Machine Learning at the Network Edge for Automated Home Intrusion Monitoring

Jose Sepulveda

Authors

Aditya Dhakal,
K. K. Ramakrishnan,
Both from UC Riverside

Outline

Introduction (Background)

Security and monitoring of residences and businesses is a significant industry with lots of room for improvement.

These security systems can range from simple ones, to very complex ones with people monitoring alarm triggers 24/7.

These systems typically have loads of sensors, cameras, biometric scanners, motion sensors, and window/door sensors.

A false positive can be expensive! How do we reduce this?

Background (Setup)

A automated home/business monitoring system at the edge is proposed.

It preforms online learning on the incoming stream of data.

Monitoring framework is hosted on OpenNetVM, a Network Function Virtualization platform.

Background (NFV)

A subset of SDN

Allows for the specialized hardware tasks (firewalls, DPI, Load Balancers) to run in a virtualized enviroment on a server.

Why NFV?

For this application we need scalability, if the home has camera sensors we can spin up a new NF for face detection.

Motivation

Monitoring systems are only becoming more complex, with new sensors and detection algorithms.

Having a human keep watch over the stream of data would be difficult.

We need a scalable, cheap, and easily updated solution.

Motivation (Automated Monitoring)

3 different methods of automated monitoring:

- Include computation, communication, and sensors within each home.
- Sharing resources across all homes.
- Distributed edge computing on the neighborhood scale.

System Design

They focus on home monitoring by using a small apartment as an example.

Homes and businesses may have significantly more complicated placements and sensors.

System Design

Sensor placement

M# - motion sensor

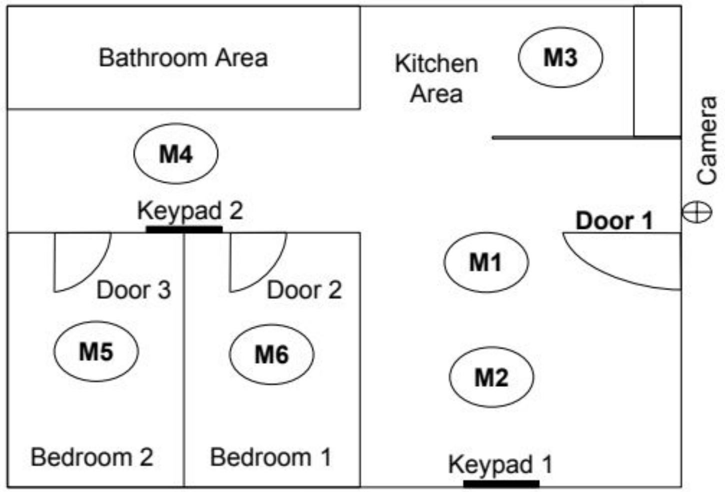


Fig. 1: Apartment Layout

System Design

They define a “home entry” as a sequence of triggers of the sensors which starts as soon as the door is opened.

Even if there is a camera present outside, the stream of data is not processed until the front door is opened.

System Design

Each sensor will send a time-stamped packet to the edge server at time of triggering.

An “home entry” path is not complete until the PIN is entered into the keypad.

System Design

Normal “home entry” paths.

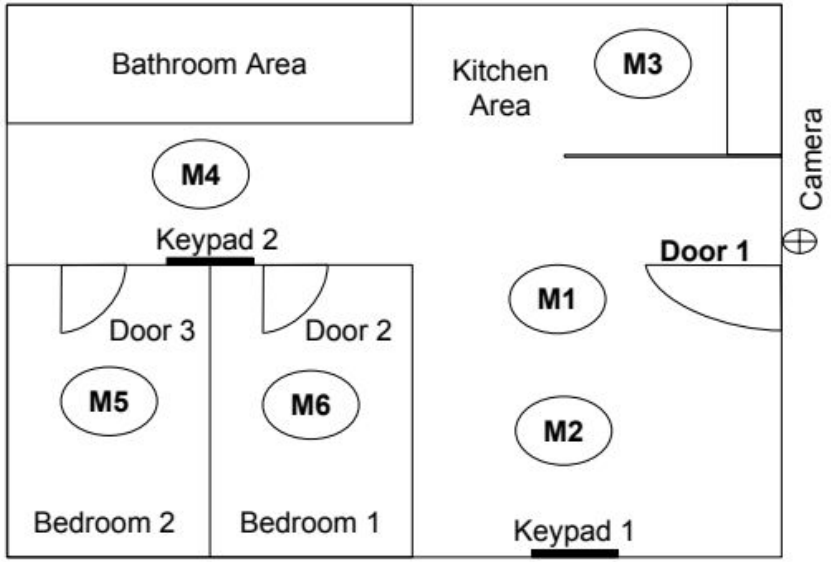


Fig. 1: Apartment Layout

System Design

They modeled there data for home entry after the real life home entry data collected in the kyoto dataset.

The Kyoto dataset is the data collected from the testbed used in:

“Activity Learning as a Foundation for Security Monitoring in Smart Homes”

NFV & Testbed Implementation

The NFV platform they used is OpenNetVM.

OpenNetVM is built on DPDK and docker containers.

DPDK - libraries/drivers built for fast packet processing.

Docker - Operating system level virtualization, avoids cost associated with creating and maintaining VM by using linux resource isolation features to create application specific containers.

NFV & Testbed Implementation

OpenNetVM's use of docker allows for new NFs to be added easily and cost effectively.

This is important in the security market sector.

It provides:

- Variability in home Security options
- Able to add new NF's to service new sensors
- Allows algorithm's to run contained for performance and security.

NFV & Testbed Implementation

The testbed is a single system which intercepts packets coming from the various sensors.

For the tests, instead of using an actual apartment they used another system running OpenNetVM that generates traffic, it emulates the sensors, keypad, and even contains images.

Algorithms

They picked 4 features that they thought would be the most useful.

1. Time between activation of each sensor
2. Time taken to complete “home entry” event
3. The image from the camera sensor
4. The number of times the PIN is entered on the keypad

Algorithms

For each of the four features, their is an associated classifier.

Each classifier also provides a belief value between [0, 1]. This number is a measure of how confident the classifier is, 1 being fully confident.

This belief number is used along with the Dempster-Shafer combination rule to reach an overall consensus.

Algorithms (K-NN)

Feature: Time between activation of each sensor

K nearest neighbor or K-NN is the classifier chosen for this feature.

A K-NN classifier is the right choice because it lacks a training phase, it’s online, and is based on feature similarity to classify into discrete groups (normal entry vs intrusion).

Belief value:

$$Bel(x) = \begin{cases} 1, & \text{if } n < K \\ 0, & \text{if } n > \text{KNN}_{avg} \\ 1 - \frac{n-K}{\text{KNN}_{avg}-K} & \text{otherwise} \end{cases}$$

Algorithms (Disarm-Time)

Feature: Time taken to complete “home entry” event

$$Bel(x) = \Phi\ (z), \text{where,}\ z = \frac{x - \mu}{\sigma}$$

x - the event
z - z-score of an inferred value
μ - the mean time of entry
σ - standard deviation
Φ(z) - probability from probability density function of the z-scores

Algorithms (Face Detection)

Feature: The image from the camera sensor

OpenFace, a free and open source face recognition solution using deep neural networks.

They used a pretrained model with 6000 images of 10 different celebrities.

It returns value between [0,1] on how likely a new image is of a member of it’s dataset.

Belief value: $Bel(image) = 1 - \text{OpenFace}(image)$

Algorithms (Keypad)

Feature: The number of times the PIN is entered on the keypad.

Classifies an intrusion if the PIN is entered more than 3 times incorrectly.

Belief value:

$$Bel(x) = \begin{cases} 1, & \text{if } x > 5 \\ 0, & \text{if } x < 3 \\ \frac{x-2}{4} & \text{otherwise} \end{cases}$$

Algorithms (Dempster-Shafer)

Dempster-Shafer theory is a widely used method to combine decisions from multiple ML classifiers and make an overall decision.

It requires no prior knowledge, meaning that is perfect for intrusion detection because it is independent of past scenarios.

D-S is used to combine all the decisions and reduce the false positive/negative rates.

Evaluation

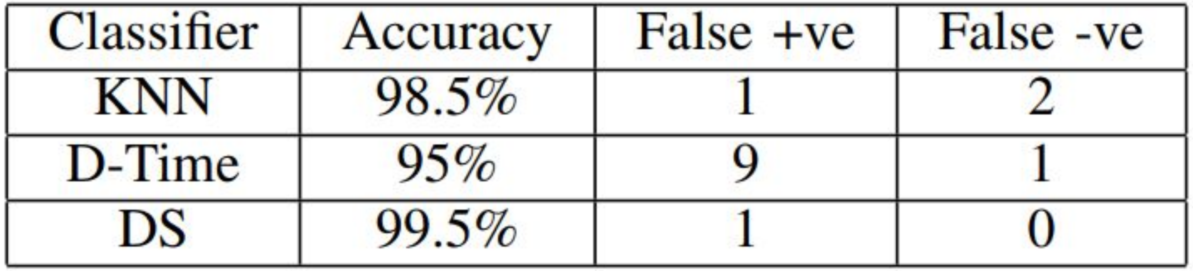
The synthetic data was generated in the timescale of seconds. In order to evaluate it quickly they scaled it down to milliseconds.

This caused some complications for Openface, so they just inferred images separately and packets only contain results, no images.

Evaluation

They ran 2 testing scenarios.

For scenario 1, intruder enters the house looking for the keypad or things to steal. There was also no camera or keypad data.



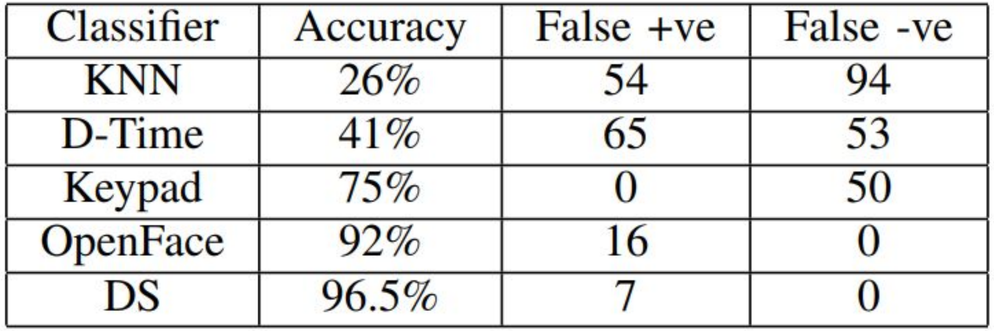
Classifier	Accuracy	False +ve	False -ve
KNN	98.5%	1	2
D-Time	95%	9	1
DS	99.5%	1	0

TABLE I: Classifiers' performance in Scenario 1

Evaluation

They ran 2 testing scenarios.

For scenario 2, intruder enters the house and goes straight for keypad, and makes multiple attempts to disable it.

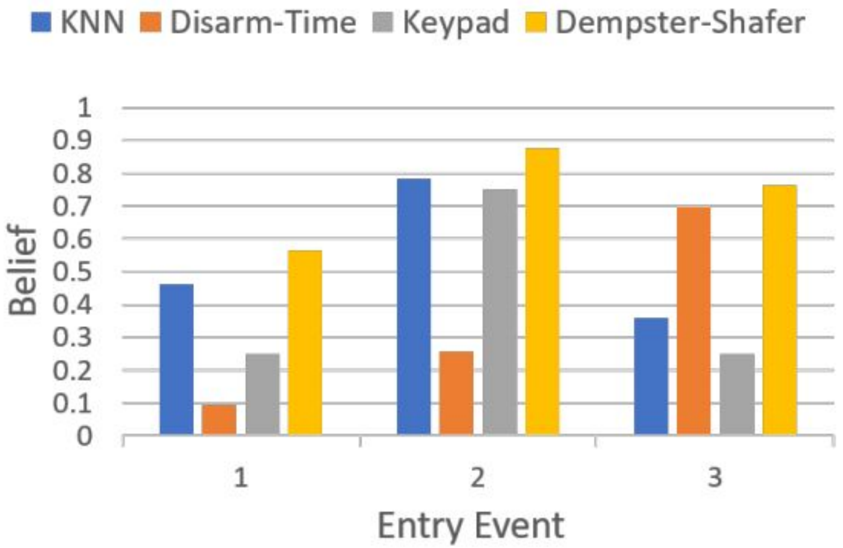


Classifier	Accuracy	False +ve	False -ve
KNN	26%	54	94
D-Time	41%	65	53
Keypad	75%	0	50
OpenFace	92%	16	0
DS	96.5%	7	0

TABLE II: Classifiers' performance in Scenario 2

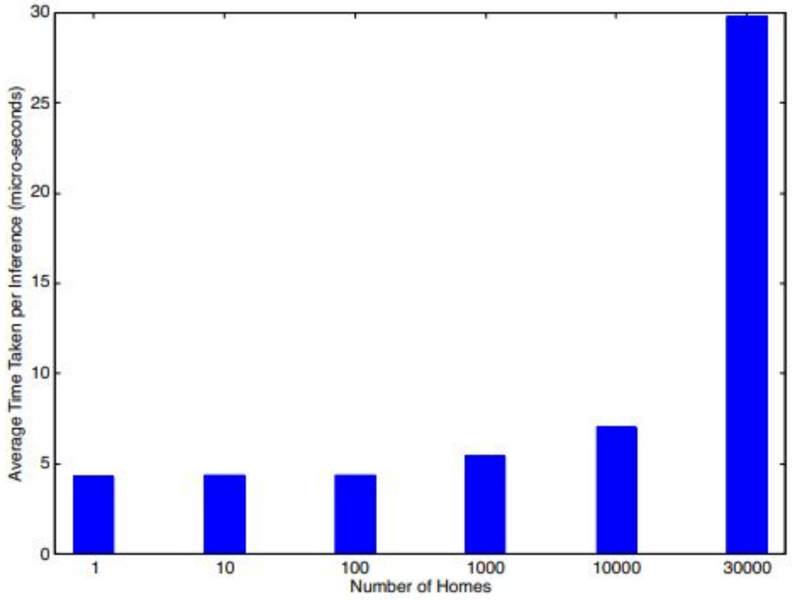
Evaluation (Belief)

Threshold for D-S to classify an entry as an intrusion is 0.75



Evaluation (Scalability)

They run a test where the system is servicing multiple homes.



Conclusion

D-S + varied classifiers is a good solution to reduce false positives/negatives.

Using NFV is a good way to make a scalable service for IoT related computation.

Criticisms:

- This paper was not written very well, many things were left out and English was ambiguous at times.
- They didn’t describe some of the classifiers well.
- They didn’t use the real camera feed.