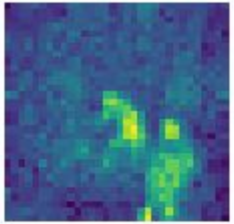
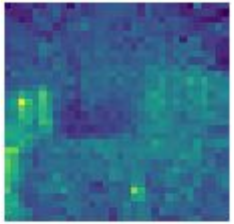
IAS0360 Final Project - Topic 2:

Thermal Image Classification





presented by: Felix, Omar, Tobias

Agenda

- **General Workflow**
- **Different Model Approaches**
  - **Semi-supervised clustering**
  - **Squeezenet**
  - **Gradient Boosted Trees**
- **Conclusion**

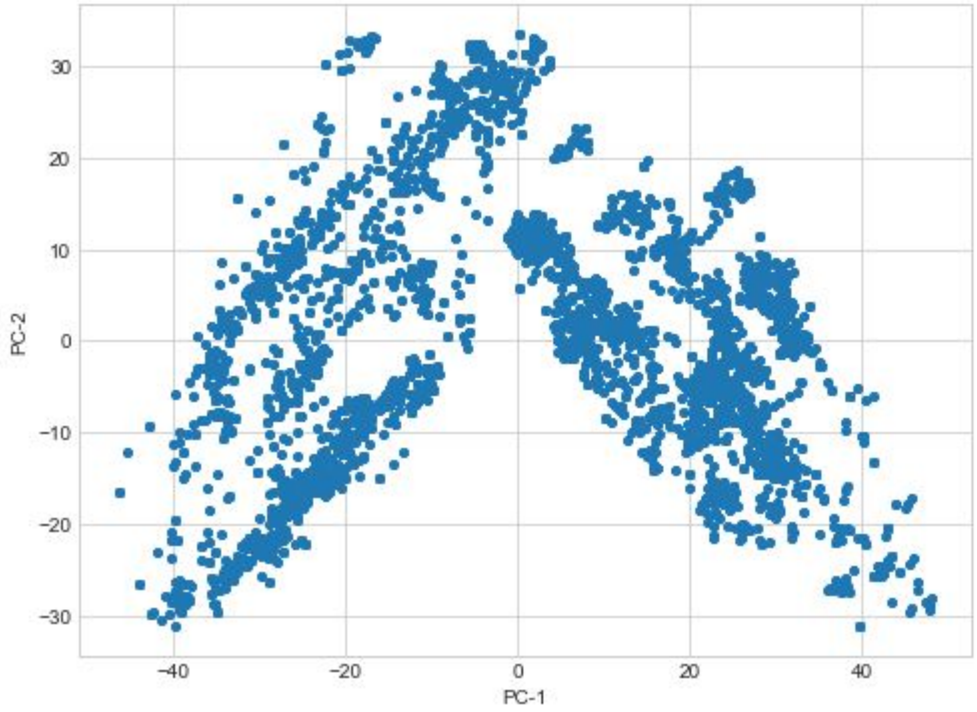
Workflow - CRISP DM

Cross Industry Standard Process for Data Mining

- Business Understanding
- Data Understanding
- Data Preparation
- Modelling
- Evaluation
- Deployment

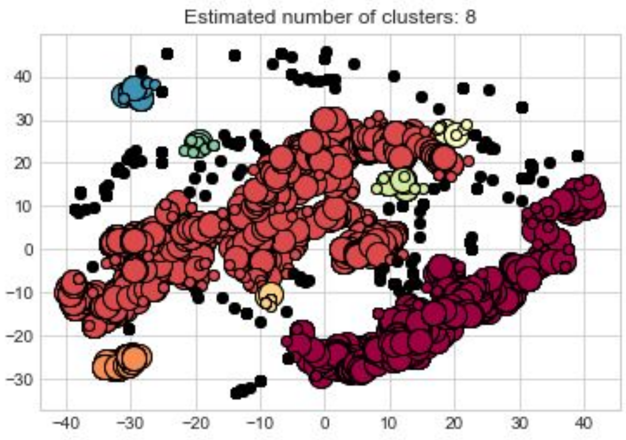
Data Exploration (1)

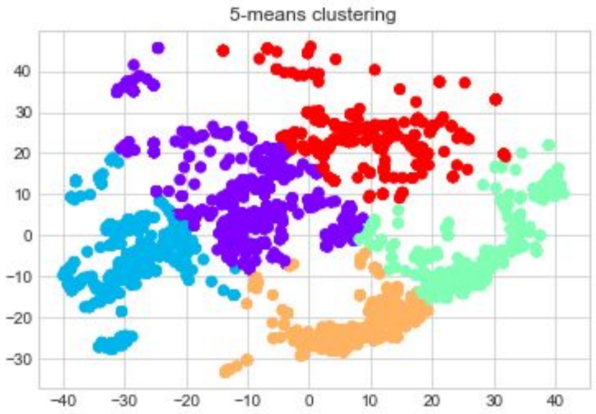
- C088 & 3078 sensor
    - 32x32 dimensions
    - PCA-2
    - 35% explained

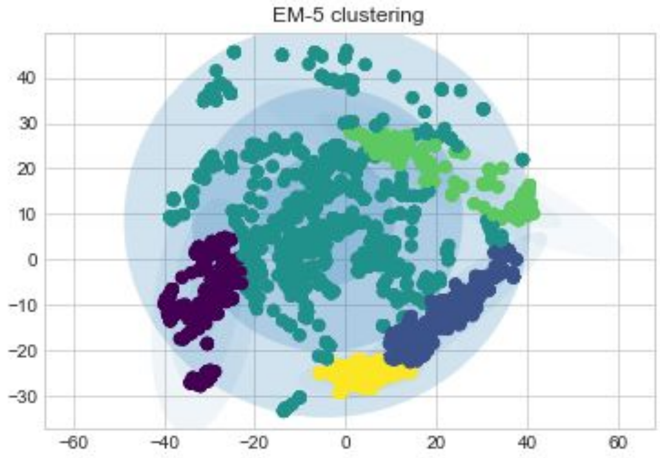


Data Exploration (2)

- Exploring sensor C088:
  - DBSCAN
  - EM
  - K-means

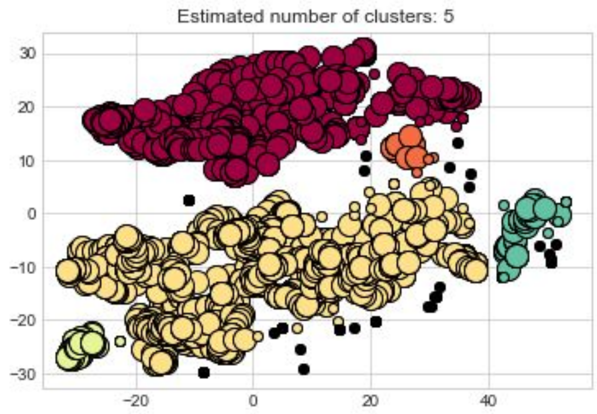


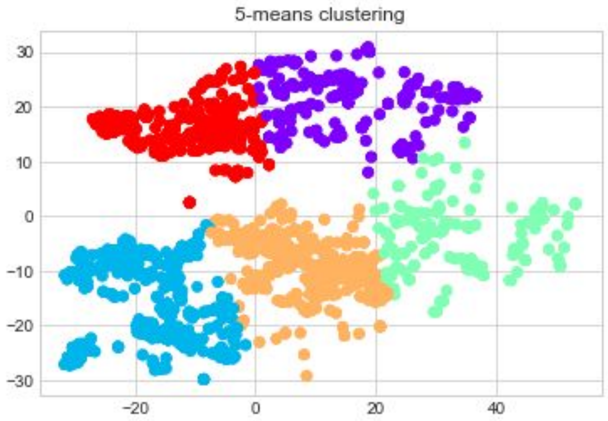


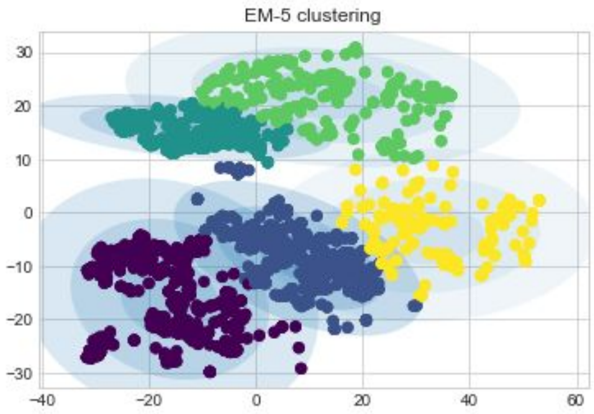


Data Exploration (3)

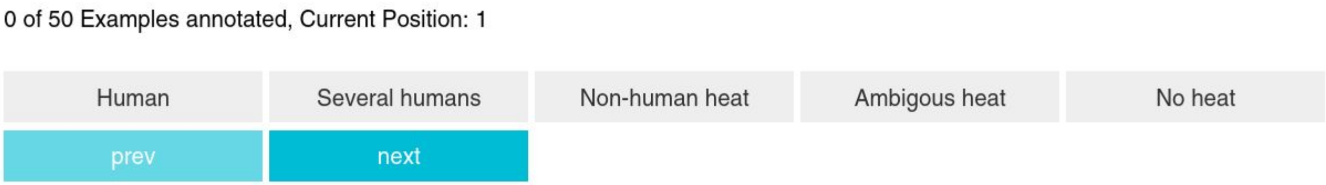
- Exploring sensor 3078:
  - DBSCAN
  - EM
  - K-means



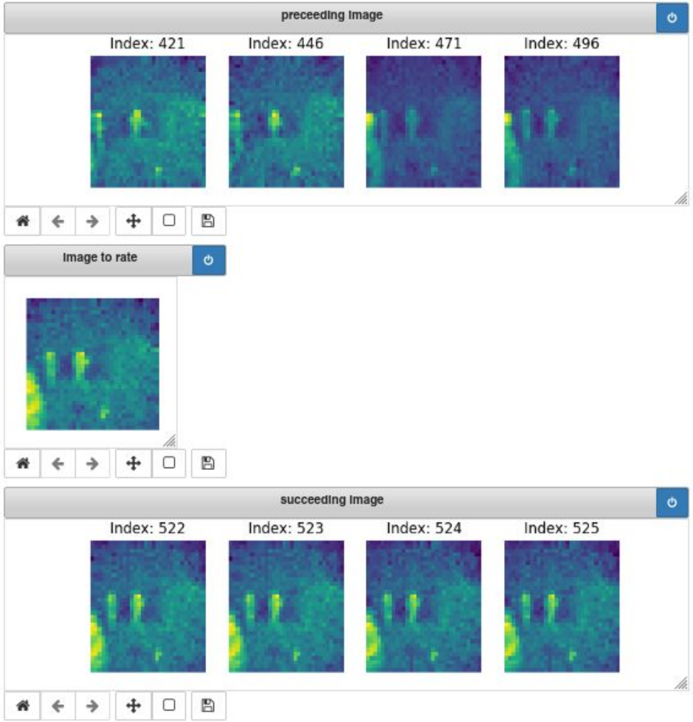




Labelling

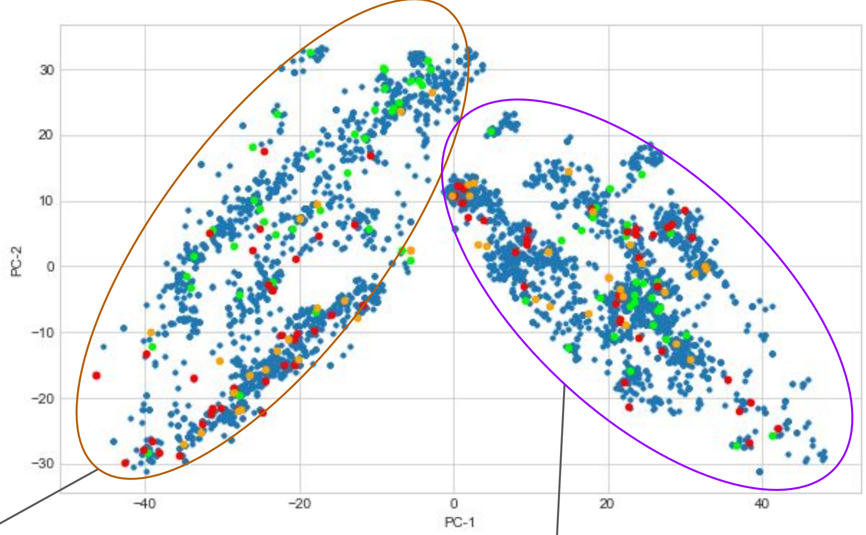


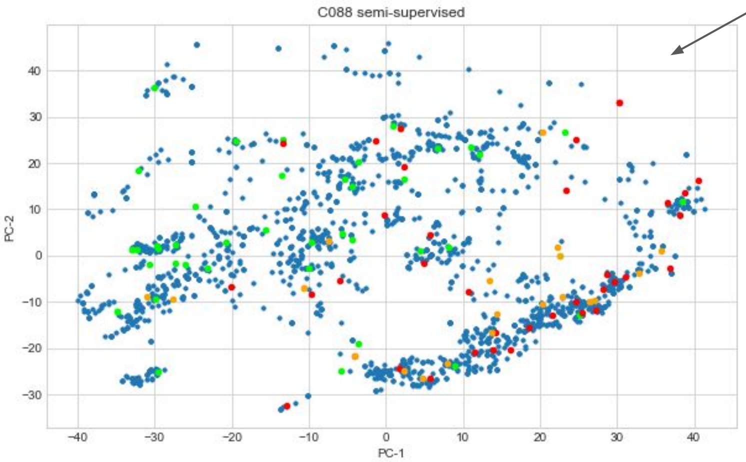
- Jupyter Notebook
  + Pidgeon XT
  https://bit.ly/3qOtV2n
- Splits:
  4 x 50 = 200 samples
- Labelled
  - 1 individually each
  - 1 badge collaboratively

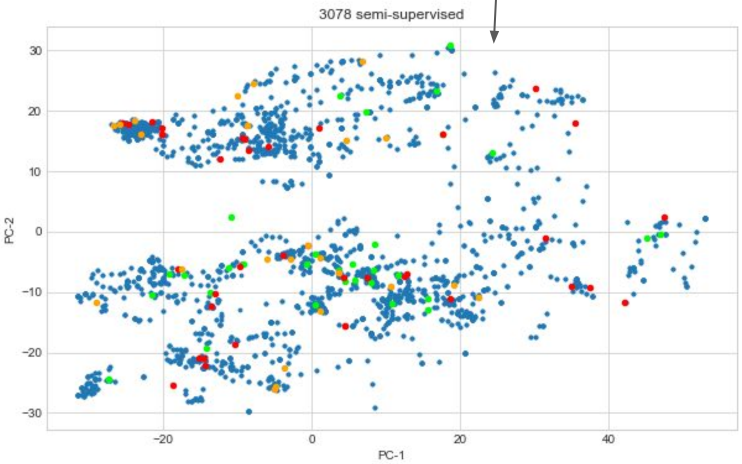


Data Exploration (4)

- Semi-supervised
  - Human | Several humans (green)
  - Non-human | No heat (red)
  - Ambiguous (orange)
- **DBSCAN yields best insights**



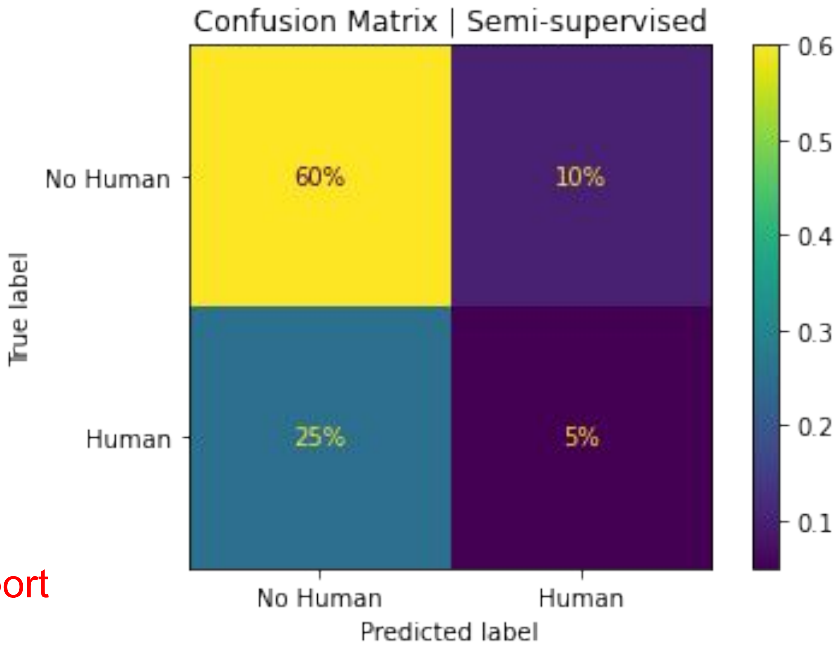




Semi-Supervised Classification

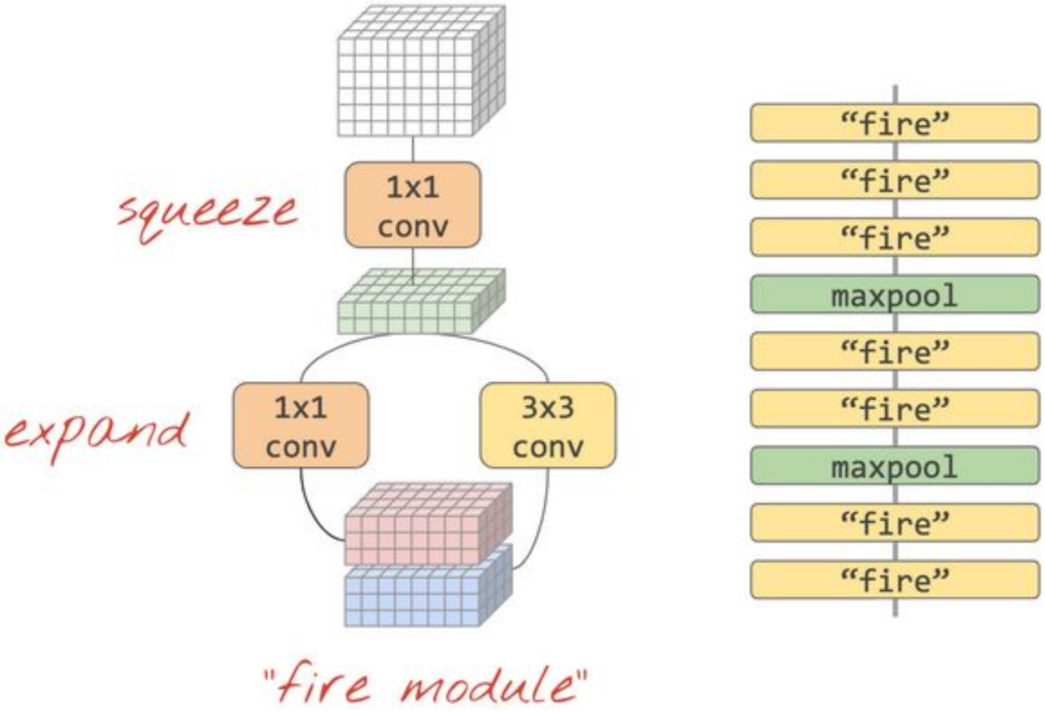
- Semi-supervised - Test Accuracy: 65 %
  - 180 Training, 20 Test Samples
- Self-trained classifier
  - Support Vector Classifier (SVC)
  - Other models possible (OUT OF SCOPE)

Train 180 → Convert 10.000 → Predict 20



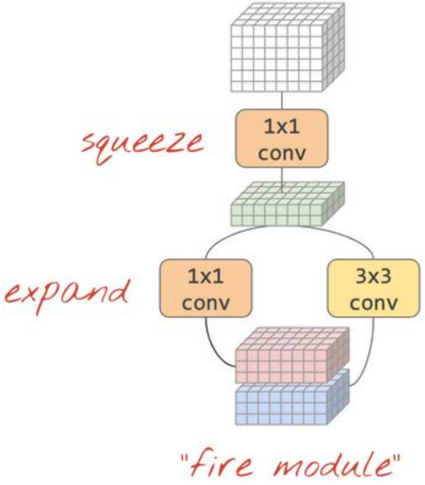
Problems: Bias in 180 labels & No sklearn/ONNX support

Squeezenet - CNN Approach



https://bit.ly/3nB1Ize

Squeezenet - Fire Module



- Leveraging Keras **functional** model approach:

```
def fire(x, squeeze, expand):
    y  = tf.keras.layers.Conv2D(filters=squeeze, kernel_size=1, activation='relu', padding='same')(x)
    y = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y)
    y1 = tf.keras.layers.Conv2D(filters=expand//2, kernel_size=1, activation='relu', padding='same')(y)
    y1 = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y1)
    y3 = tf.keras.layers.Conv2D(filters=expand//2, kernel_size=3, activation='relu', padding='same')(y)
    y3 = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y3)
    return tf.keras.layers.concatenate([y1, y3])
```

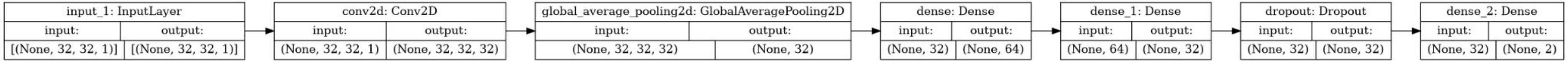
https://bit.ly/3nB1Ize

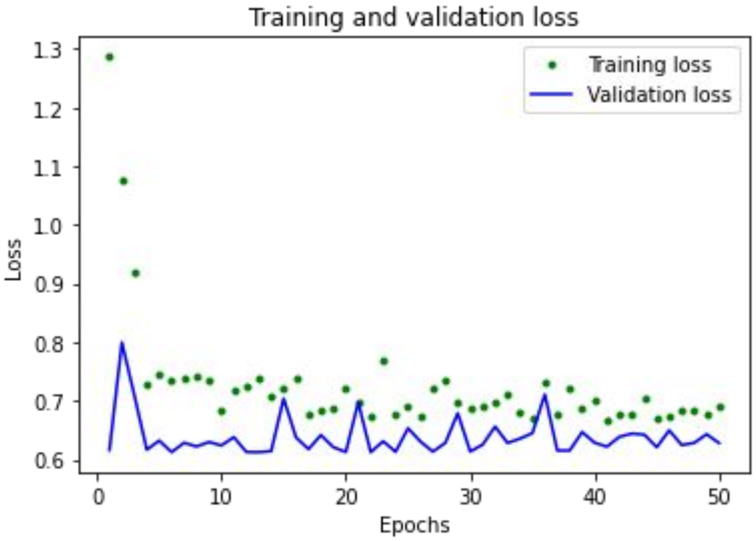
Squeezenet - Complete Model

```
x = tf.keras.layers.Input(shape=[*IMAGE_SIZE,1]) # input is 32x32 pixels RGB
y = tf.keras.layers.Conv2D(kernel_size=3, filters=32, padding='same', use_bias=True, activation='relu')(x)
y = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y)
y = tf.keras.layers.Dropout(0.2)(y)
y = fire_module(24, 48)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(48, 96)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(64, 128)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(48, 96)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(24, 48)(y)
y = tf.keras.layers.GlobalAveragePooling2D()(y)
y = tf.keras.layers.Dense(2, activation='softmax')(y)
```

https://bit.ly/3nB1Ize

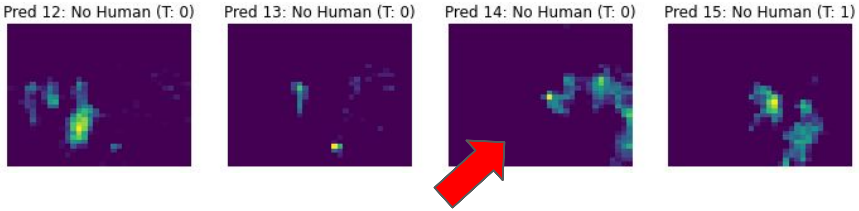
Baseline CNN





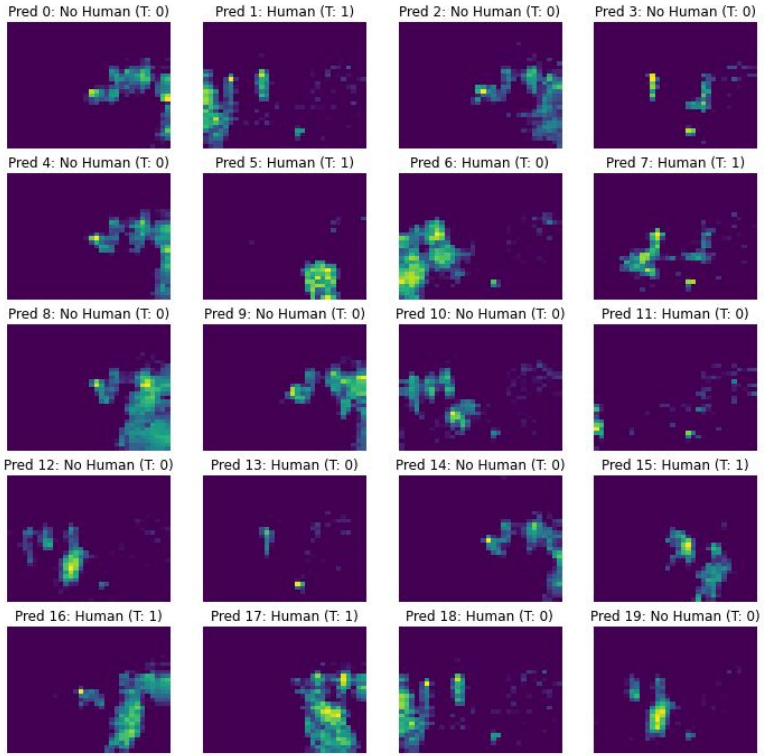
Epoch 50/50 - 0s 29ms/step
loss: 0.6900 - **accuracy: 0.6062**
val_loss: 0.6282 - val_accuracy: 0.7000

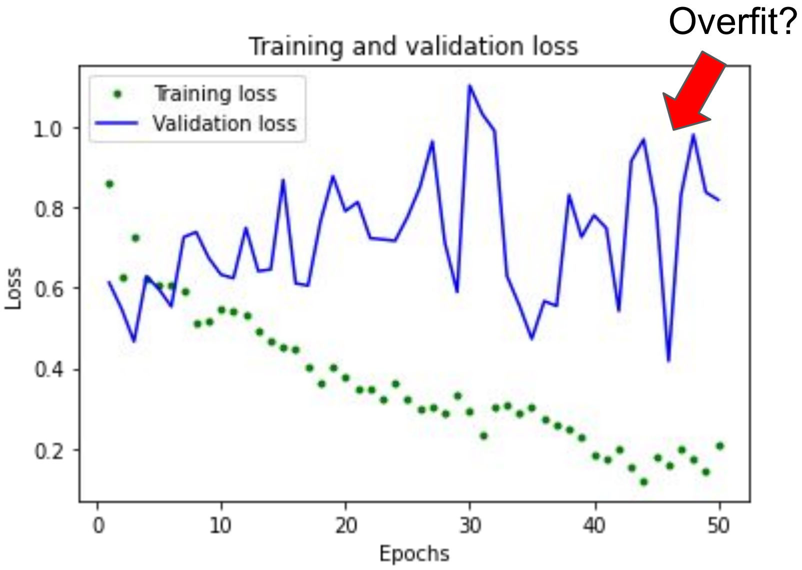
Test loss: 0.62011 / **Test accuracy: 0.69999**



Lowpass Threshold at 15.0

Squeezenet CNN





Epoch 50/50 1s 51ms/step
loss: 0.1196 - **accuracy: 0.9563**
val_loss: 0.6548 - val_accuracy: 0.8000

Test loss: 0.79038 / **Test accuracy: 0.75**

Squeezenet Implementation (1)

- Issues with STM32 'shrinking tool'

 # stm32ai quantize -q cfg_squeezenet-binary.json
Neural Network Tools for STM32AI v1.5.1 (STM.ai v7.0.0-RC8)

-- Loading/Initializing dataset
 X_test   (20, 32, 32, 1)
 y_test   (20, 2)
-- Loading/Initializing dataset - done (elapsed time 0.002s)
-- Loading original model
-- Loading original model - done (elapsed time 0.457s)
-- Testing original model
 Original model - test accuracy (loss): 0.75 (0.202)
-- Testing original model - done (elapsed time 0.664s)
The original network has several branches.
We don't support quantization of these topologies.
Process finished.

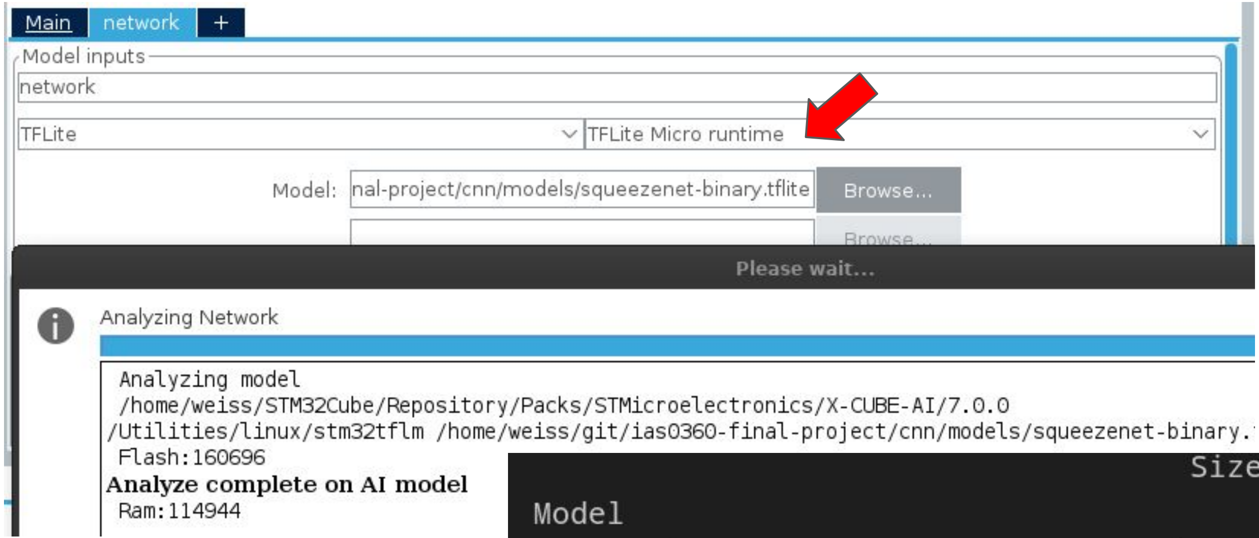
```
Total params: 119,890
Trainable params: 118,578
Non-trainable params: 1,312
```

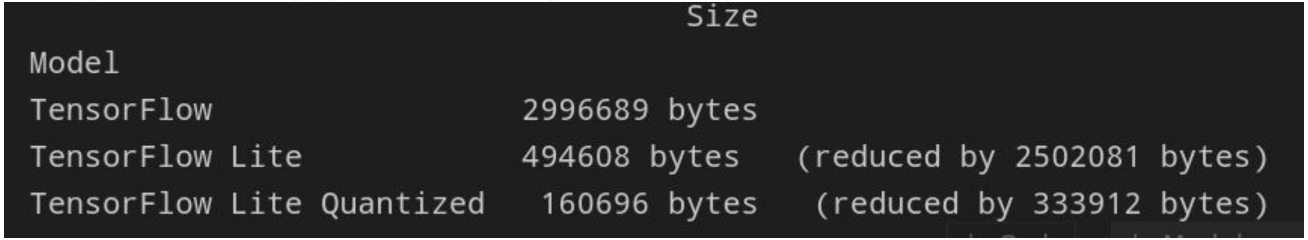
```
1.7M    squeezenet-binary.h5
```

Too big for MCU

Squeezenet Implementation (2)

- TF lite (micro) with **quantization**





Model	Size	
TensorFlow	2996689 bytes	
TensorFlow Lite	494608 bytes	(reduced by 2502081 bytes)
TensorFlow Lite Quantized	160696 bytes	(reduced by 333912 bytes)

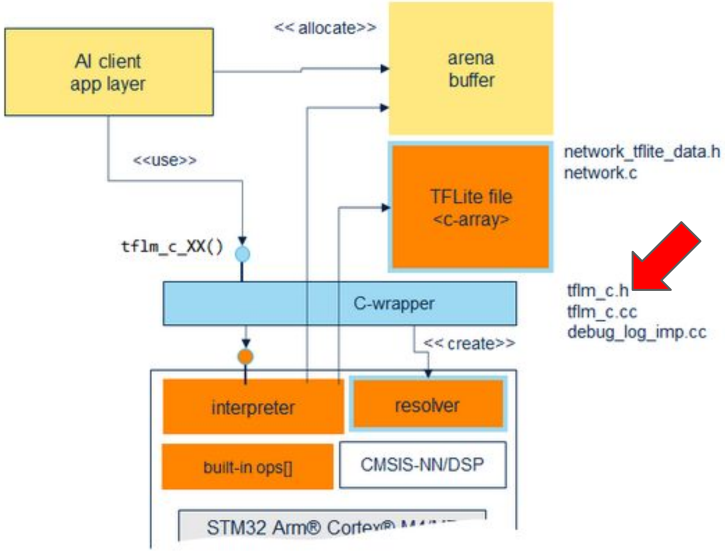
Squeezenet Implementation (3)

- Local CubeMX AI 7.0.0 Documentation:

  ~/STMicroelectronics/X-CUBE-AI/7.0.0/Documentation/tflite_micro_support.html#x-cube-ai-solution

-

```
Include paths (-I)
../Inc
"${workspace_loc:/${ProjName}/Middlewares/tensorflow/third_party/ruy}"
"${workspace_loc:/${ProjName}/Middlewares/tensorflow}"
"${workspace_loc:/${ProjName}/Middlewares/tensorflow/third_party/cmsis/CMSIS/DSP/Include}"
"${workspace_loc:/${ProjName}/Middlewares/tensorflow/third_party/cmsis}"
"${workspace_loc:/${ProjName}/Middlewares/tensorflow/third_party}"
"${workspace_loc:/${ProjName}/Middlewares/tensorflow/third_party/gemmlowp}"
"${workspace_loc:/${ProjName}/Middlewares/tensorflow/third_party/cmsis/CMSIS/NN/Include}"
"${workspace_loc:/${ProjName}/Middlewares/tensorflow/third_party/flatbuffers/include}"
"${workspace_loc:/${ProjName}/Drivers/CMSIS/Include}"
"${workspace_loc:/${ProjName}/Drivers/BSP/STM32F429I-Discovery}"
"${workspace_loc:/${ProjName}/Drivers/BSP/Components/Common}"
../Drivers/STM32F4xx_HAL_Driver/Inc
../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy
${workspace_loc:/${ProjName}/Drivers/CMSIS/Device/ST/STM32F4xx/Include}
```



Squeezenet Implementation (4)

- CMSYS Fast MCU NN Kernels:

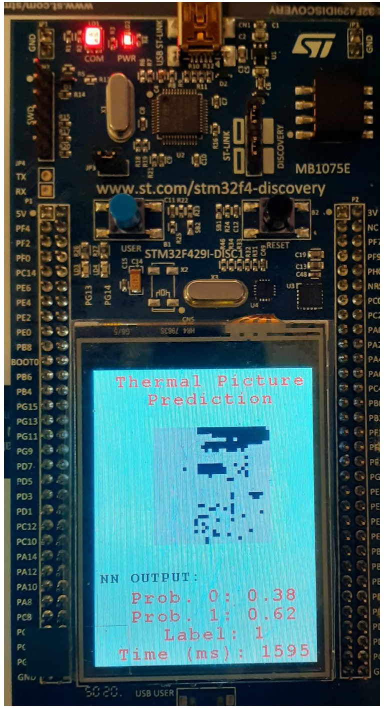
"New CMSIS-NN Neural Network Kernels
Boost Efficiency in Microcontrollers by ~5x"

https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/new-neural-network-kernels-boost-efficiency-in-microcontrollers-by-5x?utm_source=keil_com

Squeezenet Implementation (5)

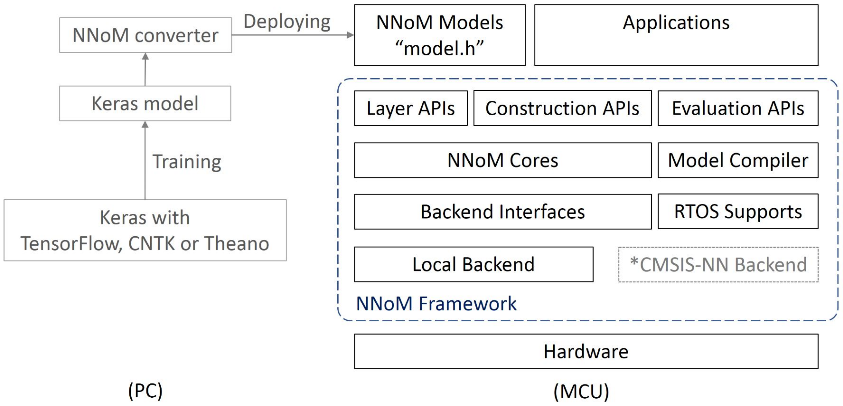
- STM TF lite C API
  - Experimental
  - No documentation
- Code glimpse / Demo

```c
// Run inference
nn_status = tflm_c_invoke(hdl);
if(nn_status)
    _Exit(nn_status);
```



Squeezenet Conclusion

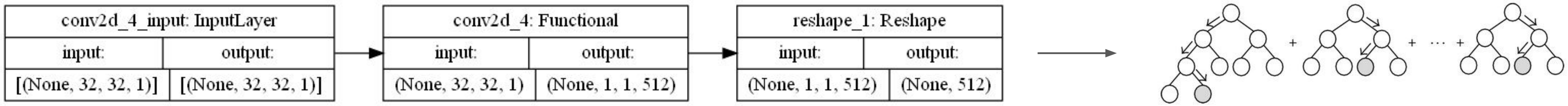
- Maybe consider other toolchain:
  - C++ (with Mbed?)
  - NNoM https://majianjia.github.io/nnom/



Gradient Boosted Trees (GBT) (1)

General Approach

- Implementation of 2 Models:
    a. Feature Generator - Convolutional Layers from CNN
    b. Classifier - GBT (& SVC) Classifier
- Classification based on 512 Features from CNN



Gradient Boosted Trees (GBT) (2)

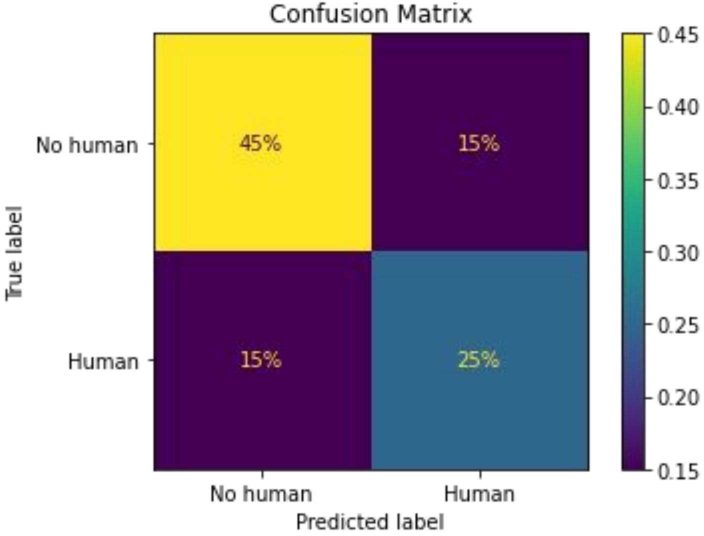
Implementation and Procedure

- Hyperparameter Tuning of CNN and Extraction of Convolutional Layers
- Hyperparameter Tuning of GBT
- Additional Approach: Support Vector Classifier (SVC) as Classifier on Same Features

Classification Results

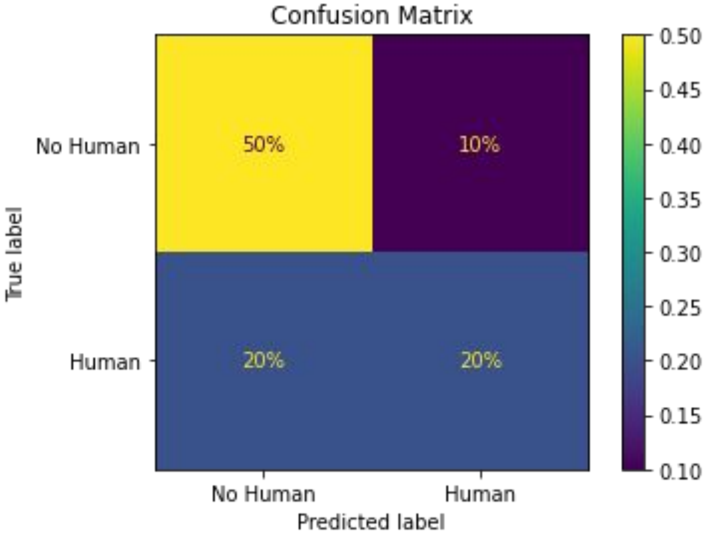
GBT Classifier

Training Accuracy: 98.8%
Test Accuracy: 70.0%



Support Vector Classifier

Training Accuracy: 94.4%
Test Accuracy: 70.0%



GBT Implementation (1)

Issues with CubeMX AI 7.0.0

- GBT Classifiers not supported ⟶ SVC
- Joined ONNX-Models not supported
- Multi-Output Models not supported
    - Classifier as ONNX-Model has two outputs: Labels & Probabilities
    - ONNX-Model Conversion ⟶ OUT OF SCOPE!

GBT Implementation (2)

Sizes of Networks on Board

Feature Generator

Support Vector Classifier (SVC)

Model manager

Name	RAM	Flash	Complexity
network	10.28 KiB	151.88 KiB	78780 MACC
network_3	2.01 KiB	118.24 KiB	302080 MACC
Total (2)	**12.29 KiB (192.00**	**270.12 KiB (2.00**	**380860 MACC**

Approach would be RAM- and Flash-efficient!

Conclusion (1)

Data

- Labelling is highly subjective and ambiguous
- (Test) Dataset is very small compared to statistical practice (Here: 20 test samples vs. Good practice: 50 test samples and more)
- Class imbalance in the dataset

Conclusion (2)

Models and Implementation with CubeMX AI 7.0.0

- Some state-of-the-art models are not supported
- CNN has big size compared to other classifiers with comparable accuracy
- Implementation of non-standard models is workintense and out of scope for this project