Detecting Human Presence in a Smart Home

Recognizing Room Occupancy by using Convolutional Neural Networks with Inexpensive Thermal Sensors

Ryan Divigalpitiya | rdivigal@uwo.ca

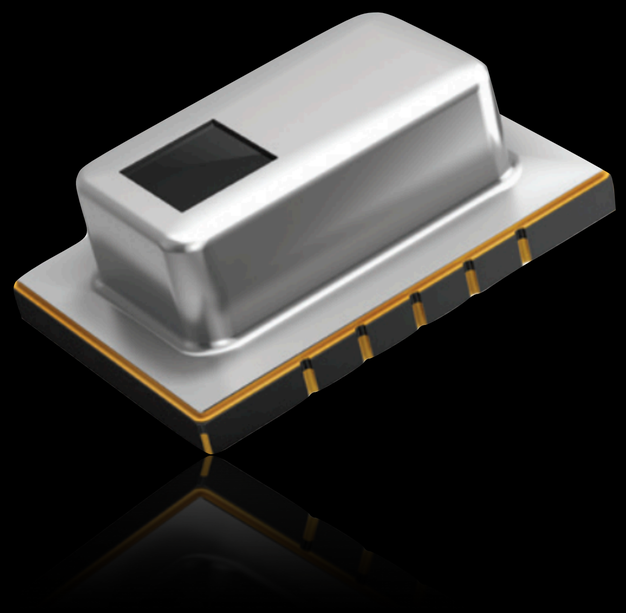
CS9860 Advanced Machine Learning

Western UNIVERSITY · CANADA

Introduction: Smart Homes

- Occupancy detection?
- Off-the-shelf products + sensors are expensive
    - hundreds to thousands of dollars

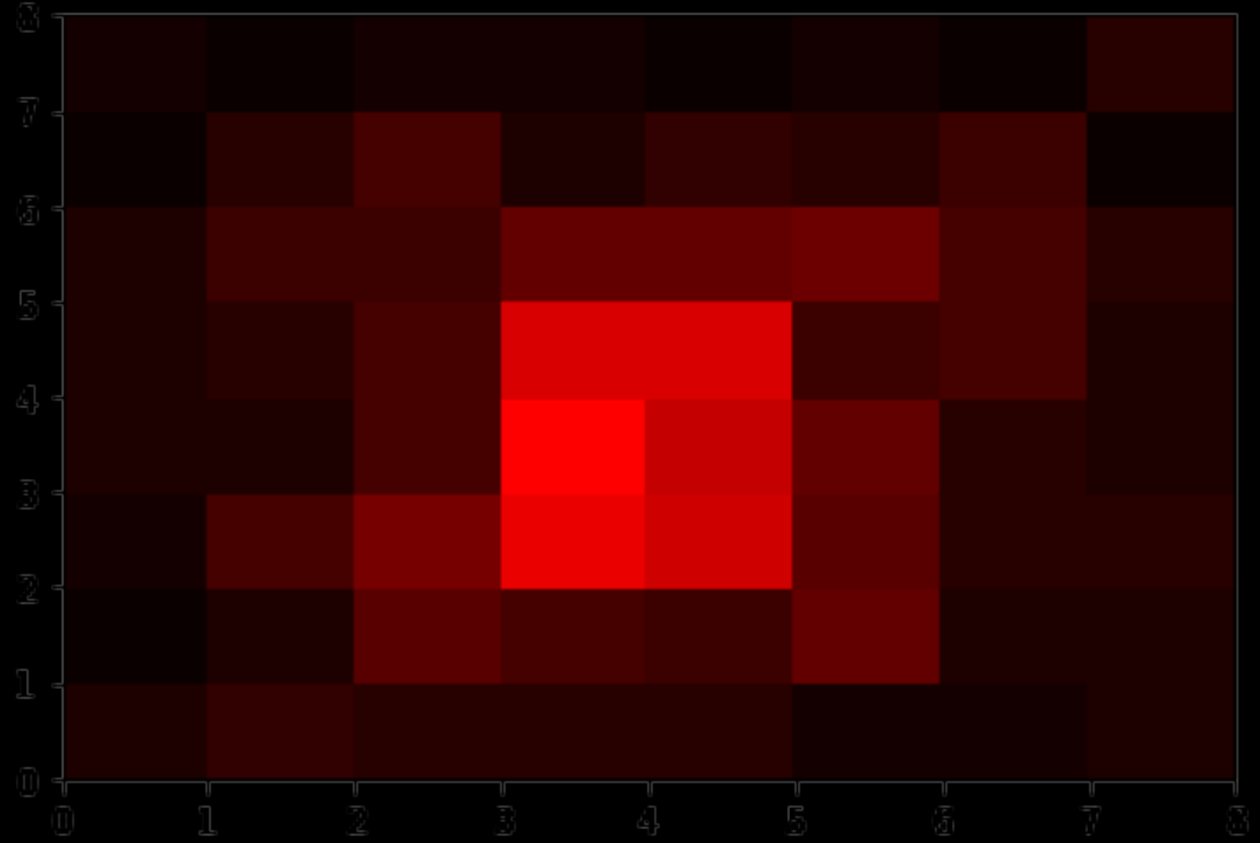
Introduction: Cheap Thermal Sensors



PANASONIC AMG8833

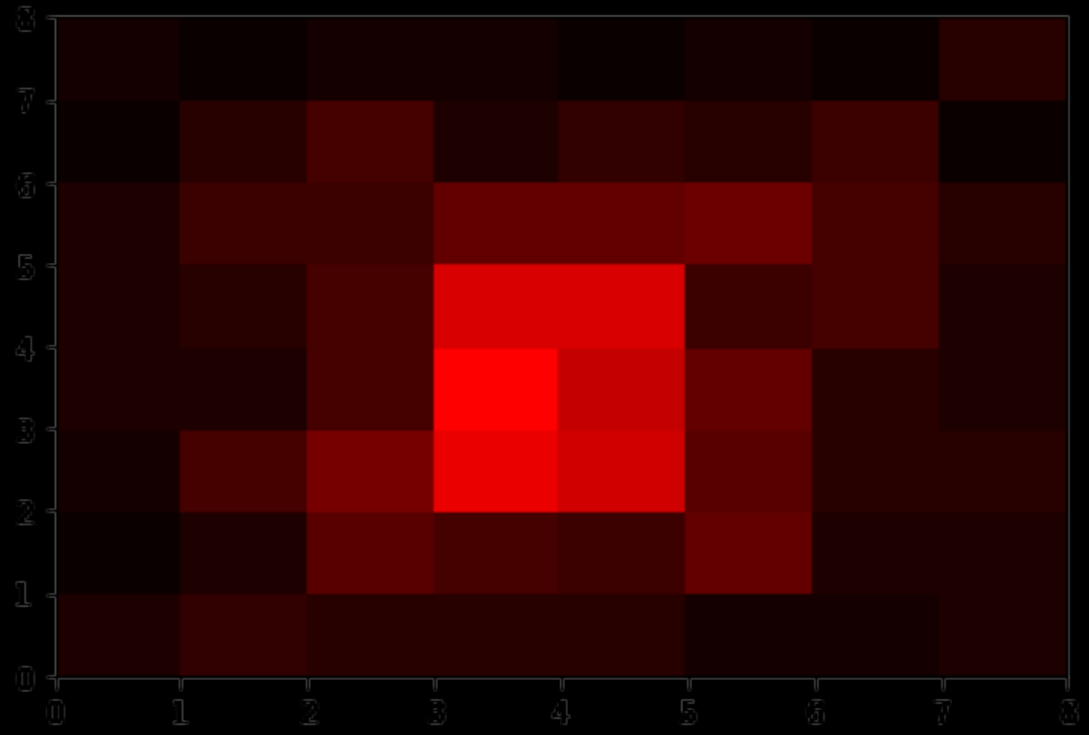
OMRON D6T

Introduction: Cheap Thermal Sensors



8x8 image size

Introduction: Cheap Thermal Sensors

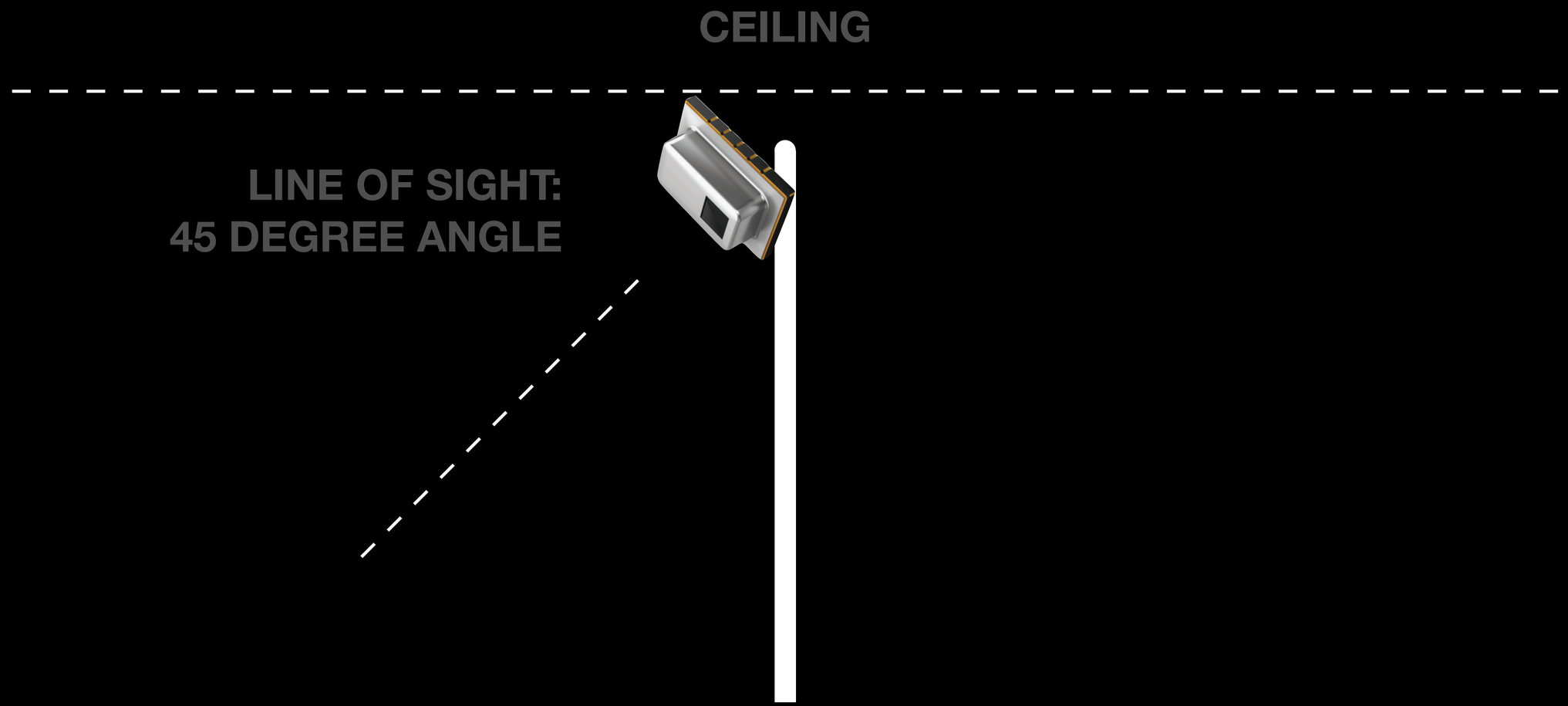


Project Goal

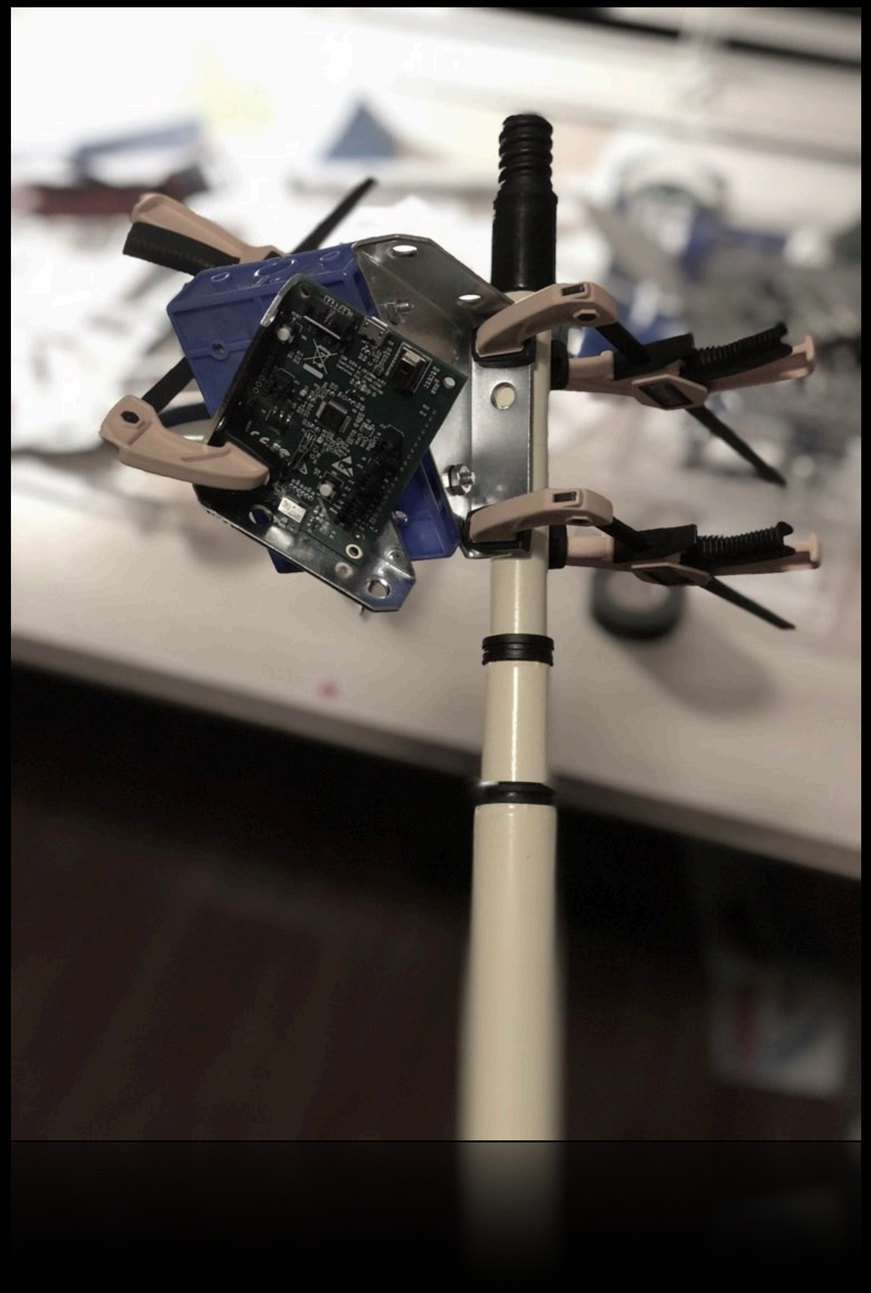
Project Goal

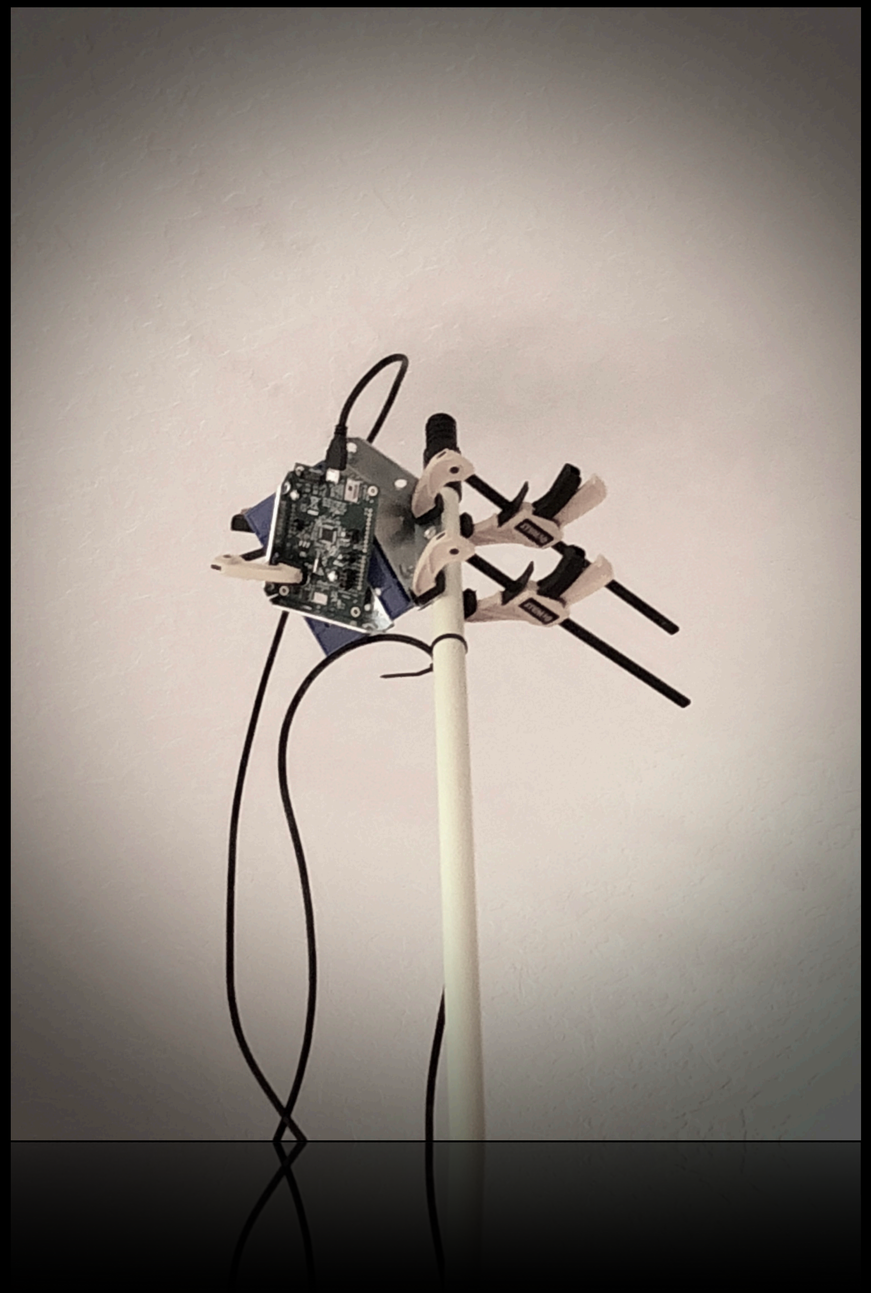
Can we build an AI-based solution that can distinguish between false-positive and true-positive signals using a very low-resolution, inexpensive thermal sensor?

Methods: Experimental Setup



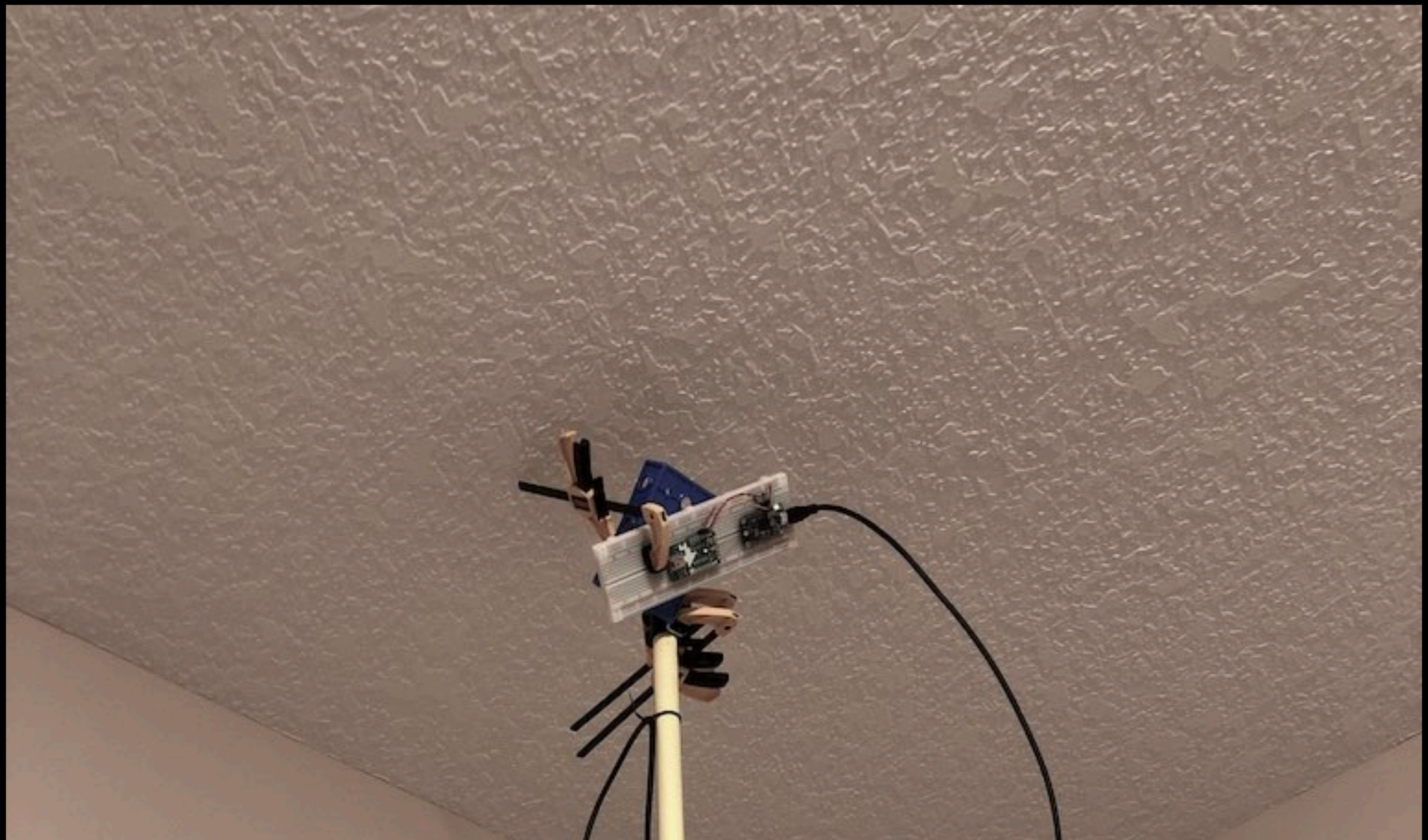
Methods: Experimental Setup



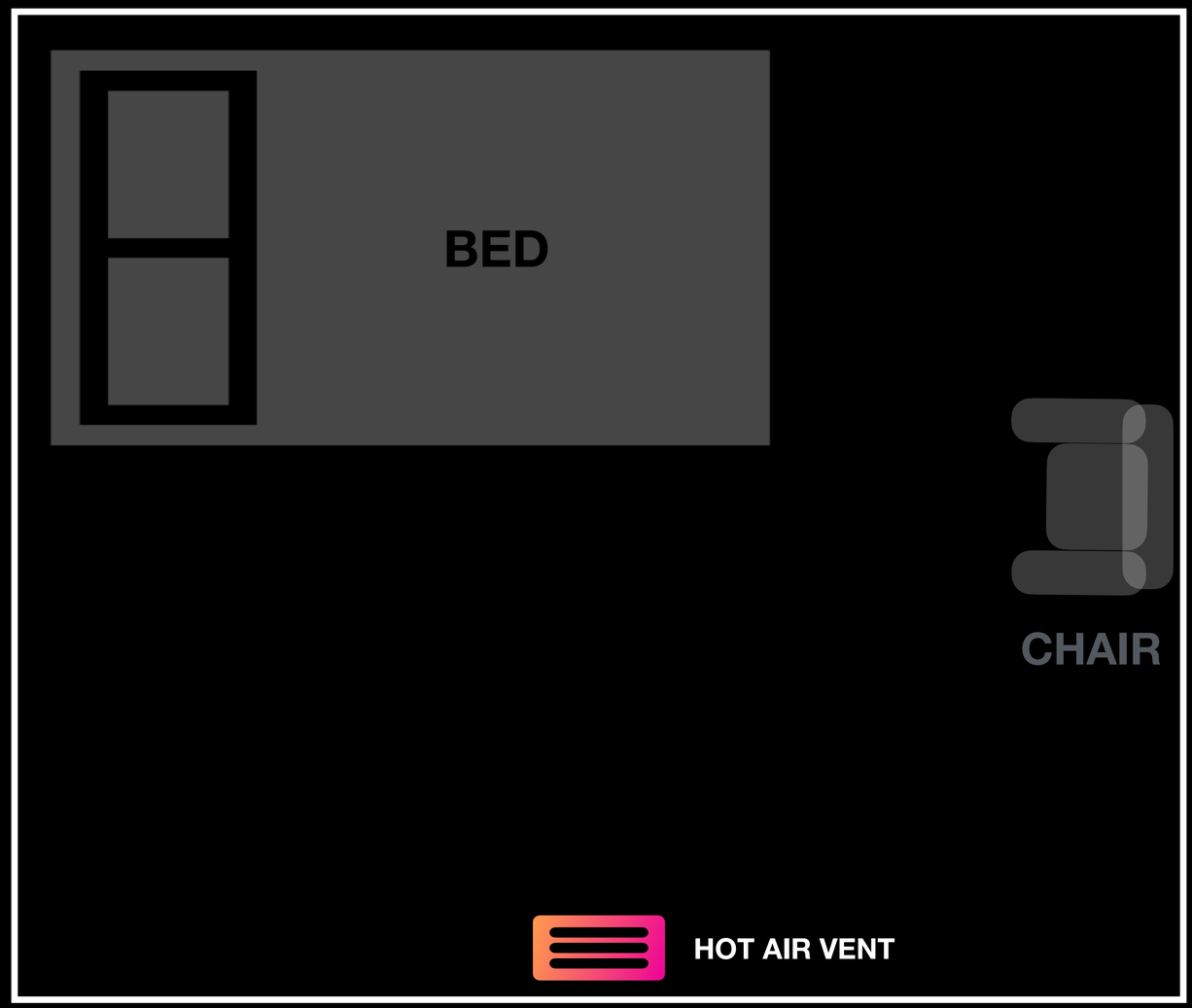


Methods: Experimental Setup

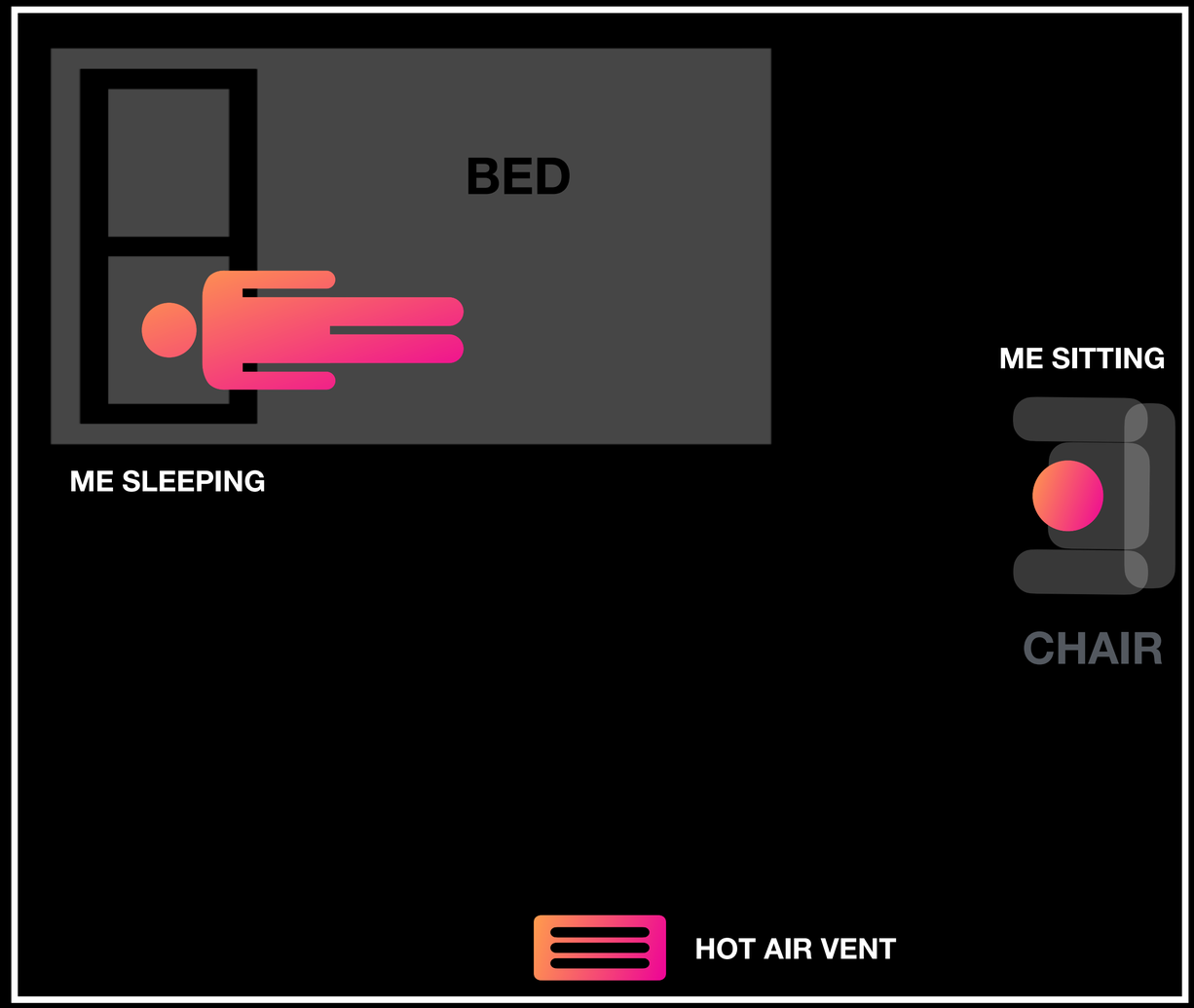
Methods: Experimental Setup



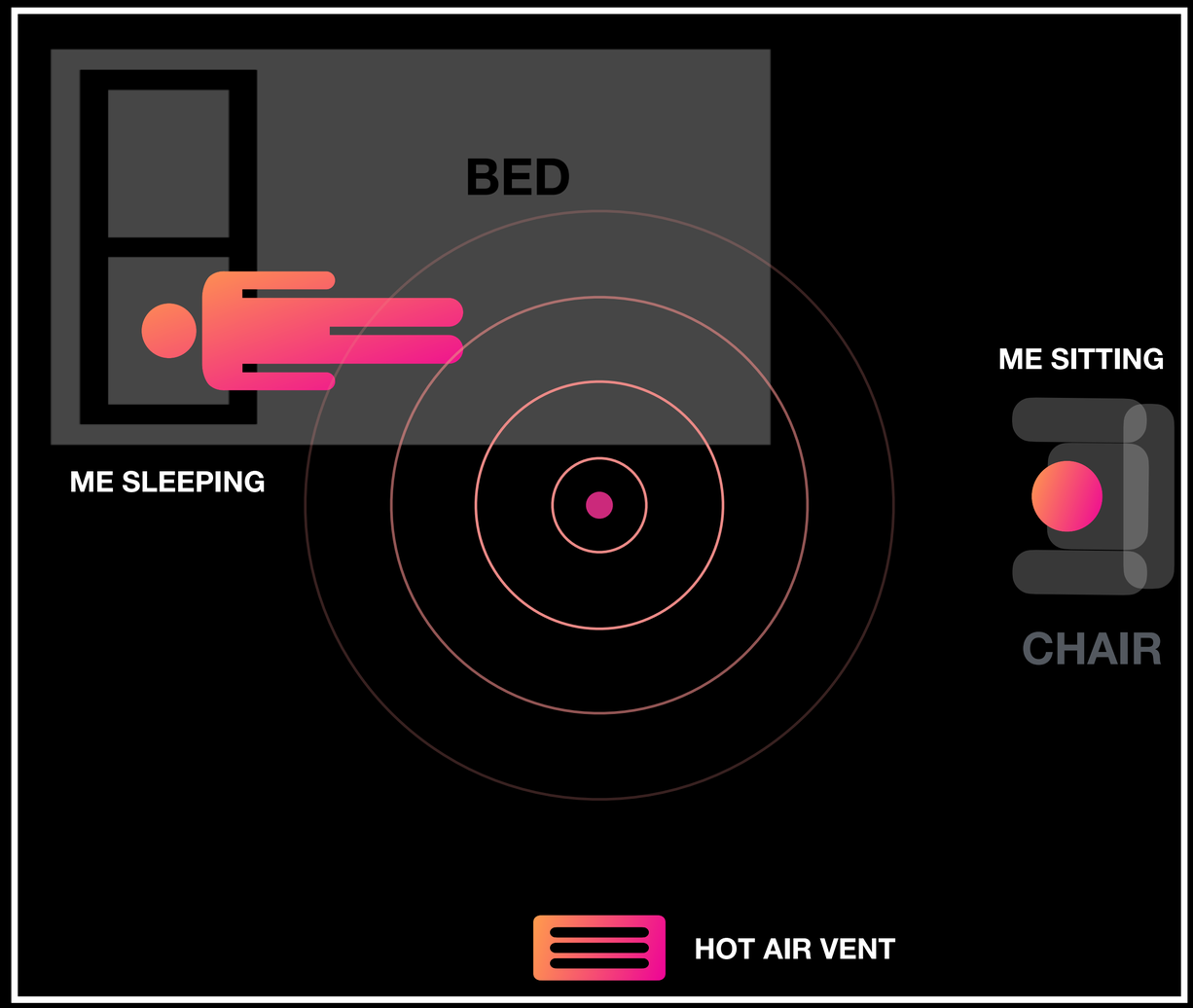
Methods: Experimental Setup



Methods: Experimental Setup



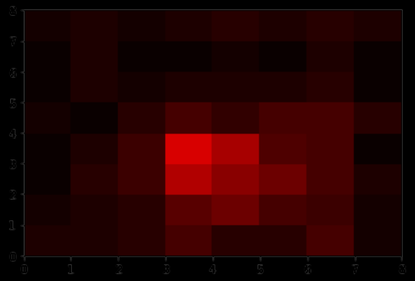
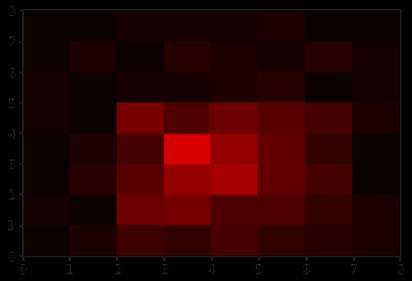
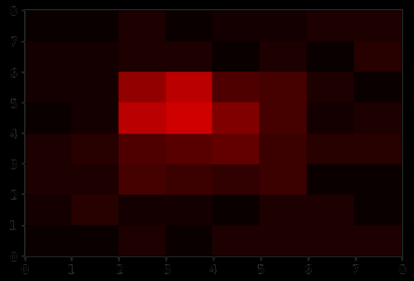
Methods: Experimental Setup



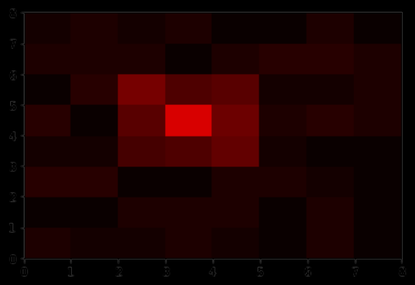
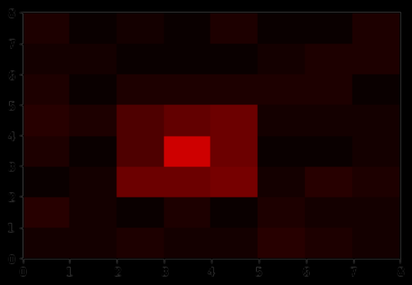
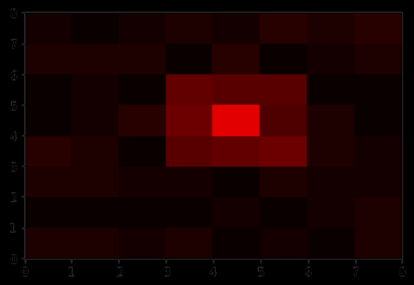
Methods: Acquired Data

The Panasonic AMG8833 sensor has a sampling rate of 10 images per second. Recording sensor data for approximately 10 minutes yields around 6,000 still frames (images).

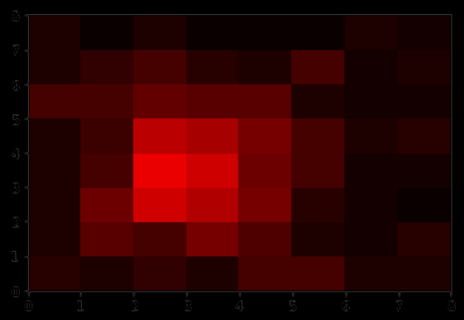
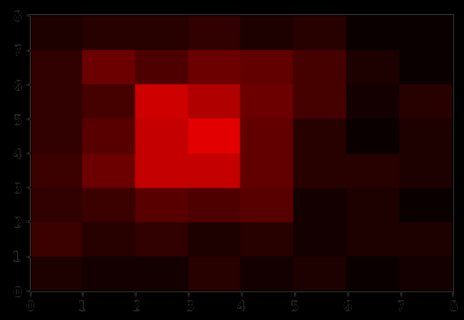
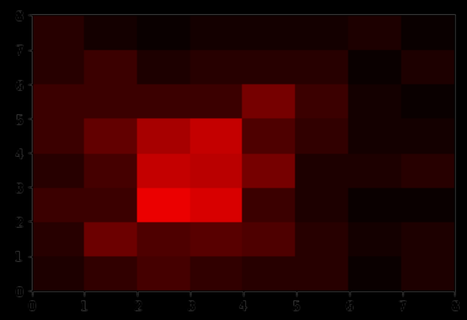
Set 1	6,000 images of **me** laying in my bed	8x8. pixel: 0-255	Class Label **1**
Set 2	6,000 images of **me** sitting in my chair	8x8. pixel: 0-255	Class Label **1**
Set 3	6,000 images of the **air-vent**	8x8. pixel: 0-255	Class Label **0**

Three instances of images of me lying in bed.

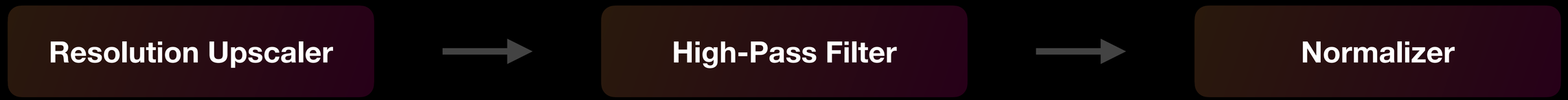




Three images of me sitting in my chair

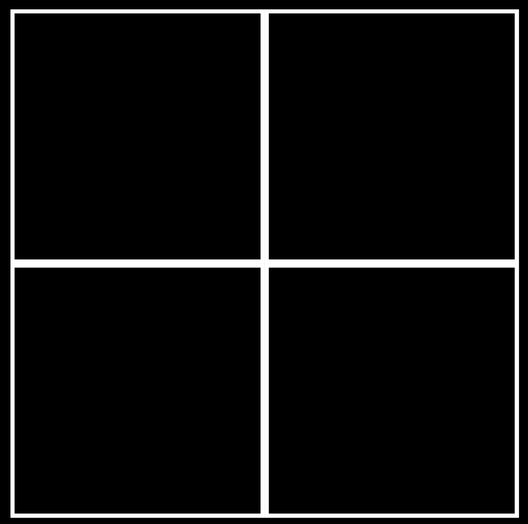
Three images of air vent

Methods: Pre-Processing Pipelines



Methods: Pre-Processing Pipelines

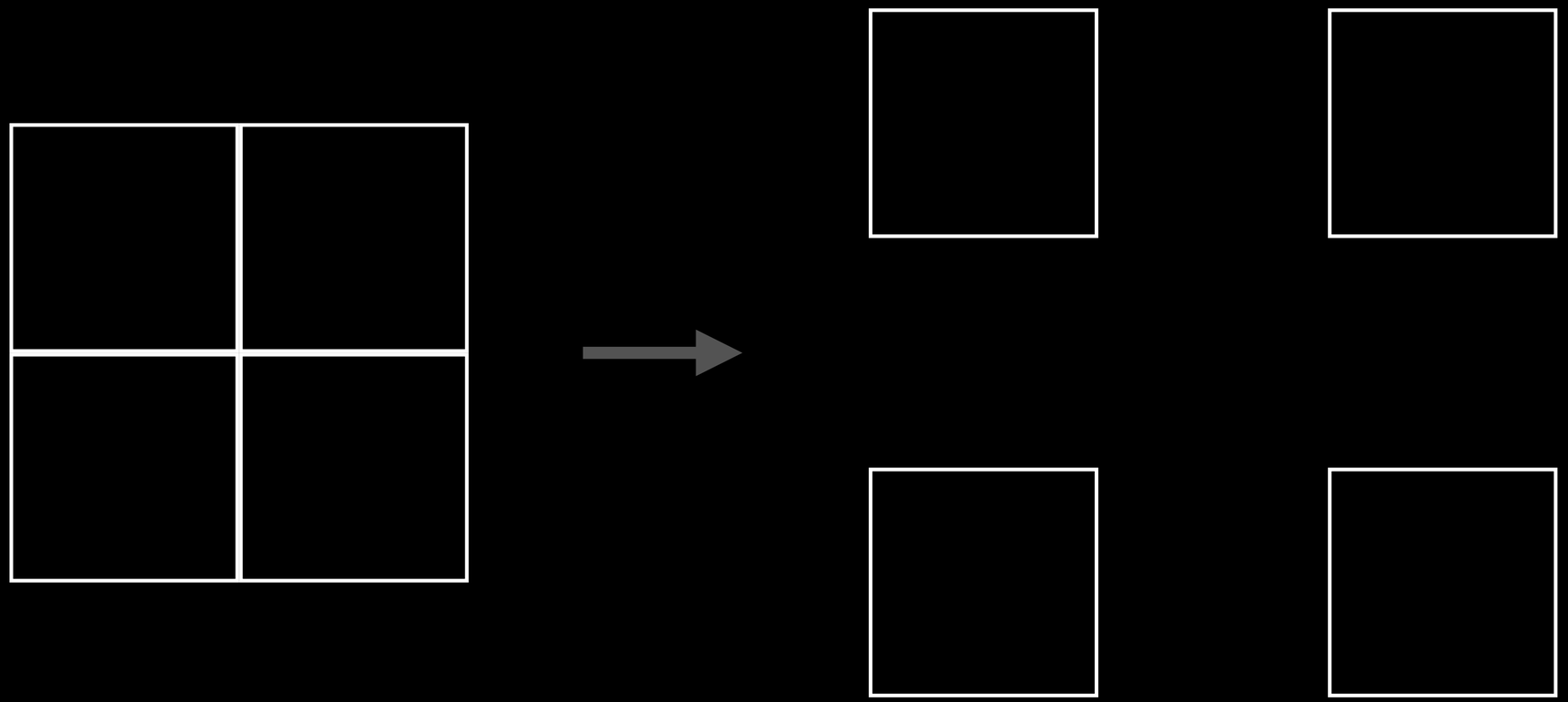
Resolution Upscaler



ORIGINAL IMAGE
2x2

Methods: Pre-Processing Pipelines

Resolution Upscaler

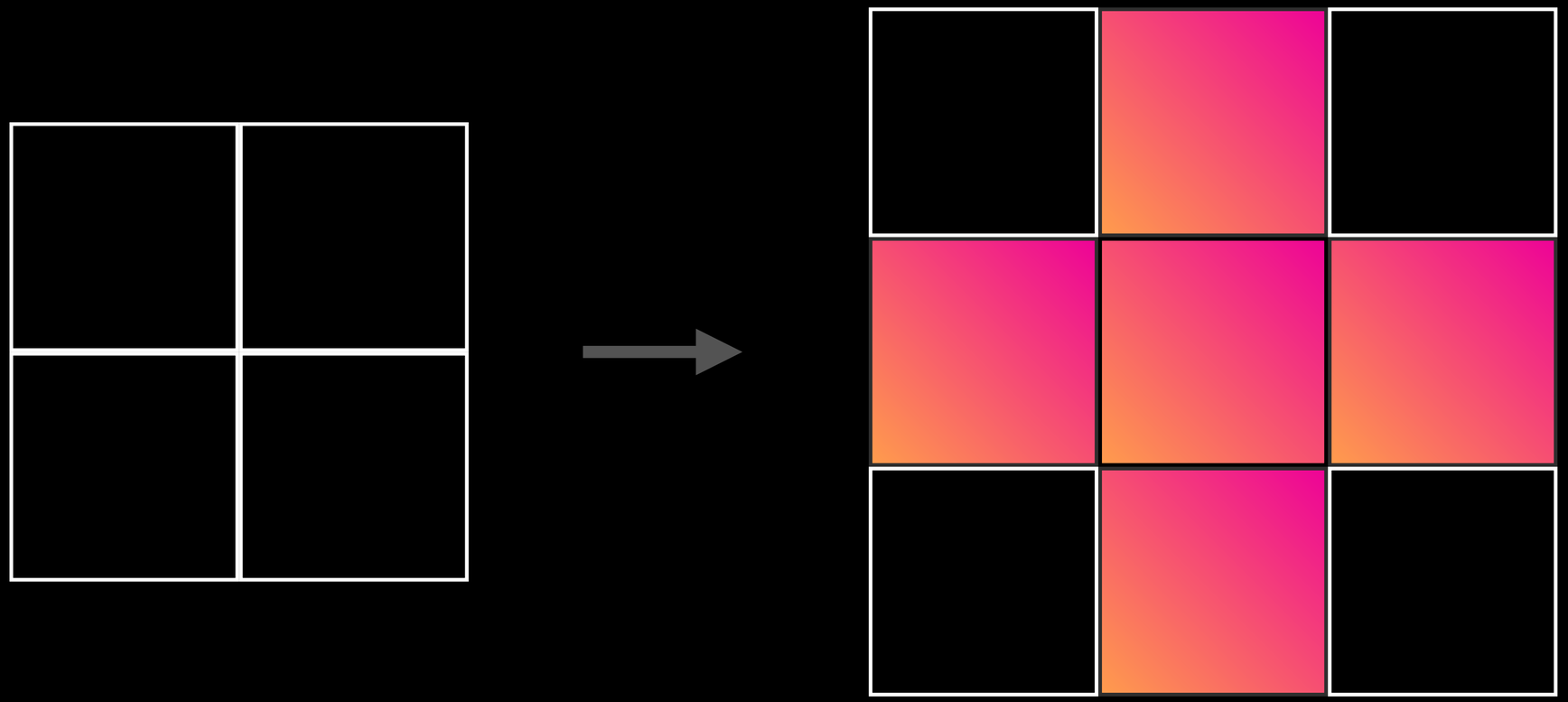


ORIGINAL IMAGE
2x2

INTERPOLATED IMAGE
3x3

Methods: Pre-Processing Pipelines

Resolution Upscaler

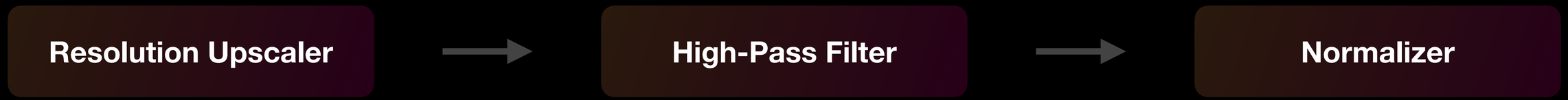


ORIGINAL IMAGE
2x2

INTERPOLATED IMAGE
3x3

INTERPOLATED VALUES

Methods: Pre-Processing Pipelines



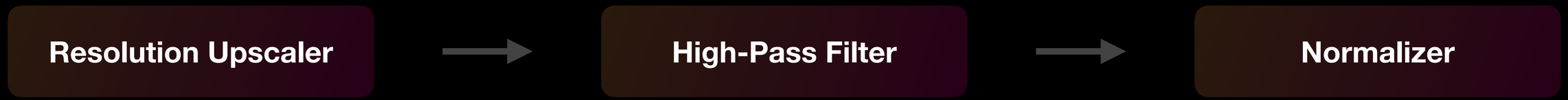
Methods: Pre-Processing Pipelines

High-Pass Filter

Filter out background noise: If pixel values were below a certain threshold value, they would be set to zero.

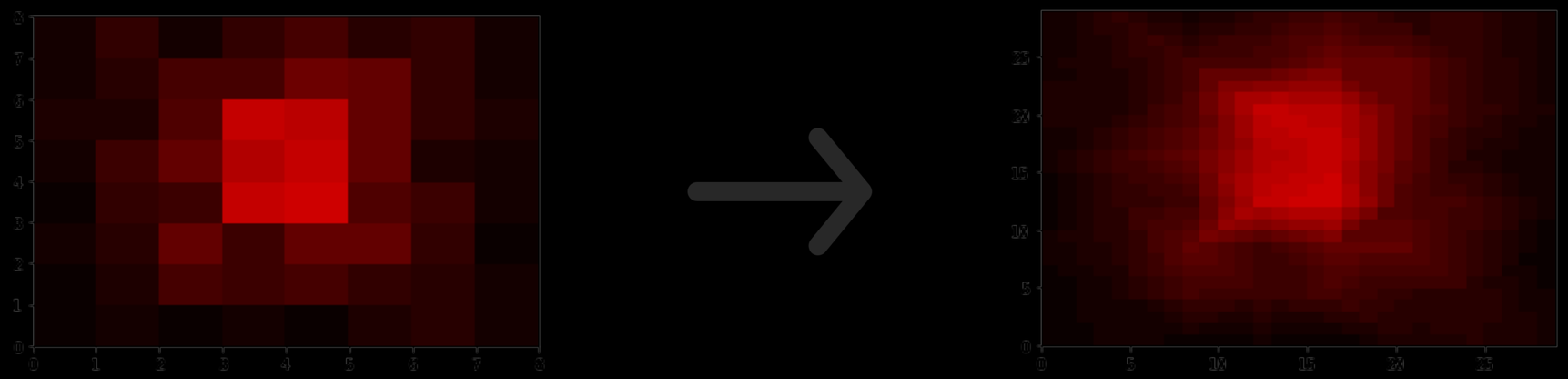
Intuition behind this was that this would help the model focus on learning the differences between the heat signatures of my class labels rather than subtle differences picked up in the background of my room

Methods: Pre-Processing Pipelines



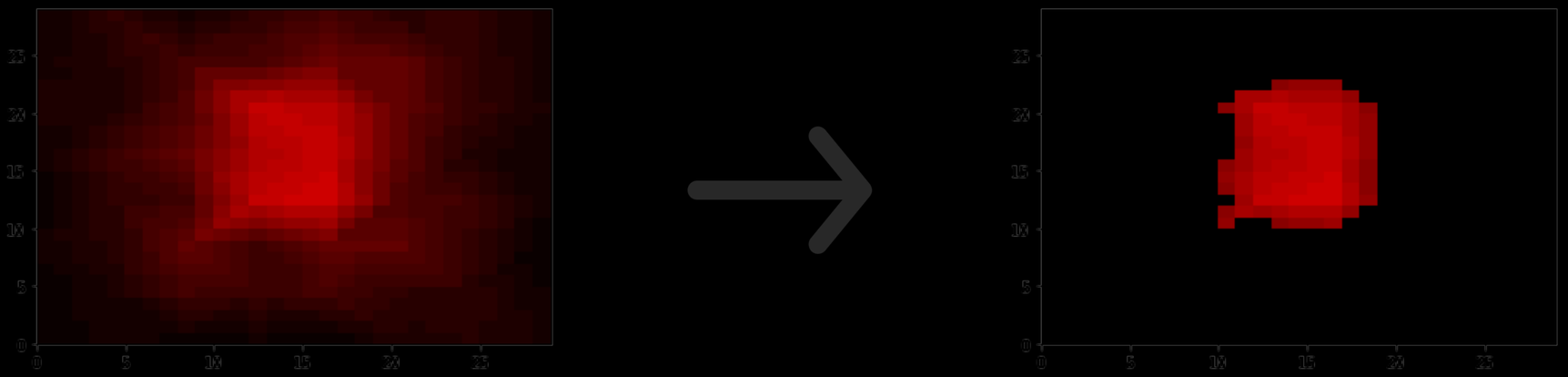
Methods: Pre-Processing Pipelines

Resolution Upscaler



Increasing the resolution of the original image by inserting 3 extra pixels in-between each pixel in both the x-y direction and using 2D linear interpolation to fill in the new pixels.

High-Pass Filter



Using a High-Pass Filter to allow the model to focus more on the signal's heat signature. Here, I used a threshold value of 80. The background noise was around 40-80 and foreground values were roughly around ~90 ±25

Model Type & Design: LeNet-5 CNN

INPUT
ScalingFactor=3 => 29x29

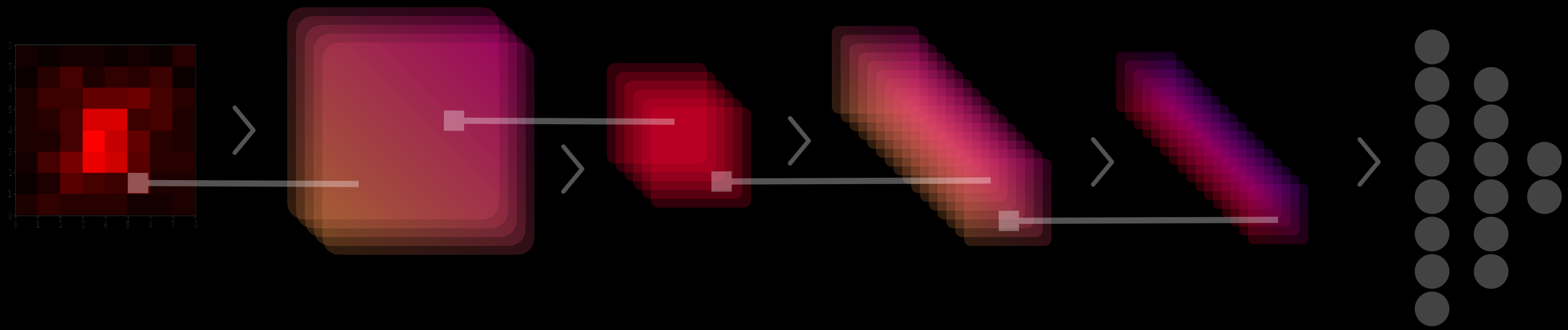
C1: feature maps x 6

S2: feature maps x 14

C3: feature maps x 16

S4: feature maps x 16

3 Dense, fully-connected layers
Last layer = **OUTPUT (SOFTMAX)**



Model Type & Design: LeNet-5 CNN

INPUT
ScalingFactor=3 => 29x29

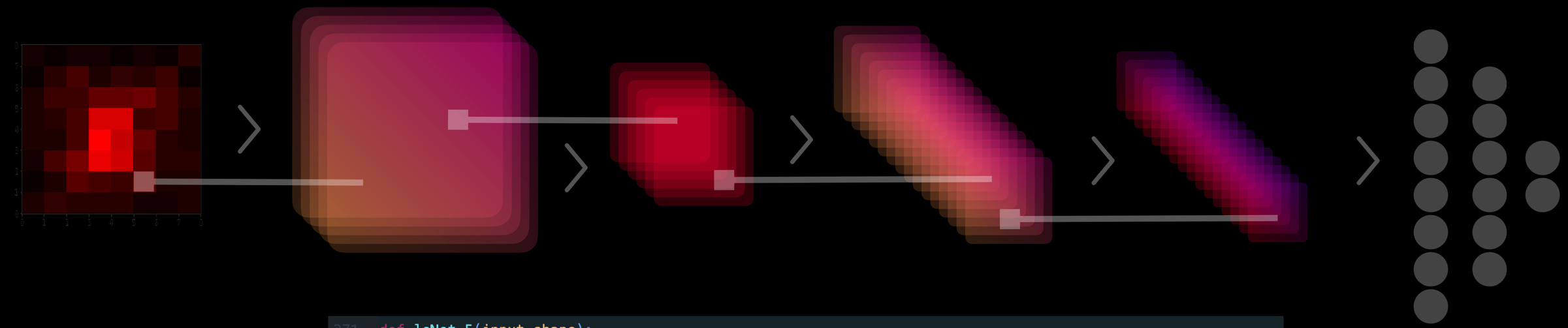
C1: feature maps x 6

S2: feature maps x 14

C3: feature maps x 16

S4: feature maps x 16

3 Dense, fully-connected layers
Last layer = **OUTPUT**
(SOFTMAX)



Keras Implementation:

```
271 def leNet_5(input_shape):
272     print("Input Layer Dimensions:",input_shape)
273     #Implementing LeNet-5 CNN:
274     model = tensorflow.keras.Sequential()
275     model.add(layers.Conv2D(filters=6, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
276     model.add(layers.AveragePooling2D())
277     model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
278     model.add(layers.AveragePooling2D(pool_size=(2,2),padding='same'))
279     model.add(layers.Flatten())
280     model.add(layers.Dense(units=120, activation='relu'))
281     model.add(layers.Dense(units=84, activation='relu'))
282     model.add(layers.Dense(units=2, activation = 'softmax'))
283 
284     #Compile model
285     model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
286     return model
```

Model Results: Accuracy

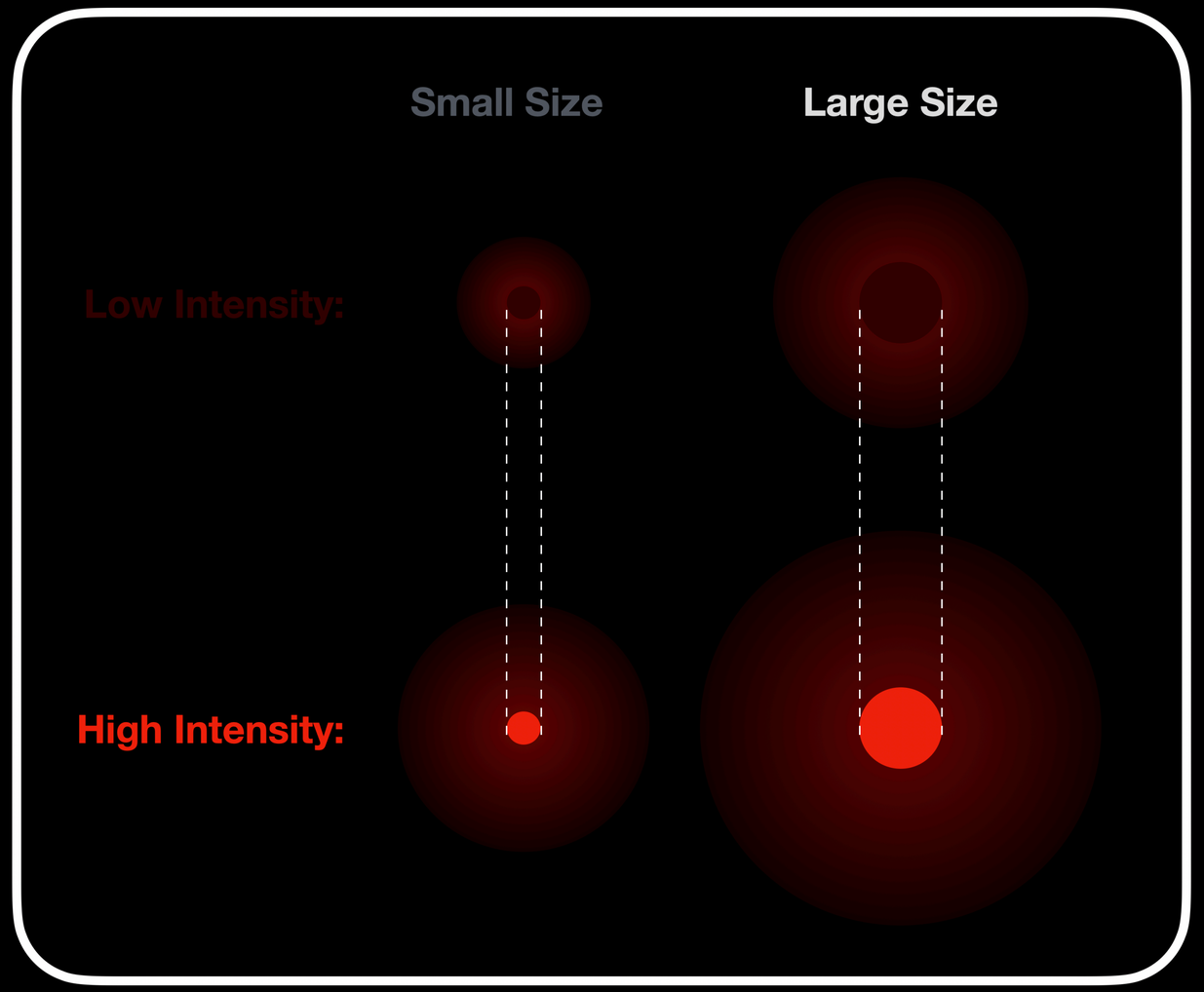
Unprocessed Images

Training Accuracy	Cross-Validation. Accuracy
74.4%	68.0%

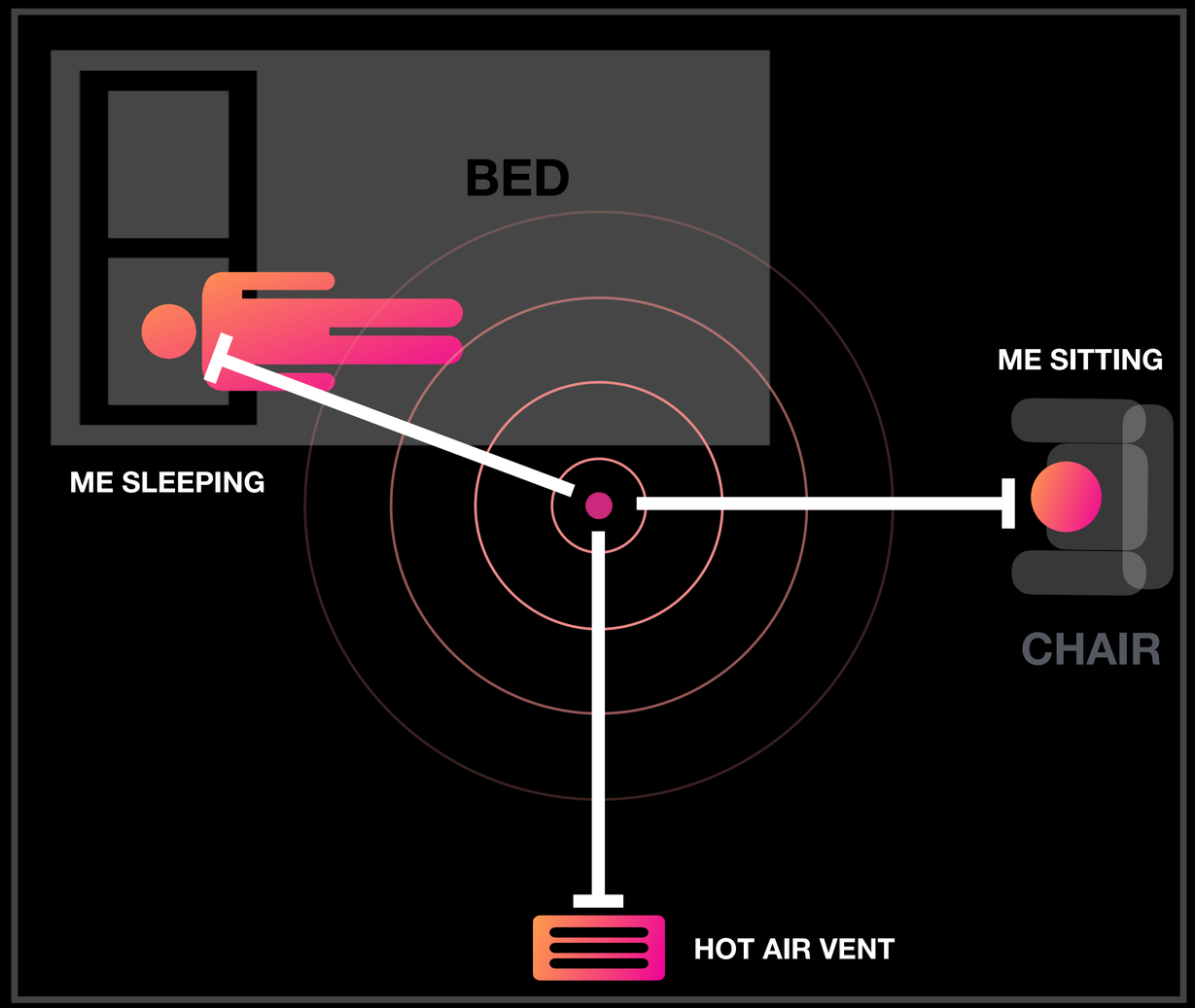
Preprocessed Images

Training Accuracy	Cross-Validation. Accuracy
100%	100%

Discussion: Suspicious Results?



Discussion: Experimental Setup…



Future Work

- In addition to exploring repeating the same experimental procedures at various distances, more class types should be included if a model can be shown to handle varying distances.

- Varying distances will most likely make it very difficult for the LeNet-5 model to learn effectively, meaning more complex architectures, or different approaches entirely, may need to be explored when trying to build a model than can handle different sized rooms, and, eventually, more positive classes.

- That being said, it has demonstrated that it works perfectly for my particular room — begging the question of whether such a sensor could be effectively trained for each room's surroundings as it is. Thus, this is another avenue that could be explored.

- For *my* room at least, I was able to deliver true occupancy detection, something off-the-shelf products have failed to do

Thank you