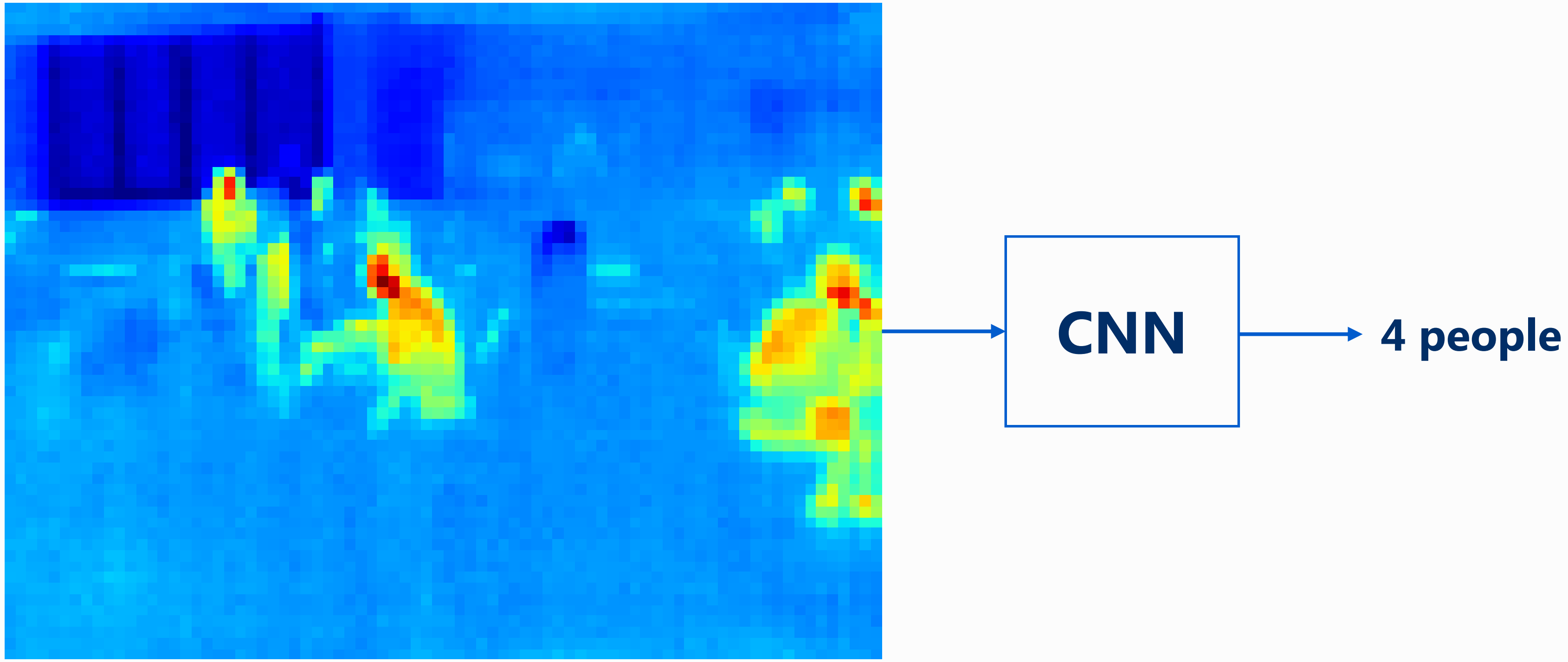
How can we count people?

- In order to count, we need to detect first: use a **CNN**
  - known to be effective on visual problem; popular; efficient libraries are starting to appear



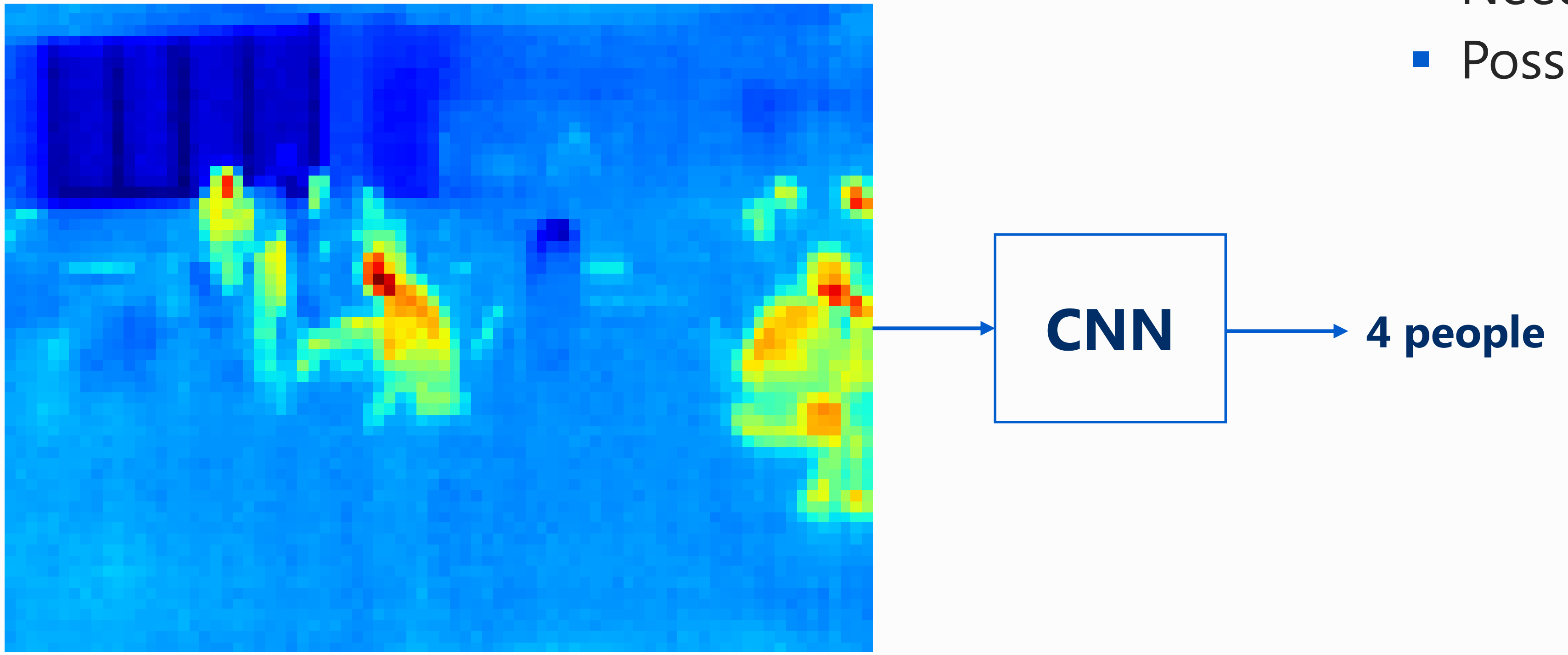
How can we count people?

- In order to count, we need to detect first: use a **CNN**
  - known to be effective on visual problem; popular; efficient libraries are starting to appear
- Apply CNN to input image?



How can we count people?

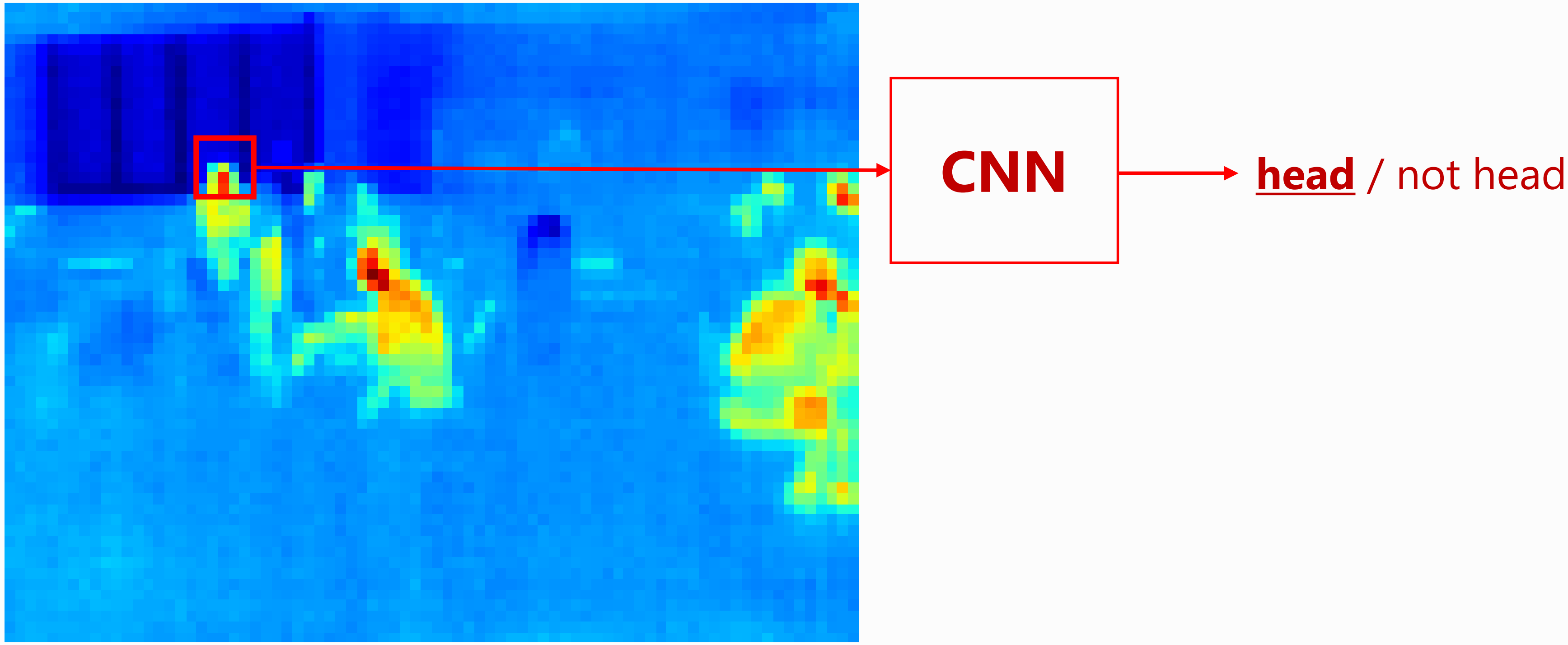
- In order to count, we need to detect first: use a **CNN**
  - known to be effective on visual problem; popular; efficient libraries are starting to appear
- Apply CNN to input image? ✗



- Problems:
  - High memory use
  - Needs many training images
  - Possible overfitting to scene

How can we count people?

- In order to count, we need to detect first: use a **CNN**
  - known to be effective on visual problem; popular; efficient libraries are starting to appear
- Sliding detection window?

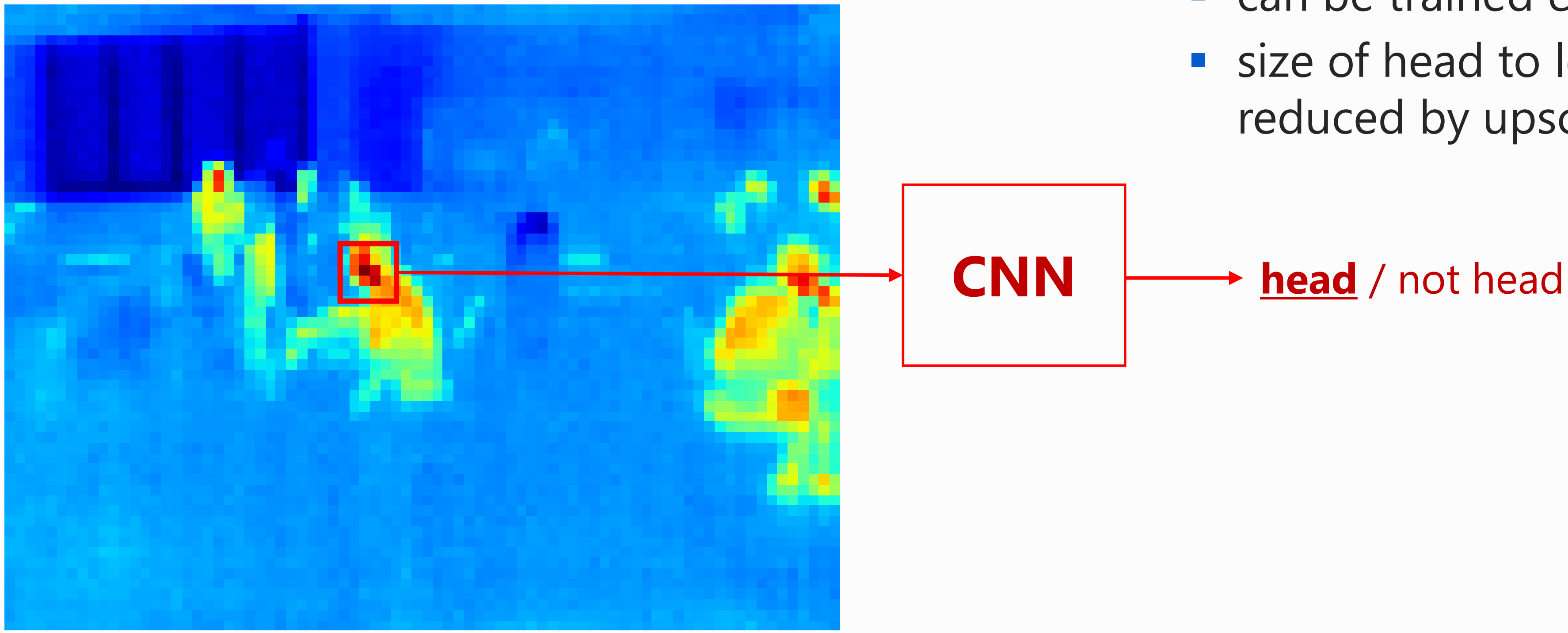


How can we count people?

- In order to count, we need to detect first: use a **CNN**
  - known to be effective on visual problem; popular; efficient libraries are starting to appear
- Sliding detection window? ✓

Binary classification problem:

- fed with a small 12x12 patch
- can be trained efficiently
- size of head to look for can be reduced by upscaling input image

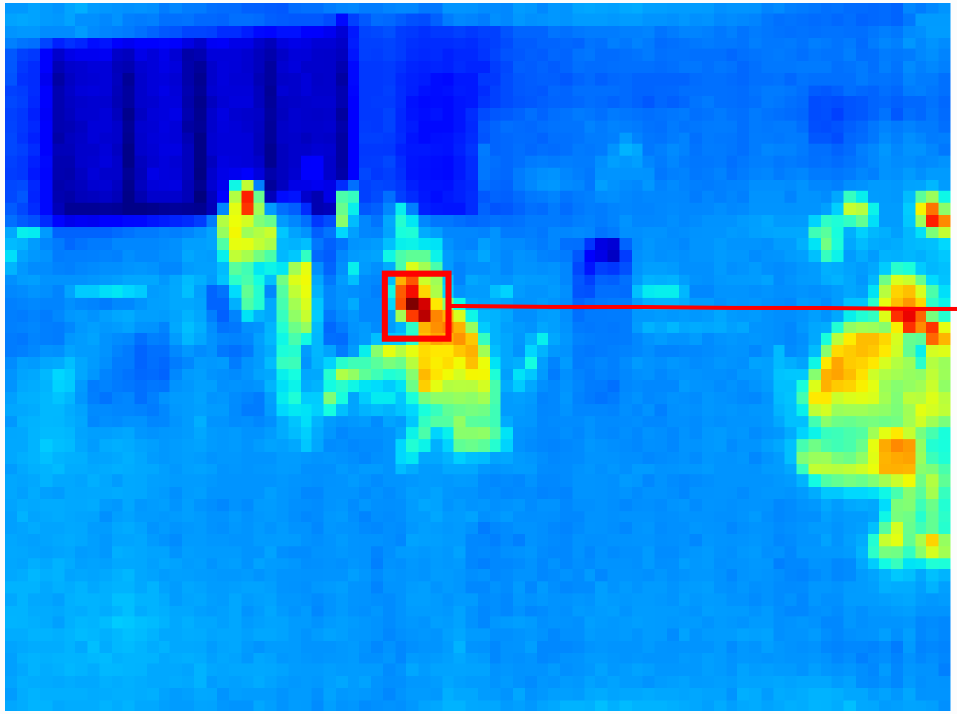


How can we count people?

- In order to count, we need to detect first: use a **CNN**
  - known to be effective on visual problem; popular; efficient libraries are starting to appear
- Sliding detection window? ✓

Binary classification problem:

- fed with a small 12x12 patch
- can be trained efficiently
- size of head to look for can be reduced by upscaling input image

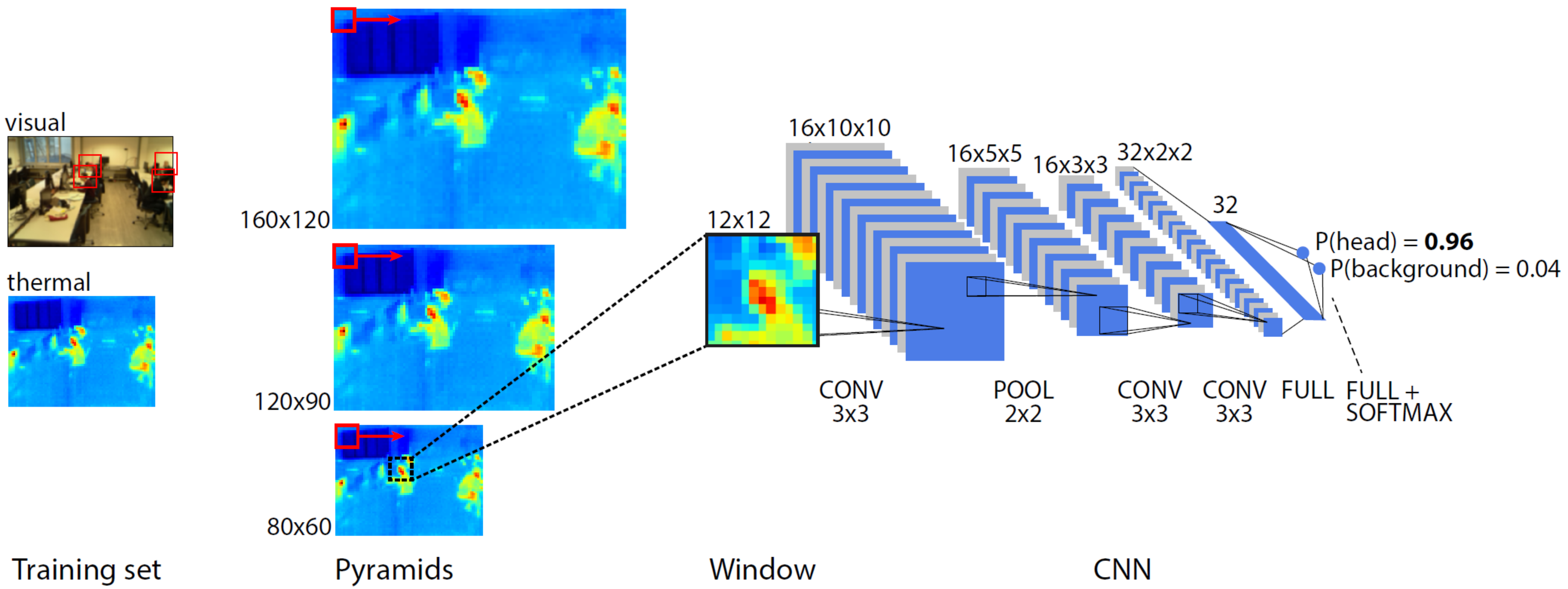


CNN → **head** / not head

Targeting embedded platform

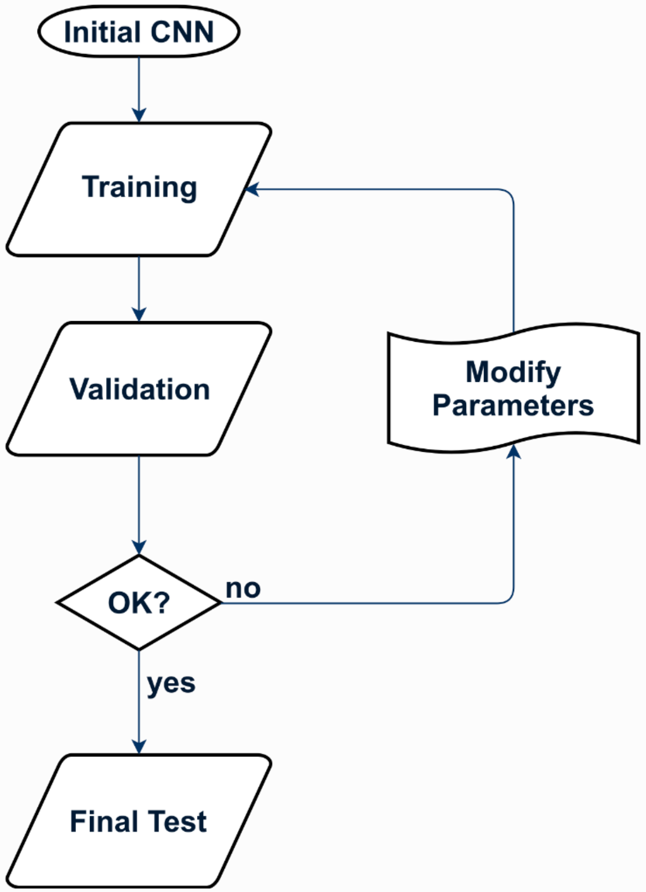
- COTS LPC microcontroller
- ***~500kB*** memory constraint
- ***80MHz***

Convolutional Neural Network Topology



Training methodology

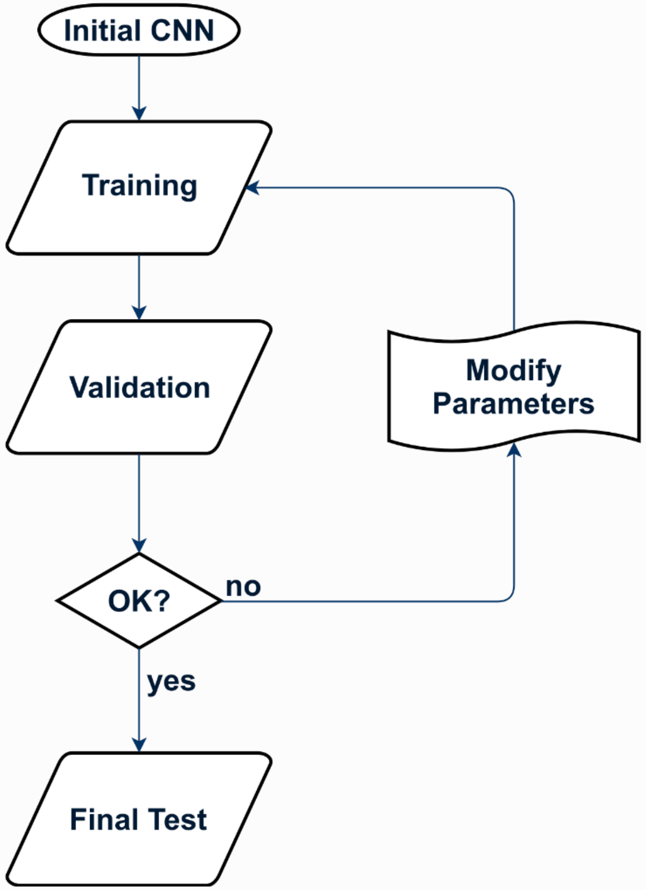
- CNN input sliding window dataset for training built from all *head cuts* + random background cuts:
  - ***training set*** with 4203 head cuts + 5000 random backgrounds built from full-image training set in the three scales
  - ***validation set*** with 850 head cuts + 5000 random backgrounds from full-image validation set, + 4250 heads constructed with data augmentation (noise / gradient)
  - ***test set*** with 872 head cuts + 67540 background cuts
- CNN is trained with backpropagation
  - using **Keras/Tensorflow** as backend
  - **Adam** optimizer with lr = 5e-5, **L2 penalty** of 0.05 on Conv layers
  - **validation loss** is used to select the best result over 300 epochs



Training methodology

- CNN input sliding window dataset for training built from all *head cuts* + random background cuts:
    - ***training set*** with 4203 head cuts + 5000 random backgrounds built from full-image training set in the three scales
    - ***validation set*** with 850 head cuts + 5000 random backgrounds from full-image validation set, **+ 4250 heads constructed with data augmentation (noise / gradient)**
    - ***test set*** with 872 head cuts + 67540 background cuts
- CNN is trained with backpropagation
    - using **Keras/Tensorflow** as backend
    - **Adam** optimizer with lr = 5e-5, **L2 penalty of 0.05 on Conv layers**
    - **validation loss is used to select the best result over 300 epochs**

reduce overfitting

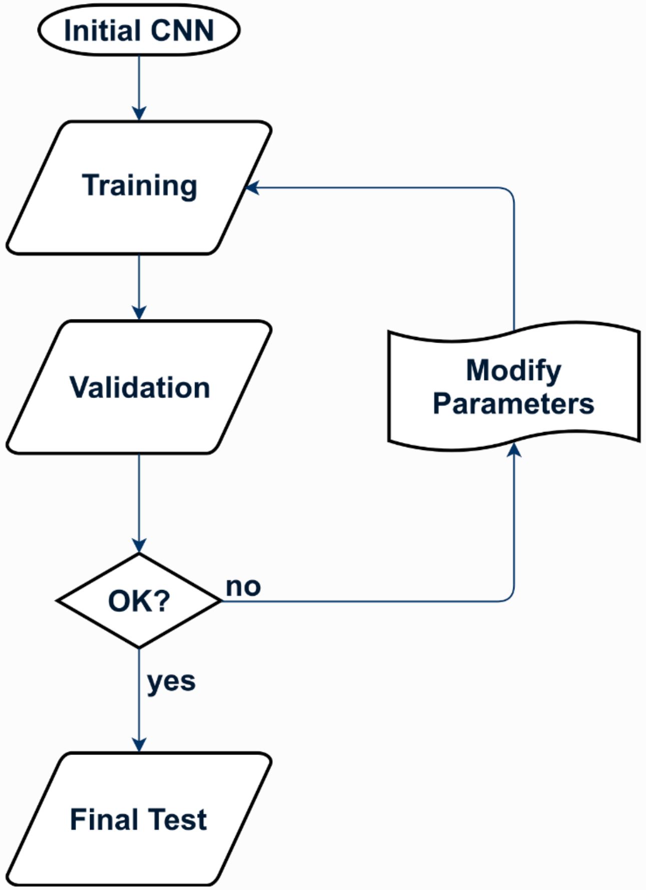


Training methodology

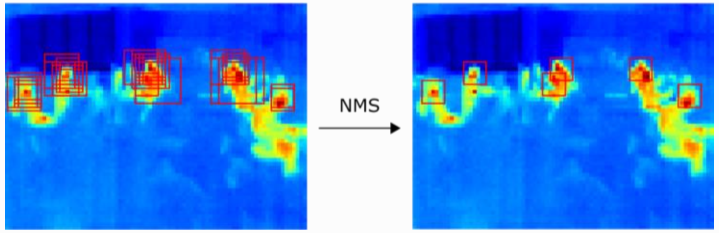
- CNN input sliding window dataset for training built from all *head cuts* + random background cuts:
  - ***training set*** with 4203 head cuts + 5000 random backgrounds built from full-image training set in the three scales
  - ***validation set*** with 850 head cuts + 5000 random backgrounds from full-image validation set, **+ 4250 heads constructed with data augmentation (noise / gradient)**
  - ***test set*** with 872 head cuts + 67540 background cuts

reduce overfitting

- CNN is trained with backpropagation
  - using **Keras/Tensorflow** as backend
  - **Adam** optimizer with lr = 5e-5, **L2 penalty of 0.05 on Conv layers**
  - **validation loss is used to select the best result over 300 epochs**

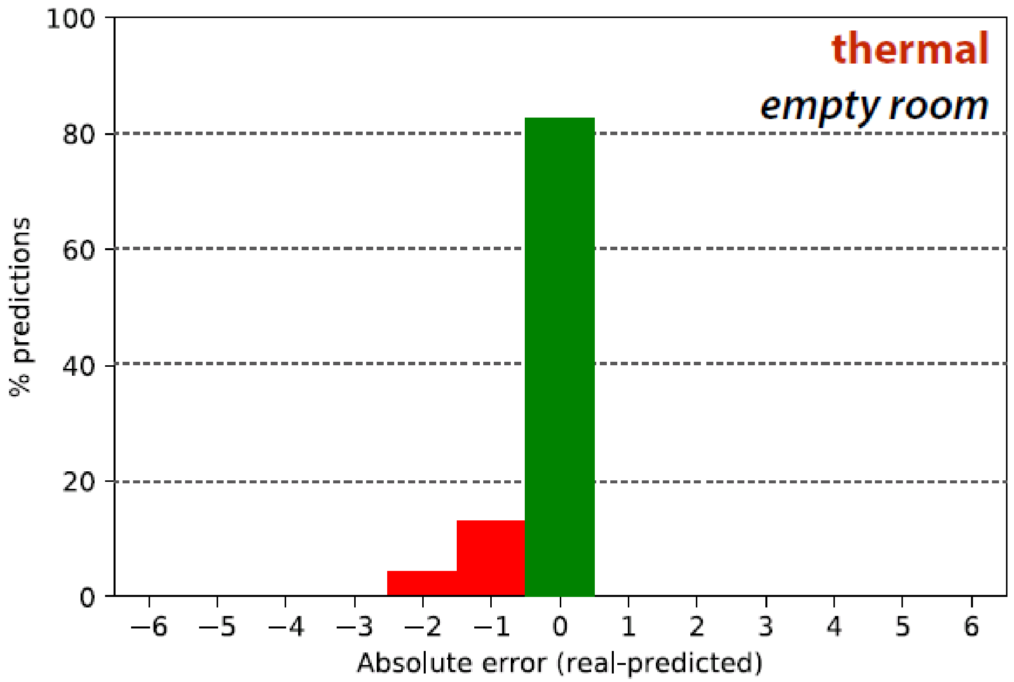


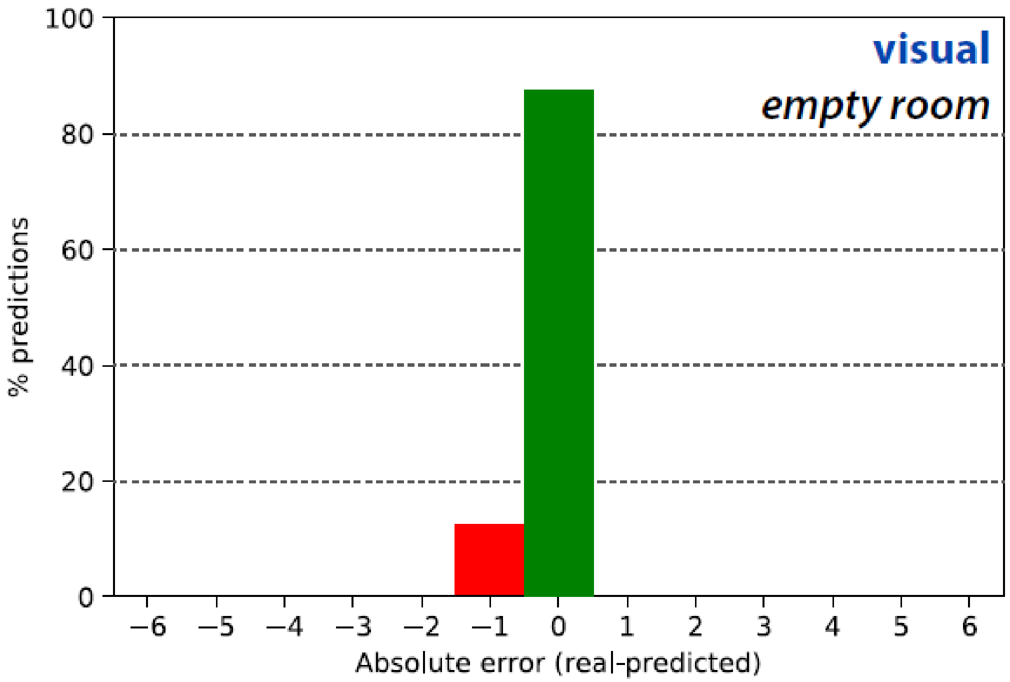
- Non-maximum suppression
  - removes duplicate matches



Evaluating Accuracy

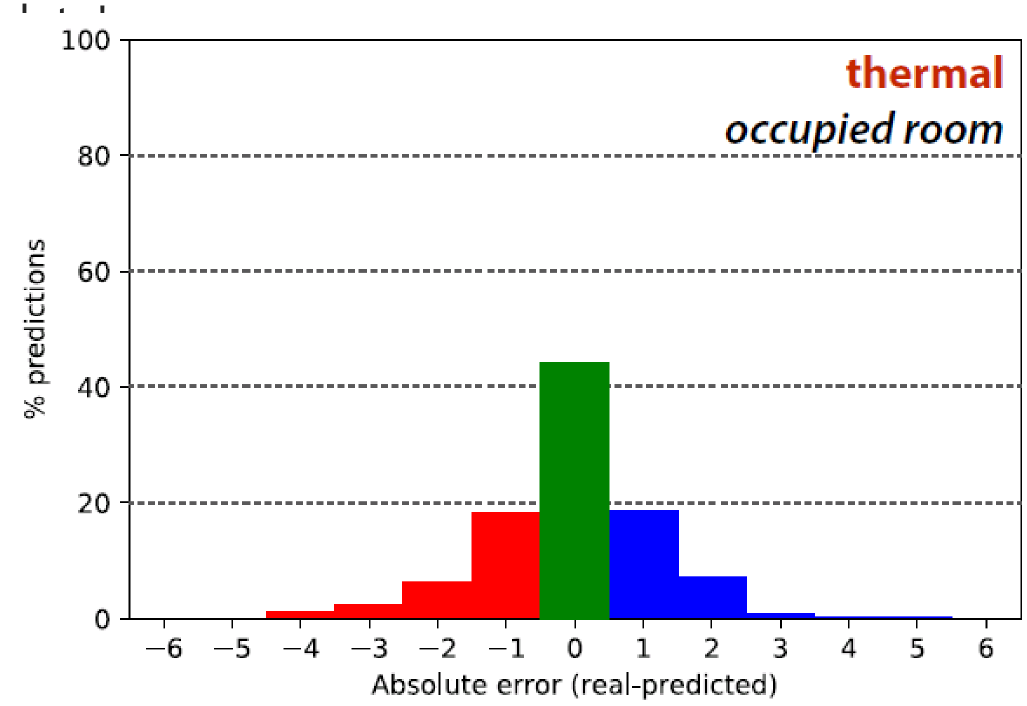
- Test accuracy on *head cuts* dataset: **95.9%**, up to **99.0%** with NMS
- Test accuracy on *full images* test set hit by two separate mechanisms
  - many sliding windows taken into account -> even a small percentage of classification error results in significant counting error
  - ***false positives*** (red bars)
  - ***missed predictions*** (blue bars)
- For ***empty rooms***, only false positives are relevant -> both visual and thermal achieve good accuracy

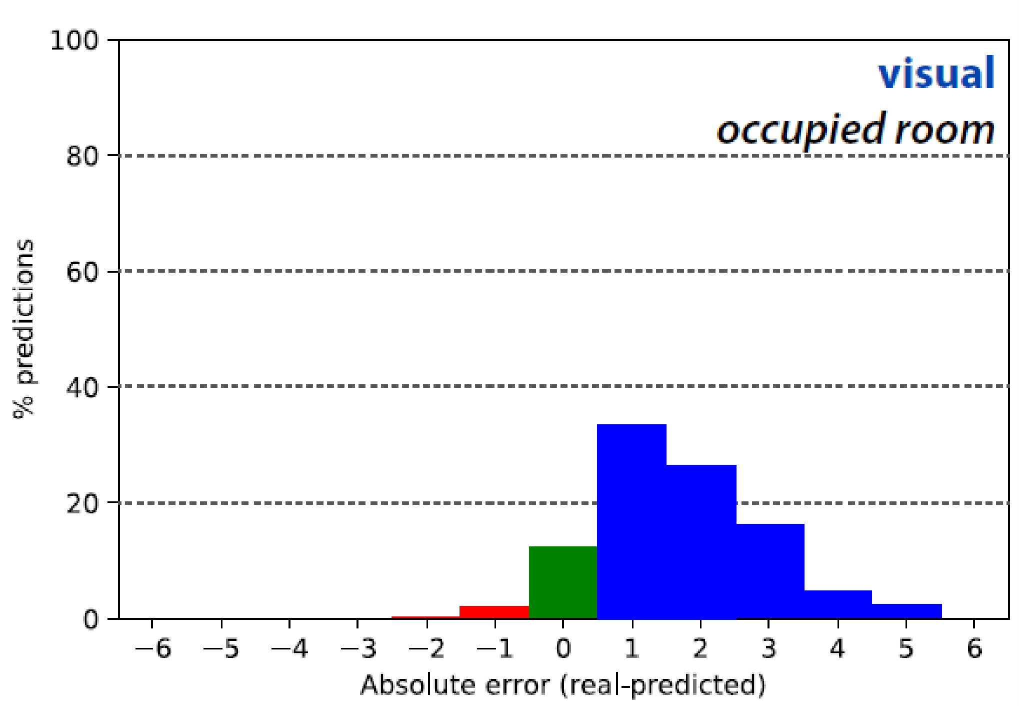




Evaluating Accuracy

- Test accuracy on *head cuts* dataset: **95.9%**, up to **99.0%** with NMS
- Test accuracy on *full images* test set hit by two separate mechanisms
  - many sliding windows taken into account -> even a small percentage of classification error results in significant counting error
  - ***false positives*** (red bars)
  - ***missed predictions*** (blue bars)
- For ***occupied rooms***, also missed predictions are relevant -> our tiny CNN cannot generalize on visual





Evaluating Accuracy

- Test accuracy on *head cuts* dataset: **95.9%**, up to **99.0%** with NMS
- Test accuracy on *full images* test set hit by two separate mechanisms
  - many sliding windows taken into account -> even a small percentage of classification error results in significant counting error
  - ***false positives*** (red bars)
  - ***missed predictions*** (blue bars)

- Overall, correct people count with thermal images for **45%** of test images, error within ±1 for **81%** images
- Visual counting is ~garbage: **10%** of correct counts (mainly empty images!)
  - the visual image is "noisy" -> a bigger CNN would be required

Embedded deployment

Custom-built evaluation board

- Energy harvesting (beyond this work)
- BLE (beyond this work)
- FLIR lepton camera
- LPC54110 @ 80 MHz, 2.8 V

Figures of merit

- Power consumption of the LPC microcontroller
- Processing time (frame rate)
- Memory breakdown

Experimental Results

- Can we run on this platform at all?

Experimental Results

- Can we run on this platform at all? ✓

Memory Breakdown

Section	Size [B]
Text	245×10^3
BSS	63×10^3
Data	186

Experimental Results

- Can we run on this platform at all? ✓
- How fast / how good is the deployment?

Memory Breakdown

Section	Size [B]
Text	245×10^3
BSS	63×10^3
Data	186

Experimental Results

- Can we run on this platform at all? ✓
- How fast / how good is the deployment?

Memory Breakdown

Section	Size [B]
Text	245×10^3
BSS	63×10^3
Data	186

Energy Breakdown

Task	Energy [J]	Exec. Time [s]
start-up + acquisition	0.1	1.3
CNN stride 2x2	4.7	138.0
CNN stride 3x3	2.2	63.0

Experimental Results

- Can we run on this platform at all? ✓
- How fast / how good is the deployment?

Memory Breakdown	
Section	**Size [B]**
Text	245×10^3
BSS	63×10^3
Data	186

Energy Breakdown		
Task	**Energy [J]**	**Exec. Time [s]**
start-up + acquisition	0.1	1.3
CNN stride 2x2	4.7	138.0
CNN stride 3x3	2.2	63.0

- ~2.3 minutes, 4.8 Joules
- **Near-autonomy** is achievable:
  - assume 1 inference every 10 minutes for 8 hours a day
  - **156 days** of autonomy on a standard **3600 mAh** battery

Summary & Future work

- Developed a CNN-based, head-detection algorithm with <500kb footprint
  - can be deployed on a LPC54110 COTS microcontroller
- Trained CNN with thermal and visible images
- Evaluated the accuracy of head detection on thermal and visible images
  - achieved **99%** classification error and error bound of ±1 on **81%** of full images
- Implemented the final algorithm on the LPC54110 platform
  - 5.8 MMAC/s on custom code, 4.8 J/image
  - achieves near-autonomy

Summary & Future work

- Developed a CNN-based, head-detection algorithm with <500kb footprint
  - can be deployed on a LPC54110 COTS microcontroller
- Trained CNN with thermal and visible images
- Evaluated the accuracy of head detection on thermal and visible images
  - achieved **99%** classification error and error bound of ±1 on **81%** of full images
- Implemented the final algorithm on the LPC54110 platform
  - 5.8 MMAC/s on custom code, 4.8 J/image

- Currently working on several improvements
  - using *CMSIS-NN* library (up to 4.5x speedup possible)
  - bigger CNN topology adapted to embedding via *quantization* / *binarization*
  - deployment on more advanced low-power architectures

PULP

Thanks for your attention.

Questions?