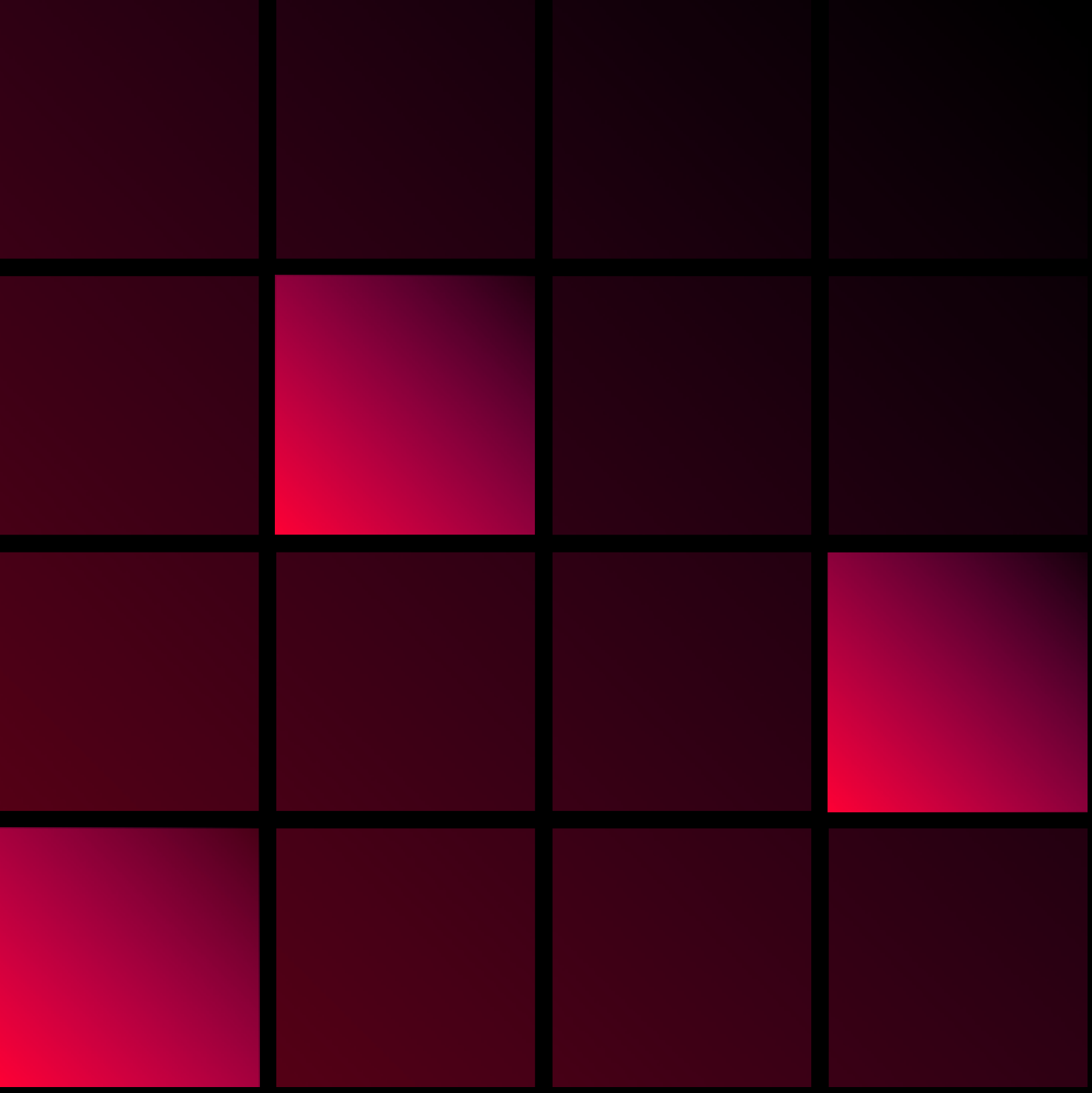


Detecting Human Presence in a Smart Home

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

Ryan Divigalpitiya | rdivigal@uwo.ca
CS 9860 Advanced Machine Learning
Western University

Introduction

Smart home technology has experienced rapid growth in the past two decades thanks to advancements in off-the-shelf products featuring Internet-of-Things capabilities and artificial intelligence-based functionalities.

A core requirement for delivering basic features in a smart home usually revolves around detecting the presence of residents, either on a room-by-room basis or on a broader scale, such as whether people are home or away from home. Typical off-the-shelf smart home products that can bring such functionality to a home usually utilize motion sensors to detect presence. However, the use of motion sensors for the purpose of detecting if a room is occupied often fails when the person occupying the room is stationary. Examples include sitting on a sofa and watching tv, or simply sleeping at night in their bedroom.

For a long time, smart home product companies have been keen on developing novel sensors and new detection algorithms to help remedy this problem — to try and build a true occupancy sensor that can robustly sense if a room is occupied or not, independent of the motion of the occupants. Currently, there are several promising sensor technologies that may can deliver such a solution but almost always at a very steep price. Such examples include high-sensitivity radars and Lidar sensors. As mentioned above, they are incredibly expensive, ranging somewhere in the hundreds of dollars for a single sensor unit up to thousands of dollars.

Another promising sensor type that could be used for motionless detection of occupancy are thermal-based sensors, as illustrated inFigure 1 below.

Figure 1. A company called FLIR manufactures high-resolution thermal cameras[1]

However, thermal sensors capable of producing such high-resolution thermal images are both expensive (ranging from hundreds to thousands of dollars) and also violate privacy since it has the ability to resolve spatial features enough to identify faces and what people are doing on camera — something that must be taken into account when installing smart home products into peoples' homes.

If we take a look at the cheapest thermal sensors available, there are two companies that produce such products — Omron[2] and Panasonic[3]. The Panasonic AMG8833 costs about $45, making it a much cheaper alternative[3]. However, their is a substantial resolution trade-off — these sensors only achieve a resolution of 8x8 pixels (viewing angles are around 60 degrees in the x-y axis)[4] as shown in Figure 2 below.

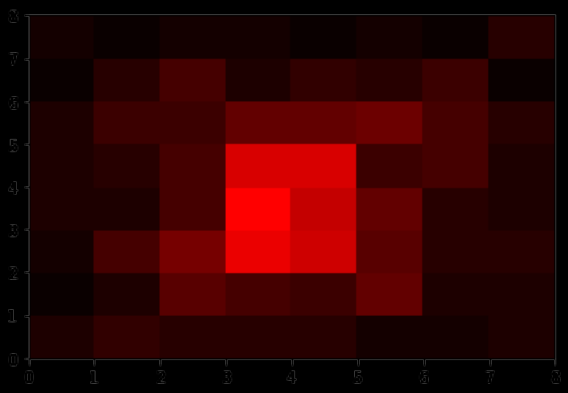


Figure 2. Example of what the Panasonic AMG8833 sees with a heat source at the centre of its line of sight. Compare the difference between Figure 2's resolution to the expensive FLIR-based imaging system in Figure 1! (Visualization done by Python and matplotlib library. Code found in Github link in the References section)

However, if we can somehow utilize these cheap thermal sensors to achieve our goal of delivering true occupancy detection, their cheap cost allows it to be financially feasible to scale the deployment of these sensors throughout an entire house, transforming it into a true smart home with true occupancy detection.

So what is stopping the industry from doing so? A big reason is **false-positive signals** — a true positive signal in this context occurs when the sensor detects a heat source and that heat source happens to be a human occupying the room. A false positive signal occurs when that heat source is not a human, such as a hot air-vent during winter, a hot electronic device, or a hot appliance.

So, the focus of my project will be attempting to answer the following question:

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?

Experimental Method

In order to keep the scope of this project within the timeframe of this semester, I decided to try and build an AI model that can distinguish between a human's heat signature and the most common false-positive (FP) signal found in a home: a hot air-vent. Thus, I will not explore being able to discriminate against the many other FP signals out there such as hot electronics, appliances, etc. This is also a good starting point because it initially constrains the amount of variables in the experimental setup and, thus, allows for a more elaborate project in the future that explores other possible FP discriminations *if* the initial experiments prove to be successful.

Before explaining my experimental setup, I need to first clarify a question that may popup in the reader's head when reading this — can you take motion into account when trying to discriminate between stationary warm objects versus humans that initially move and then sit down at their destination? The answer is yes, *if*, you have multiple thermal sensors all mounted on the ceiling, pointing down, such that, combined, they are able to cover the entire square footage of a room. One sensor alone does not have the field-of-view to cover an entire room and, my objective is to try and detect occupancy of an entire room. If you want to cover the entire ceiling with a grid of these sensors (or have multiple sensors on each wall, which would not look aesthetically pleasing), such an installation would be costly with the numerous wiring and fixtures required. Thus, my goal is to try and use a *single* sensor, mounted on a motor capable of swivelling the sensor 360 degrees; this gadget would then be mounted on the ceiling at the centre of the room with the sensor tilted at a 45 degree angle from the ceiling's plane, giving it the ability to slowly scan the entire room by rotating in a 360 degree arc. Once it detects a heat source, the motor will center the sensor's line of sight onto the heat source and attempt to evaluate what kind of heat signature it is: a human or a false-positive signal (a hot air-vent in this case). Thus, we cannot take a heat signature's movement into account. Such a setup would be cheap and deliver the functionality of being able to sense occupancy of a full bedroom — such as my bedroom, and, thus, I have built my experimental setup in my own bedroom to conduct my experiments. However, to keep my work simple, I simply mounted my sensor on a pole and hoisted it near the ceiling while the sensor was fixed in a 45-degree angle pointing down — the sensor's line of sight was 45 degrees to the plane of the ceiling — shown in figure 3, 4 and 5 below. I could then manually rotate the sensor myself by rotating the pole to aim the sensor at desired locations in my room.

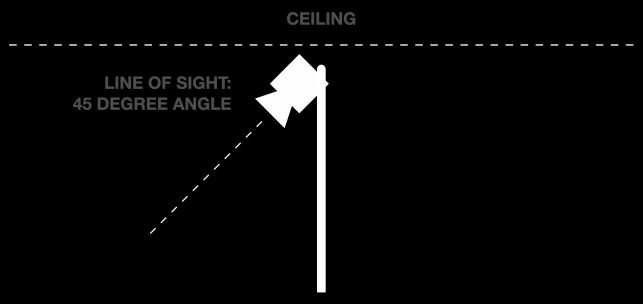
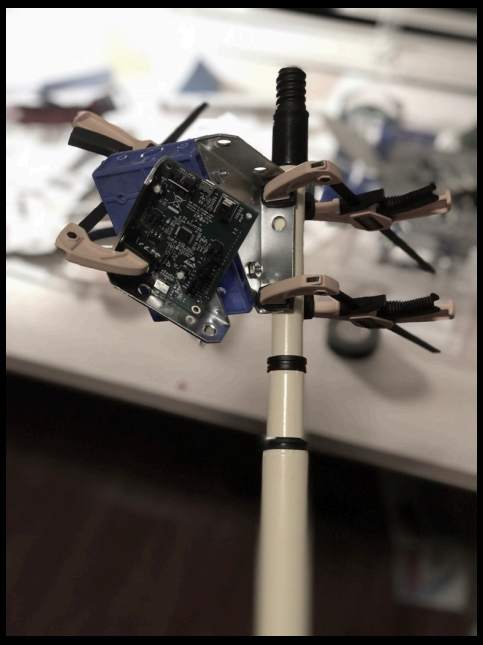


Figure 3. Cross-sectional diagram of sensor mounted on a pole. Photos shown in Figure 4 and 5.



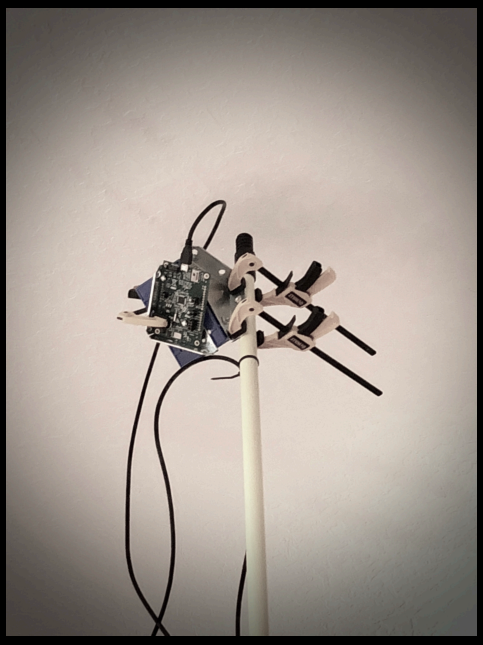


Figure 4 and 5. Figure 4 (left) shows how I mounted the sensor at a 45-degree angle onto a pole. Figure 5 shows the pole positioning the sensor near the ceiling. I can then manually rotate the pole to aim the sensor at locations in my bedroom. The pole was held in place with c-clamps and a chair.

As shown in Figure 4 and 5, I purchased a prototyping AMG8833 thermal sensor from Panasonic and, using software provided by Panasonic, was able to connect it to my laptop in order to read the incoming data via Python. I then positioned the sensor in the centre of my bedroom and pointed it at three locations in my room:

1) My bed, where I napped for 10 minutes and recorded myself on the thermal sensor
2) The hot air-vent (after turning up my thermostat to 25 degrees for 10 minutes)
3) The chair in my room, where I sat reading a book for 10 minutes

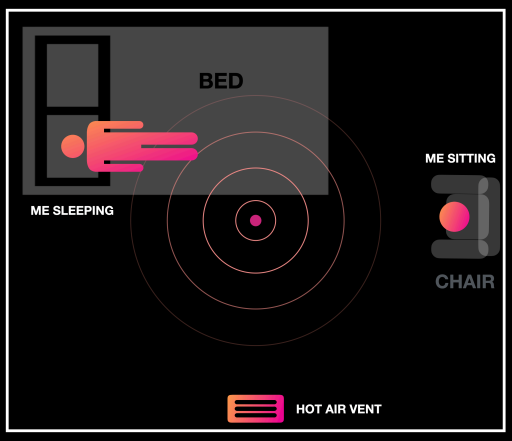


Figure 6. Bird's eye view schematic of my bedroom.
At the centre: 360 degree rotating sensor placed on ceiling at centre of room.
My experiments consisted of mounting my sensor on a pole that I could then manually rotate myself to aim its line of sight at either the bed, the hot air-vent or the chair. I then took ~ten minute readings from each of these locations in my room.

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**

Class labels were **1 = human** (me, true positive) and **0 = air-vent** (false-positive). The resulting dataset, when all 3 sets are combined into one, was 18,000 images in total to experiment with when training/evaluating my models. Examples of instances of each set are shown below.

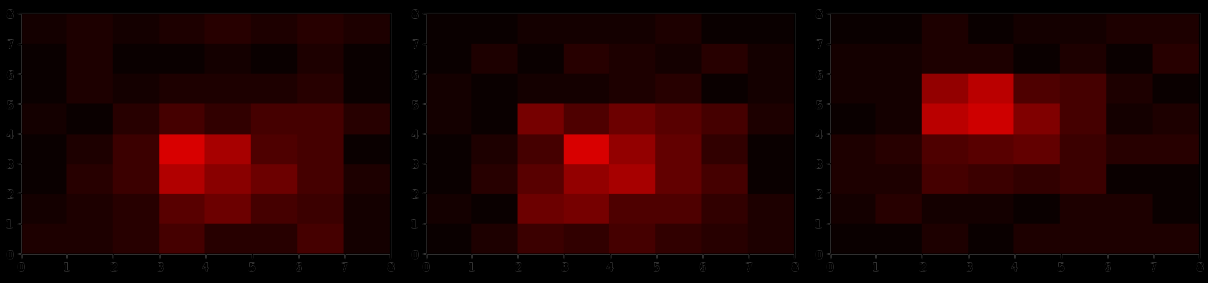


Figure 7. Three instances of images of me lying in bed.
Note that my head is really the only skin that's exposed since I was wearing pants and long-sleeves, thus, my heat signature has a single centroid with heat dissipating away from it.

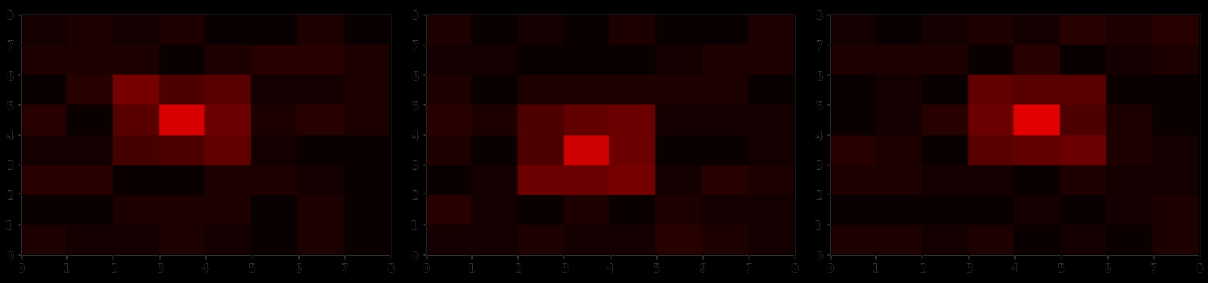


Figure 8. Three images of me sitting in my chair
Note that my head is really the only skin that's exposed since I was wearing pants and long-sleeves, thus, my heat signature has a single centroid with heat dissipating away from it.

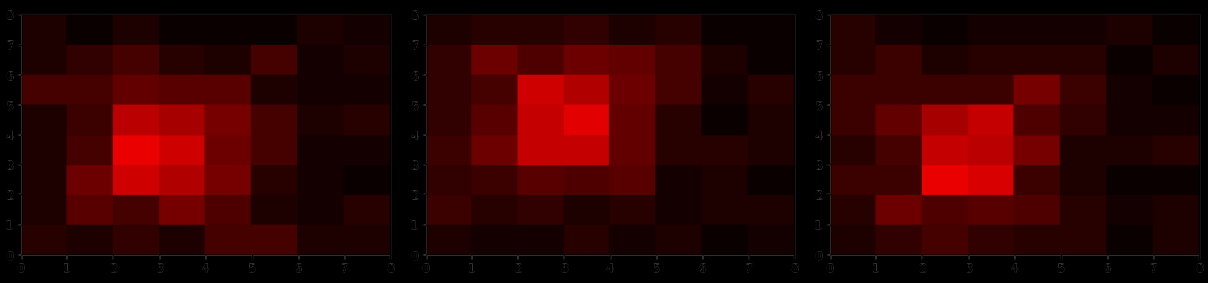


Figure 9. Three images of air vent. I slightly rotated the pole every now and then to simulate the sensor using a rotational motor to lock on the signal, thus, we have some spatial variation in the centroid shown above.

Pre-Processing Pipelines

Before I built my model and train/evaluated on my data, I first considered what type of pre-processing would aid in the performance of my model. We can intuitively guess that the low-resolution of these images may prove difficult for any model to learn from. So the first thing I tried was to build a resolution up-scaler that could increase the resolution of the images by using 2-dimensional linear interpolation — illustrated in Figure 10 below.

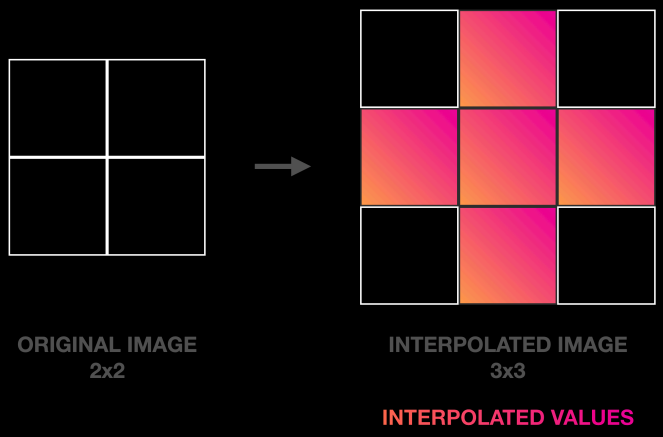


Figure 10. An example of 2D interpolation on a 2x2 image where we upscale by 1 new pixel in the x and y direction. First, we must calculate the dimensions of the new image, then populate it with the original data at the correct coordinates in the new image. Finally, we use 2D linear interpolation to fill in the missing pixel values, highlighted in colour above.

By using the formula,

$$ImageOriginal_{y\text{-}dim} \times (scalingFactor + 1) - scalingFactor$$

where,
$ImageOriginal_{y\text{-}dim}$ = y-dimension of original image
scalingFactor = the number of new pixels you want to insert in the x and y direction.
scalingFactor would equal 1 in the example shown in Figure 10.

By combining NumPy's `.mgrid()` function with the above, I was able to build my own custom resolution upscaling pipeline where I could specify how many new pixels I wanted in the upscaled image. This could be treated as a parameter that could technically be tuned in order to see how it impacts the model performance (while being weary that increasing the resolution too high may yield unrecognizable/inappropriate training data along with long training times).

Next, I decided to implement a second pipeline that functioned as a High-Pass Filter (HPF). My HPF would filter out useless background noise in order to strengthen the heat signatures foreground signal — my 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, thus, improving the model's performance. The HPF pipeline I built allowed me to specify a threshold which could in turn also be another parameter tuned for better performance. Finally, I applied one last pre-processing pipeline which was simply normalizing all pixel values between 0 and 1.

Pipeline Output

Before going over the model design and results, let's take a look at the output of each pipeline (minus the normalization pipeline).

For the resolution upscaler pipeline, examples of scaling each x-y dimension by 3 pixels are shown below in figure 11.

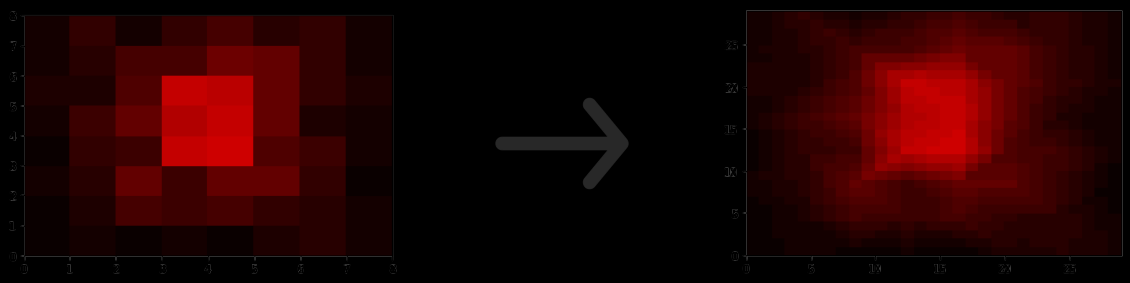


Figure 11. Increasing the resolution of the original image by inserting 3 extra pixels in-between each pixel in both the x-y direction and using 2D linear interpolation to fill in the new pixels.

For the High-Pass Filter pipeline, examples of filtering out background noise is shown below in figure 12.

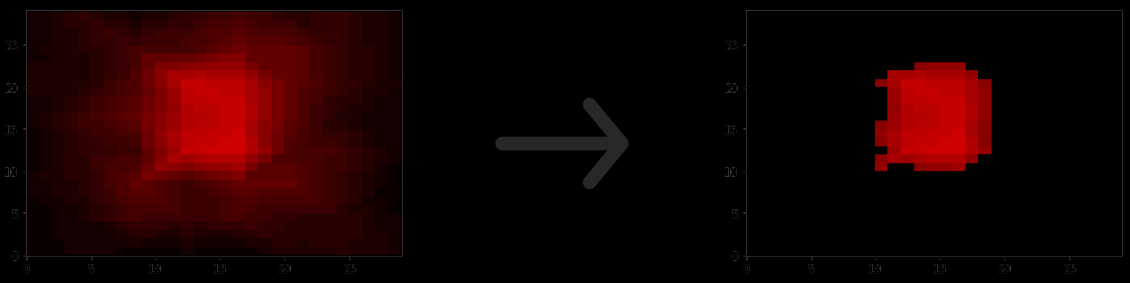


Figure 12. Using a High-Pass Filter to allow the model to focus more on the signal's heat signature. Here, I used a threshold value of 80. The background noise was around 40-80 and foreground values were roughly around ~90 ±25

Finally, after normalizing pixel values between 0 and 1, I was able to pass the pre-processed data to my LeNet-5 model for training.

A scaling factor of 3 was using for upscaling the images and a threshold value of 80 was used for the high-pass filter. The threshold value was determined by manually trying different values in my HPF pipeline and eyeballing the resulting filtered images in terms of what looked like the best threshold value to use. Scaling factor of 3 was arbitrarily chosen as a starting point. Both parameters could be tuned if the model performance turned out to initially be poor.

Model Type and Design

Distinguishing between the heat signature of a human versus an air-vent presents itself as an image recognition task. Thus, the first type of model I wanted to try was a Convolutional Neural Network (CNN) since CNNs are a type of neural network architecture that is highly optimized for this task type. However, there are numerous types of architectures one could experiment with so I decided to research which types of architectures are commonly used for my problem. After learning about the classic LeNet-5 architecture and how it performed extremely well on images of handwritten digits that had poor resolution (32x32)[5,6], I experimented with this model first. The network architecture is shown below, in **figure 13**.

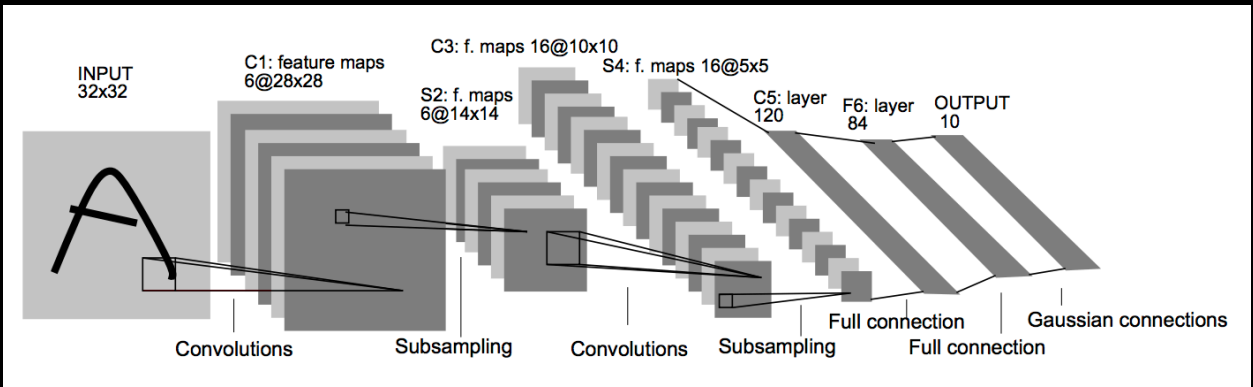


Figure 13. LeNet-5's architecture by LeCun et al., 1998[6].

Note that the subsampling being done in this architecture is Average Pooling which is rare these days (Max Pooling is usually used instead). I used Keras, with TensorFlow as its backend, to implement the above architecture, shown in figure 14 below.

```
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
```

Figure 14. LeNet-5's architecture by LeCun et al., 1998[6] implemented in Keras.
Input shape will be the dimensions of the pre-processed images (remember we are upscaling them before hand, thus, the dimensions will be higher than 8x8) and we have one channel per pixel.

Model Results

For training the LeNet-5 model, I used an epoch of 10 iterations, the 'adam' optimizer, and 'categorical_crossentropy' for loss. For comparison purposes, I fed the unprocessed image data first, and then fed the pre-processed image data. I report the training and cross-validation accuracy, as reported by Keras when training the model, which simply measures the proportion of predictions that were correct.

Unprocessed Images

Training Accuracy	Cross-Validation. Accuracy
74.4%	68.0%

Preprocessed Images

Training Accuracy	Cross-Validation. Accuracy
100%	100%

Figure 14. Model performance for unprocessed data vs. processed data

I then created a test set and my model also scored 100% accuracy on the test set. Since 100% accuracy was seemingly achieved, I did not bother with more accuracy measurement methods such as confusion matrices, ROC curves, etc.

Discussion

At first, I was highly suspicious at how accurate my model was performing, but we have the much poorer unprocessed dataset's performance to compare with as a sanity check. After deeper thought, I realized that my experimental setup in my room, along with my pre-processing pipeline, has greatly made the task of distinguishing between the two heat signature classes very easy. Allow me to explain.

A heat signature can be broken down into two characteristics: its intensity at the centroid (assuming one heat source within the sensor's view) and the spread of the heat signature. The size and shape of the heat source will determine how much the heat signature registered on the sensor will spread out. These two characteristics are shown below in Figure 15.

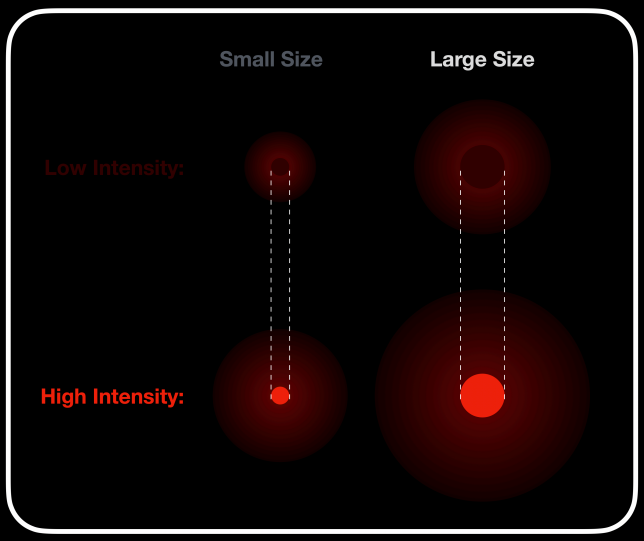


Figure 15. The two characteristics of a heat signature: Intensity and Size/Shape

However, this only holds true if *distance* from the heat source is fixed — if distance varies, then we have a third attribute that influences the heat signature. In my experimental setup in my bedroom, I had set up the sensor such that it is equidistant between me laying in bed, the air-vent and me sitting in my chair. However, if the chair had been a bit closer or farther, its not hard to imagine the heat signature of me sitting in the chair changing dramatically — it could even morph into a signature similar to shape and intensity of the hot air vent for all we know. Now, in experimental design, we should try to constrain variables that are not being investigated as much as possible — and this distance variable was rightly constrained — however, in doing so, I believe I made it too easy for my LeNet-5 model to learn the differences between the heat signature of me versus the heat signature of a hot air-vent given; in other words, my model has overfitted itself to my room's surroundings.

What this means is that the 100% accuracy performance should be taken with a grain of salt, mainly because this type of solution has not at all been proven to be scalable to other rooms of my house. However, this project has been, so far, a good foundational starting point — I have demonstrated that, with the distance variable constrained, the LeNet-5 is very good at distinguishing between me and a false-positive air-vent. If this wasn't shown to be true, this project would essentially be dead in the water and there would be no justification to try and do further experimentation with varying distances.

Future Work & Outlook

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. Other class types would include other common false-positive signals found in the typical home: hot electronics, appliances, space heaters, etc.

However, my intuition tells me that varying distances will make it very difficult for the LeNet-5 model to learn effectively. Thus, more complex architectures, or even more advanced AI algorithms, may be needed to be explored when trying to build a model than can handle different sized rooms, and, eventually, more positive classes.

That being said, if I wanted to go ahead and keep this thermal sensor as a permanent installation in *my* bedroom, I could — as it has demonstrated that it works perfectly for my particular room. This begs the question of whether such a sensor could be effectively trained on a per room's basis, learning the individual surroundings in each one. In other words, this is another line of questioning that could be explored in future work.

In conclusion, the implications of installing this sensor as a permanent sensor in *my* room would allow me to tinker with integrating the sensor with my wifi network and then integrating it with various smart home products I have in my home, such as smart lights, speakers and thermostats. I could have my lights automatically turn off when I'm not there, saving energy, but won't have to worry about them turning off when I'm stationary, reading in my chair — thus, delivering true smart home functionality that can't be done with existing "occupancy sensor" products bought off the shelf.

Thank you for your time and I hope you enjoyed reading about my project.

References

1 https://www.flir.com/discover/public-safety/thermal-imaging-for-detecting-elevated-body-temperature/

2 https://components.omron.com/product-detail?partNumber=D6T

3 https://www.digikey.ca/en/products/detail/panasonic-electronic-components/AMG8833/5825302

4 https://media.digikey.com/pdf/Data%20Sheets/Panasonic%20Sensors%20PDFs/Grid-EYE_AMG88.pdf

5 https://www.jeremyjordan.me/convnet-architectures/

6 Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, november 1998.

Github Link:

github.com/RyanDivigalpitiya/CS9860FinalProject