CAM-AI: An Open-Source Artificial Intelligence Solution for False Alarm Reduction in Security Cameras

Abstract: False alarms in security camera systems can lead to unnecessary disruptions and reduce the effectiveness of surveillance. The CAM-AI camera software addresses this issue by integrating advanced artificial intelligence (AI) capabilities with security cameras. This paper introduces CAM-AI, highlighting its innovative approach to differentiating between harmless movements and genuine threats using a unique combination of motion detection and AI image classification. The project offers an open-source software solution, written in Python and accessible on GitHub, making it versatile and compatible with a range of devices from a Raspberry Pi to a powerful server PC running Debian Linux. CAM-AI's core capabilities include pre-trained recognition of humans, dogs, cats, cars, and other objects, ensuring accurate detection right from the installation. Further improvement of the reliability of the image classification can be achieved by specific training on the data from a specific camera.

1. Common problems with security cameras and what CAM-AI can do to fix them

1.1 Security cameras are prone to false alarms, especially in outdoor setups. Many cameras use motion detection to generate alarms. This works well with indoor setups (as long as you don’t have pets in your home) but is not very good outdoors. Pure motion detection registers irrelevant movements, such as leaves blowing in the wind, quickly moving clouds or animals passing by.
CAM-AI uses a combination of motion detection and AI **image classification**[1], thus eliminating false alarms.

1.2 Some camera suppliers have recently added AI features to their products, mostly by opening the users’ cameras to their remote servers which perform all needed AI operations. The precision of these servers’ image classification is limited because they all work with standardized **models**[2]. Improving the **models** by training with customer data is not possible.
CAM-AI does exactly this: The user may train his individual **model** on individual data from his camera, thereby significantly improving the quality of the **predictions**[3].

1.3 Many customers require their image and video data to remain on their own systems for data protection reasons. Sharing the data with some unknown server abroad for processing is not an option. In some countries, it is even illegal to move customer data to another server without the explicit permission of the user.
Since CAM-AI is a Python-made web application, it offers a broad range of options for the installation of the server, according to the users’ needs:

1.3.1 The easiest way is to create an account on **CAM-AI’s cloud server** (https://django.cam-ai.eu/accounts/register/). The server gets the video data from the users’ cameras and does all the processing. This includes for example alarm procedures such as sending emails. Data is safe on that server, it is located in the EU, but most of all it comes with state-of-the-art protection. All the user needs to run this solution is a browser and an internet connection.

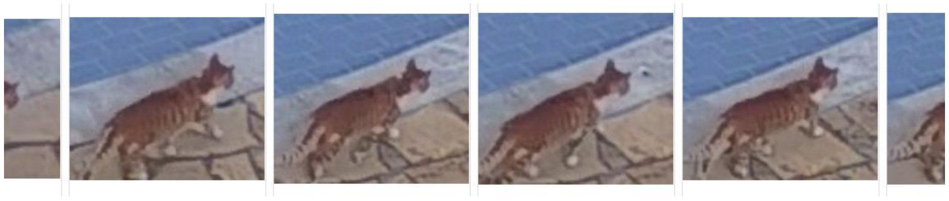
1.3.2 Users, who want or need their storage server within their own infrastructure, may e. g. install **CAM-AI on a Raspberry Pi**. This very small computer takes care of the storage as well as the logistics, the AI processing is done by the CAM-AI's cloud server. Transfers to and from the server are performed using end-to-end encryption. The Raspberry Pi is operated by a browser via a web interface as well.

1.3.3 Completely autonomous CAM-AI systems are set up on PCs running **Debian Linux OS**. No need to transfer images or videos beyond your own domain. Predictions are generated locally, local training requires the installation of a third-party plugin.

2. Short introduction to AI image classification using CAM-AI

2.1 How to get images with motion detection:

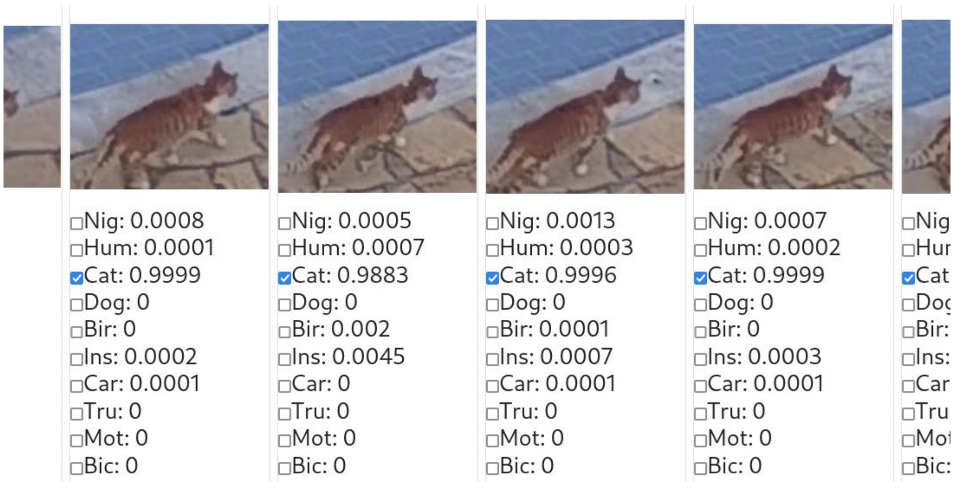
Two times per second the motion detection pulls a sample image from the video stream and compares it with the last image. The system cuts out the areas that are different (i. e. the areas where some movement took place). This process is repeated every half second. The result is a sequence of images of every object that moves within the camera's view. Here is an example of the resulting small images:



At this point, the content of the images is unknown to the system and therefore it also is unknown whether the images are relevant for surveillance. This is where AI **image classification** is needed:

2.2 How to analyze image content with AI:

Each of these images now is processed by an AI software system called **Convolutional Neural Network**[4] which supplies a set of 10 tags per image:

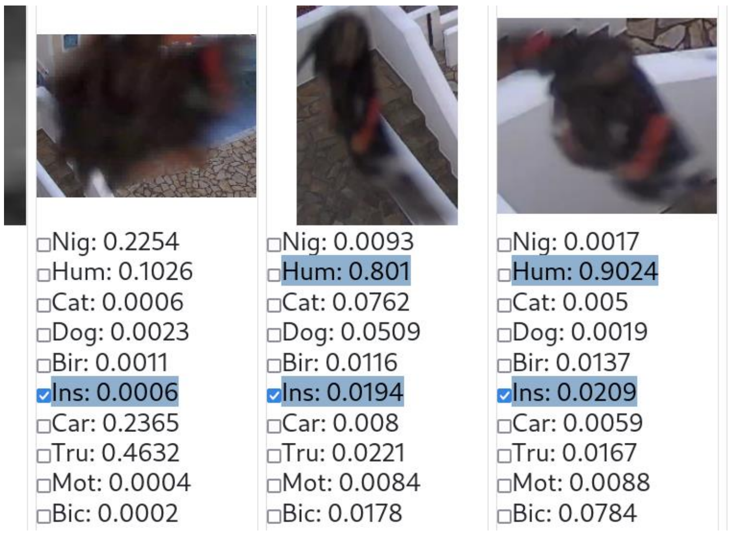


☐Nig: 0.0008	☐Nig: 0.0005	☐Nig: 0.0013	☐Nig: 0.0007	☐Nig
☐Hum: 0.0001	☐Hum: 0.0007	☐Hum: 0.0003	☐Hum: 0.0002	☐Hur
☑Cat: 0.9999	☑Cat: 0.9883	☑Cat: 0.9996	☑Cat: 0.9999	☑Cat
☐Dog: 0	☐Dog: 0	☐Dog: 0	☐Dog: 0	☐Doç
☐Bir: 0	☐Bir: 0.002	☐Bir: 0.0001	☐Bir: 0	☐Bir:
☐Ins: 0.0002	☐Ins: 0.0045	☐Ins: 0.0007	☐Ins: 0.0003	☐Ins:
☐Car: 0.0001	☐Car: 0	☐Car: 0.0001	☐Car: 0.0001	☐Car
☐Tru: 0	☐Tru: 0	☐Tru: 0	☐Tru: 0	☐Tru
☐Mot: 0	☐Mot: 0	☐Mot: 0	☐Mot: 0	☐Mot
☐Bic: 0	☐Bic: 0	☐Bic: 0	☐Bic: 0	☐Bic:

As you might have guessed: The numbers between 0.0 and 1.0 represent the likeliness of ten different categories of objects being visible in each image. In this case, the result is very simple: There is a cat in each of the four images, but nothing else of relevance. No human, dog, bird, insect, car, truck, motorcycle, or bicycle, which are the other categories available in the standard setup. And because we are not interested in alarm emails showing cats sneaking across our yard, we configured the system to not trigger an alarm when spotting cats. This test was done using the standard model that comes with every new server installation and every new account on the cloud server. This model is good at detecting humans, cats, dogs, birds, or cars. However, there are many cases in real-life **image classification** that are not that simple. If you need more precision and reliability you need to go further:

2.3 Going beyond the simple case with individual training:

Let’s look at this example: Our standard **model** is sufficiently trained to recognize humans, cats, and dogs, but now a butterfly is flying by. The **model** has never seen this kind of butterfly before. But it has seen blurred images of men in dark working overalls with reflecting stripes on their pants and maybe with orange caps on their heads.

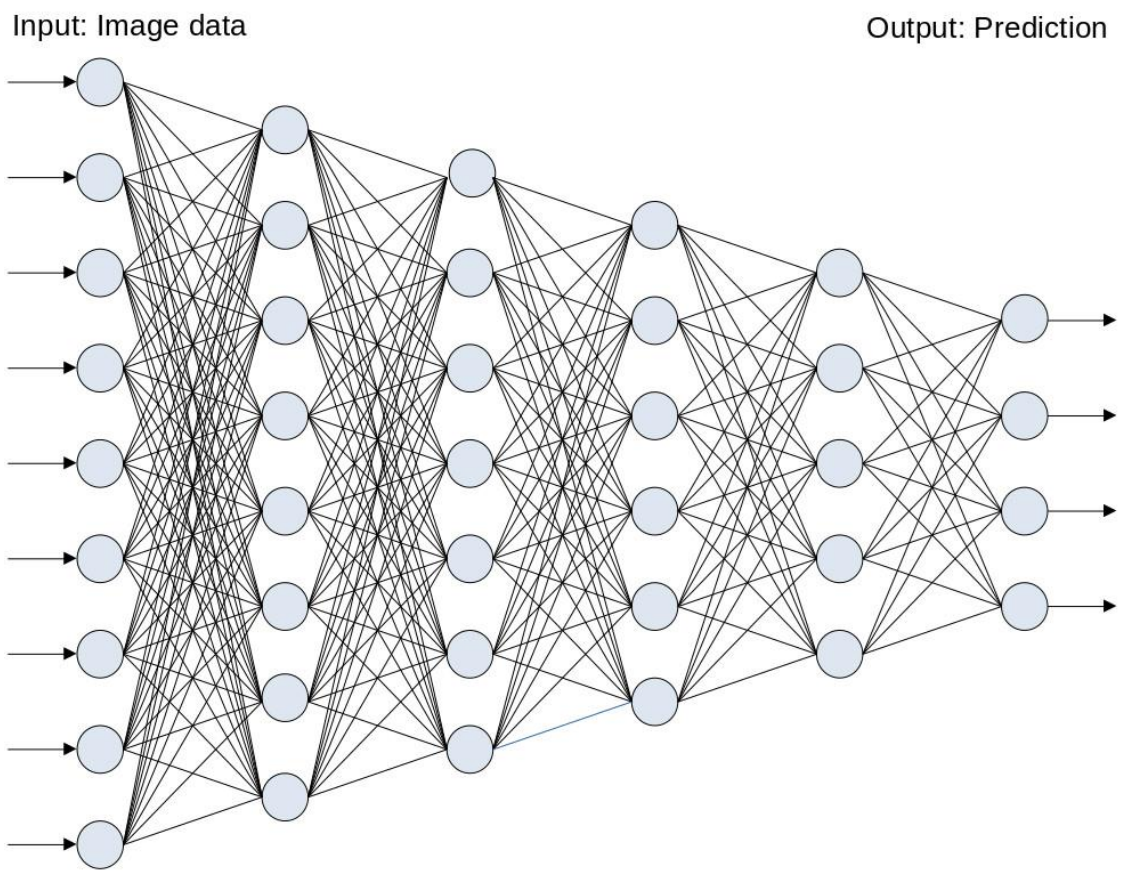


☐Nig: 0.2254	☐Nig: 0.0093	☐Nig: 0.0017
☐Hum: 0.1026	☐Hum: 0.801	☐Hum: 0.9024
☐Cat: 0.0006	☐Cat: 0.0762	☐Cat: 0.005
☐Dog: 0.0023	☐Dog: 0.0509	☐Dog: 0.0019
☐Bir: 0.0011	☐Bir: 0.0116	☐Bir: 0.0137
☑Ins: 0.0006	☑Ins: 0.0194	☑Ins: 0.0209
☐Car: 0.2365	☐Car: 0.008	☐Car: 0.0059
☐Tru: 0.4632	☐Tru: 0.0221	☐Tru: 0.0167
☐Mot: 0.0004	☐Mot: 0.0084	☐Mot: 0.0088
☐Bic: 0.0002	☐Bic: 0.0178	☐Bic: 0.0784

So, the classification of the three pictures is no surprise: The **Convolutional Neural Network** detects a human near the steps, which is, of course, wrong. This sounds like a theoretical and rare problem, but it happens a lot with AI systems that are too simple. If you want to prevent these false alarms, you need to teach your system what this image contains by checkmarking the “Ins(ect)”-line for all three images. After the next training, the **model** will have learned that this butterfly is an insect. Generally speaking, individually trained **models** know the items and issues that are **not** a reason for alarms and thus significantly reduce the number of false alarms.

2.4 Retraining the model:

Once we collect a couple of images with things to learn (like the butterfly images above), we can launch the training process with one mouse click. Before we take a closer look at this process, let’s have a look at the following simplified sketch of a neural network:



The image data is fed into a large grid of variables that is organized in layers (in this example: 6 layers, counted from left to right). Each layer consists of many cells (the light blue-grey points) that contain floating point numbers. Many of them are connected by a sophisticated pattern of relations, here represented by black lines. Each of these lines has a numerical value, the so-called weight. The weight explains the permeability of that line when a signal (image) is processed from left to right: A high weight results in the right cell getting a large portion of the value of the left cell, and a low weight reduces the permeability.

This graphic illustration contains a couple of simplifications, the most important two are:

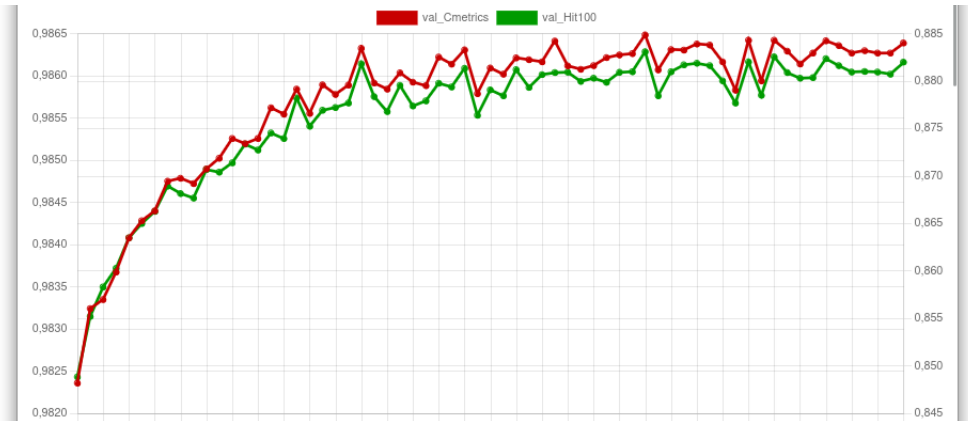
A) The dimensions of real, useful neural networks are much larger than that of the one in this graphical representation. On the input side, we might (for example) have 150.528 numbers (and not 9 like in our example) representing an image with 224 x 224 pixels and 3 color channels per pixel (224 x 224 x 3 = 150.528). On the output side, the number of channels is 10 in real-life CAM-AI (not 4), one for each category to detect (humans, cats, cars, etc.).

B) Our graphics don't show a Convolutional Neural Network, but a simpler and smaller structure. Really capable neural networks have millions of weights.

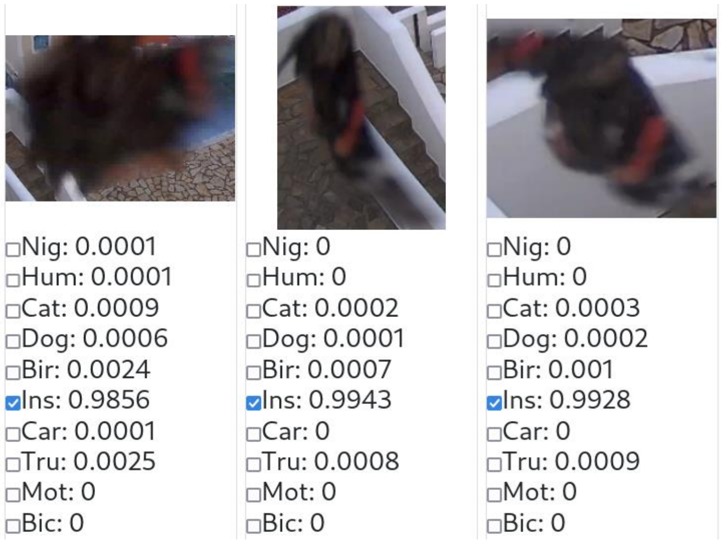
Now the training process is started by feeding the image into the first (leftmost) layer. Once this is done, the complete network (all the cells) automatically gets populated according to the weights that come with the black lines, from left to right. In the end, the last (rightmost) layer contains the prediction, meaning: The categories that the AI-system derived from the image data. Now the CAM-AI compares this resulting **prediction** with the tags the user (=teacher) attributed to this image and any error is fed back into the **model's** weights by a complex process called **backpropagation**[5]. This procedure is repeated many times with many images and their given tags and will improve the quality of the **predictions**.

2.5 The learning curve:

With the following graphic illustration, you can check the progress of a well-parameterized learning process:



The red line indicates the percentage of tags correctly recognized. The green line indicates the percentage of completely correct predictions. Let’s see what this means to our three sample images from section 2.3:



☐Nig: 0.0001	☐Nig: 0	☐Nig: 0
☐Hum: 0.0001	☐Hum: 0	☐Hum: 0
☐Cat: 0.0009	☐Cat: 0.0002	☐Cat: 0.0003
☐Dog: 0.0006	☐Dog: 0.0001	☐Dog: 0.0002
☐Bir: 0.0024	☐Bir: 0.0007	☐Bir: 0.001
☑Ins: 0.9856	☑Ins: 0.9943	☑Ins: 0.9928
☐Car: 0.0001	☐Car: 0	☐Car: 0
☐Tru: 0.0025	☐Tru: 0.0008	☐Tru: 0.0009
☐Mot: 0	☐Mot: 0	☐Mot: 0
☐Bic: 0	☐Bic: 0	☐Bic: 0

The **model** has obviously learned. The butterfly is no longer recognized as a human. The system is now sure about what that object is (values for “Ins” are all close to 1.0). We have reached the most important goal: **No more false alarms from butterflies!**

Of course, this example represents a solution for many kinds of false alarms.

Conclusion: CAM-AI offers an innovative, open-source solution to the persistent problem of false alarms in security camera systems. With smart motion detection combined with advanced AI capabilities (including individual camera training), the reliability of these systems is highly improved.

Visit https://cam-ai.de to learn more about preventing false alarms and improving home or business security.

Footnotes:

[1] Image classification is a computational task within the domain of computer vision, aimed at categorizing visual data into predefined classes. The fundamental principle involves the extraction of discriminative features from digital images and their subsequent mapping to distinct categories through machine learning algorithms. The process typically comprises a training phase, during which the algorithm learns patterns and features from a labeled dataset, and a testing phase, where the trained model is applied to unseen images for accurate classification.

[2] In the context of image classification, the term "model" refers to a computational representation or framework that has been trained to recognize and categorize objects or patterns within digital images. This model is essentially a mathematical abstraction that generalizes the relationships and features present in the training data to make predictions on new, unseen data. The process of building an image classification model involves training it on a labeled dataset. This dataset consists of images along with corresponding class labels, indicating the object or category each image belongs to (e.g., "cat," "dog," "car"). During training, the model learns to identify patterns and features that are characteristic of each class. The model's goal is to generalize from the training data, allowing it to correctly classify new, previously unseen images.

[3] In image classification, the term "prediction" refers to the output or inference made by a trained model when it is presented with a new, unseen image. The model analyzes the features and patterns within the input image and assigns it to one or more predefined classes or categories based on what it has learned during the training phase.

[4] A Convolutional Neural Network (CNN) is a specialized type of artificial neural network designed for processing and analyzing visual data, such as images and videos. CNNs have proven highly effective in tasks like image classification, object detection, and image recognition.

[5] Backpropagation, short for "backward propagation of errors," is a supervised learning algorithm used to train artificial neural networks, including Convolutional Neural Networks (CNNs). The primary objective of backpropagation is to minimize the error between the predicted output of the neural network and the actual target output by adjusting the weights of the network through gradient descent.