Vision2Sensor: Knowledge Transfer Across Sensing Modalities for Human Activity Recognition

VALENTIN RADU, University of Edinburgh, UK
MAXIMILIAN HENNE, University of Edinburgh, UK

Mobile and wearable sensing devices are pervasive, coming packed with a growing number of sensors. These are supposed to provide direct observations about user activity and context to intelligent systems, and are envisioned to be at the core of smart buildings, towards habitat automation to suit user needs. However, much of this enormous sensing capability is currently wasted, instead of being tapped into, because developing context recognition systems requires substantial amount of labeled sensor data to train models on. Sensor data is hard to interpret and annotate after collection, making it difficult and costly to generate large training sets, which is now stalling the adoption of mobile sensing at scale. We address this fundamental problem in the ubicomp community (not having enough training data) by proposing a knowledge transfer framework, Vision2Sensor, which opportunistically transfers information from an easy to interpret and more advanced sensing modality, vision, to other sensors on mobile devices. Activities recognised by computer vision in the camera field of view are synchronized with inertial sensor data to produce labels, which are then used to dynamically update a mobile sensor based recognition model. We show that transfer learning is also beneficial to identifying the best Convolutional Neural Network for vision based human activity recognition for our task. The performance of a proposed network is first evaluated on a larger dataset, followed by transferring the pre-trained model to be fine-tuned on our five class activity recognition task. Our sensor based Deep Neural Network is robust to withstand substantial degradation of label quality, dropping just 3% in accuracy on induced degradation of 15% to vision generated labels. This indicates that knowledge transfer between sensing modalities is achievable even with significant noise introduced by the labeling modality. Our system operates in real-time on embedded computing devices, ensuring user data privacy by performing all the computations in the local network.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing systems and tools**; • **Computing methodologies** → Knowledge representation and reasoning.

Additional Key Words and Phrases: knowledge transfer, transfer learning, computer vision, sensing, smartphone, smart camera

ACM Reference Format:
Valentin Radu and Maximilian Henne. 2019. Vision2Sensor: Knowledge Transfer Across Sensing Modalities for Human Activity Recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 3, Article 84 (September 2019), 21 pages. https://doi.org/10.1145/3351242

1 INTRODUCTION

With a variety of sensors being introduced to mobile and wearable devices (e.g., accelerometers, gyroscopes, barometers, thermometers, etc.) many intelligent applications can be imagined to use this growing sensing capability for complex context recognition, such as ambient and location characterization [3, 35, 47], Human Activity Recognition (HAR) [12, 34], medical supervision [9] and others. HAR solutions are particularly interesting

Authors' addresses: Valentin Radu, University of Edinburgh, UK, valentin.radu@ed.ac.uk; Maximilian Henne, University of Edinburgh, UK, maximilian.henne@sms.ed.ac.uk.

because these can be specialised for each application domain, such as for fitness tracking and for gym exercise recognition, using specialised models for each task. Smartphones are the most pervasive mobile sensing device, coming with a wide variety of sensing modalities enabled by the many embedded sensors. Despite their prevalence, smartphone sensing capability is underutilised by current mobile applications for sensor based observations.

A major barrier to wider adoption and exploitation of mobile sensing is the lack of available training data. Annotation of sensor data is hard and expensive [25] and does not generalize at scale. Having enough training data from one individual is not sufficient since natural differences exist in the way subjects perform movements and activities, and at a closer inspection also differences between sensor properties across devices [44]. To adapt models to each user and device would require a large amount of training data to be manually labeled, which currently is infeasible, so automated solutions to generate these labeled datasets should be explored.

The situation is different in the computer vision research community, which has been going through a radical transformation in the last few years [22, 37, 38], due to the growing number of available labeled datasets and benchmarks. It is also easier to annotate images and videos by direct assessment with the human eye, and this can be interpreted at any time after data collection. Unlike images, sensor signals are harder to interpret by humans in isolation from their collection context.

Although computer vision and sensor data have been used together before, for video synchronization to allow manual labeling of sensor data based on simple hand gestures observations [33], it has never been used for automatic sensor data annotation with robust modern computer vision classifiers. As previously observed by D. Cook et al. [8], very little effort has been invested in transferring knowledge between radically different sensing modalities. Previous work shows that labels produced by a sensor model can be used to train another model operating also on sensor data, but of a different characteristic [23]. With vision being one of the most robust sensing modalities, it makes sense to build on computer vision techniques to advance the ubicomp sensing space.

We aim to eliminate the current barrier of not having enough training data for diverse sensing modalities by introducing Vision2Sensor. *Vision2Sensor is an opportunistic sensor data labeling system performing knowledge transfer between computer vision and mobile sensor data when user is in front of a camera, by exploiting human actions and activities being observed from multiple sensing perspectives simultaneously.* Computer vision is already mature and robust to identify complex conditions in images and videos, due to the abundance of training data in the computer vision community. Here we concentrate on HAR as a use-case for Vision2Sensor, but this can also be applied to other tasks (location estimation, social interactions detection and medical rehabilitation monitoring). To offer a comprehensive example, we demonstrate the capability of Vision2Sensor for home automation driven by human activity and actions. The set of activities considered in our evaluation is designed for controlling connected devices through HAR on wearable sensing devices, such as by performing a full-turn to adjust sound volume, walking, sitting and standing to switch on and off devices in the user environment (TV, lights, heating).

Vision2Sensor is composed of two subsystems: one on the mobile device managing the sensor data stream, processing sensor data locally and synchronising with the camera when in sight; and a second subsystem running on a compute capable camera, which performs vision based HAR using a Convolutional Neural Network (CNN), to produce labels, which are assigned to sensor data for training the sensor based model. For precise labeling, video stream and sensor data are time synchronized. We show that the best performing CNN for vision based HAR is a 3D convolutional architecture, trained to reach a high inference accuracy with knowledge transfer from other vision datasets. Our experiments show that this model runs in close to real-time on an embedded GPU (Jetson TX2). Another important finding is that degradation to vision labeling is not immediately transferred to the sensor based Deep Neural Network (DNN), showing this is robust to errors in the quality of vision generated labels below 20%. Although not exploited here, this tolerance to noisy labels indicates that even smaller models than ours can be used on the camera to produce these estimations with less computing resources and for processing the date from many people in camera sight in parallel.

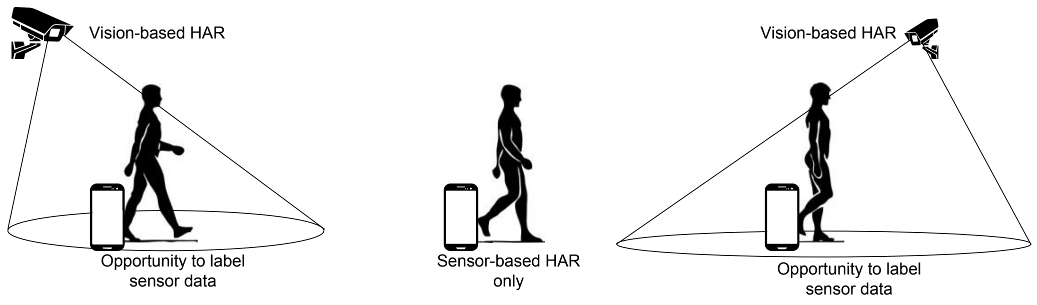

Vision-based HAR
Opportunity to label sensor data
Sensor-based HAR only
Vision-based HAR
Opportunity to label sensor data

Fig. 1. Vision2Sensor facilitates opportunistic training of smartphone human activity recognition classifiers through labels generated through visual recognition on cameras with compute resources. This enables the mobile system to perform detection on sensor data even when outside of camera sight.

This paper makes the following contributions:

- We design Vision2Sensor, an opportunistic sensor labeling system, using knowledge transfer to provide computer vision generated labels to mobile sensor data when in front of the camera.
- Knowledge transfer is used in two stages in the development of Vision2Sensor, first to determine the best computer vision classifier for our 5-class HAR, we use transfer learning from the larger dataset Sports-1M, and second to transfer knowledge from vision to sensor data as labels.
- Based on empirical observations, we find that some noise in the labeling modality is not immediately detrimental to the training quality of the receiving modality. The mobile sensor model is robust to withstand a significant amount of poisoned training instances.
- Our evaluation on real hardware shows that Vision2Sensor is efficient to run in real-time on edge computing devices (evaluated on the Jetson TX2 embedded GPU).

The remainder of this paper is organised as follows. The motivation for this work is presented in Section 2. Section 3 offers an overview of Vision2Sensor, its components and algorithms. The evaluation dataset and experiments are presented in Section 4. A brief discussion on results and potential applications of our system are presented in Section 5, followed by related work in Section 6 and conclusions in Section 7.

2 MOTIVATION

Many mobile and wearable devices today come equipped with a variety of sensors (accelerometer, gyroscope, barometer, thermometer, etc.), which can used for location tracking, activity recognition, understanding social interactions, health and exercise monitoring as commonly explored in the ubicomp community. However, labeling smartphone and wearable sensor data is hard. In data collection campaigns, it is common practice to ask participants to provide these labels themselves through mobile apps [21]. Other times participants provide succinct information on their activity in logging journals at the end of the day [24], often forgetting important details [29] or relevant micro-activities, which are hard to recreate for proper sensor data annotation. These approaches rely on remembrance and conscientiousness to provide regular and precise labels. Alternatively, users can be prompted to indicate their activity at regular intervals of time by mobile app notifications [1, 25]. The limitation with this approach is that notifications disrupt users from their ongoing activity, which causes user annoyance and more importantly changes to sensor signals, leading to sub-optimal datasets. The most reliable sensor data labels come from annotations provided by researchers under direct supervision of participants in controlled environments [44], although time consuming and expensive to perform at scale.

Another limiting characteristic of sensor data from mobile and wearable devices is that it is specific to each individual, making it hard to transfer across users due to each having a unique signatures in their movement [44], not considering differences in the physical properties of sensors across devices [10]. Labels are needed from each individual to adapt activity recognition systems to their data. Future smart environments will be operated not just with voice, as currently done by commercial personal assistants (Alexa and Google Home), but also through gestures and micro-actions. We can envision these smart environments to adapt automatically to suit activities performed in their spaces (*e.g.,* reducing room temperature on detected intensive activities, such as running on a treadmill and dancing; or increasing light intensity when standing up from the couch), but also to control devices through gestures (*e.g.,* a full body rotation to increase the volume of music, etc.). To enable such solutions considerable amount of training data is needed, which is currently hard to obtain as discussed before.

Computer vision has been growing substantially over the last few years due to the availability of training data and benchmarks for researchers to compare their solutions on [40]. The amount of data and labels in this space vastly exceeds that of available sensor datasets in the ubicomp community. It is easier to understand visual features (objects, activities, scenes, etc.) in images and videos at any time after data collection, due to our habitual reliance on vision naturally. Contrary to this, sensor signals are less easily annotated after data collection without additional hints or signal transformations for visualization. In addition, inertial sensors capture less information and are heavily reliant on body placement [48], knowledge which may not be available for offline annotation. Thus, producing large datasets in the ubicomp community to match those seen in the computer vision community is much harder, so alternative solutions are needed to generate this valuable training data.

To take advantage of all the knowledge captured in the already available labels of computer vision datasets, and of their robust recognition models, we propose to transfer this knowledge from vision to sensor signals by opportunistic synchronization on reference hints – location and time. Figure 1 presents this proposed approach, with sensor classifiers trained opportunistically when in the camera field of view using the labels generated by robust vision classifiers. *The benefit of using computer vision is that these models generalize across persons*, due to specializing in training less on unique characteristics of each individual, but more on differences between activity classes, generalizing well across people and scenes (benefit of the larger training sets).

We motivate the utility of automatic sensor data labeling with computer vision generated labels in a specific scenario, that of home automation. Among the growing number of network connected devices in the home (smart TV, speakers, lights, thermostats, mains plug sockets, fridges, kettles, etc.), one device growing in popularity is the network connected camera used for surveillance and home monitoring from distance (for pet friendly homes). Placed in strategic locations in the home (monitoring entrances, living room, kitchen, etc.), these cameras, equipped with computing resources, could perform activity recognition and transfer observed activity classes as labels to a mobile sensing device (*e.g.,* smartphone, smartwatch) carried by home inhabitants, to improve recognition of mobile sensor models. These improved sensor models can be used even when outside of camera view, such as in bedrooms or bathrooms. Relevant actions for home automation are: full body rotation, to control devices (turning left to reduce music volume and turning right to increase), walking out of a room can be a trigger to lower room temperature and to dim down the lights, sitting on the couch could turn on the smart TV, while standing up and leaving the room would turn this off. Although other interaction modes with connected devices exist, such as through voice by Amazon Alexa and Google Home, or through gestures recognised from radio waves reflections [16], these would be available only in limited spaces inside the home, near to wireless equipment. By carrying a mobile phone or smartwatch with them, users can enjoy the sensing capability of these devices anywhere in the home to control the environment seamlessly. In the following sections we present our proposed system, Vision2Sensor, demonstrating the utility of transferring vision generated labels to mobile sensor data, framed with the home automation scenario as a practical example, although this can be applicable to many scenarios, as discussed in Section 5.

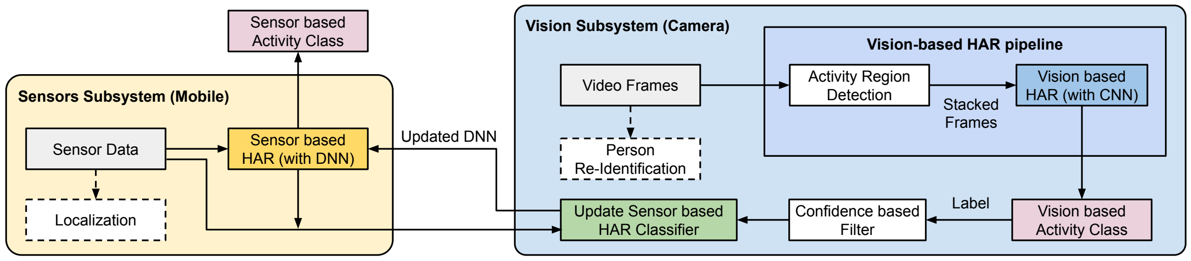

Sensor based Activity Class
Sensors Subsystem (Mobile)
Sensor Data
Sensor based HAR (with DNN)
Localization
Updated DNN
Vision Subsystem (Camera)
Video Frames
Person Re-Identification
Vision-based HAR pipeline
Activity Region Detection
Stacked Frames
Vision based HAR (with CNN)
Update Sensor based HAR Classifier
Confidence based Filter
Label
Vision based Activity Class

Fig. 2. Block components of Vision2Sensor with its two subsystems, one running on mobile device harnessing sensor data for mobile recognition, and the second running on compute enabled cameras to perform visual recognition of the person in front of it and to update the mobile sensor based model with vision generated labels.

3 SYSTEM OVERVIEW

Our Vision2Sensor system is composed of two subsystems, as presented in Figure 2: 1) the Sensors Subsystem running on smartphones or smartwatches; and 2) the Vision Subsystem, which runs on cameras with compute resources to perform visual recognition and to update the phone sensor-based DNN model.

3.1 Sensors Subsystem

The Sensors Subsystem runs on a sensing device (smartphone or smartwatch) where it performs Human Activity Recognition (HAR) continuously based on inertial sensors (accelerometer, gyroscope, magnetometer, etc.). Communication with the Visual Subsystem is initiated here, to transfer the sensor data captured in camera field of view and the sensor DNN model to be updated by the compute capable smart camera with vision generated class labels. The Sensors Subsystem routine is presented in Algorithm 1. Sensors buffering is performed only when user is in field of view since activities performed here are experienced from both perspectives (sensors and vision) and can be labeled by the vision subsystem. For this task, the Sensors Subsystem uses a Localization component to identify where the user in the indoor space. Although not implemented in Vision2Sensor since our evaluation concentrates predominantly on the quality of knowledge transfer with participants being in front of the camera during data collection, user location relative to the camera can be obtained with one of the many solutions available in the literature for mobile phone localization ([4, 7, 35, 47]). The sensor buffer is arbitrarily large, depending on how often model updates should be made, so it captures a few minutes of activities in front of the camera. Once this buffer is full or when moving outside camera view, this is sent to the Vision subsystem to generate training instances and to update the sensor based DNN (also transferred to the camera). When outside of camera view, estimations are performed entirely in the Sensors Subsystem, and sensor samples are not stored since these cannot be used for re-training the model since an external label cannot be provided. For the home automation scenario buffer size starts with a capacity of a few minutes (5 minutes in early deployment), to force more frequent labeling for training, and increases gradually once the sensor based classifier becomes more confident and training brings just incremental gains, to minutes or hours. The only condition is to have a suitable buffer capacity on the Vision Subsystem to store videos over the same time length.

Since the Sensors Subsystem initiates the communication to the other subsystem, this can be done more selectively and not continuously when in front of the camera, based on estimated utility for additional training. For instance, active learning [11] can be utilized to indicate what samples are more relevant to train on (low confidence in classification or rare occurrences of an instance variation), thus avoiding unnecessary data transfers to the camera for the rest of common samples.

Algorithm 1 Sensors Subsystem Routine

```
1: sensorBuffer ← allocateMinutesLongBuffer()
2: repeat
3:     sensorInstance ← readSensorData()
4:     location ← estimateLocation(sensorInstance)
5:     if inFrontOfCamera(location) then
6:         sensorBuffer.insert(sensorInstance, time)
7:         if sensorBuffer.isFull() then
8:             sendToCamera(sensorBuffer, sensorModel)
9:             sensorBuffer.empty()
10:        end if
11:    else
12:        if sensorBuffer not empty then
13:            sendToCamera(sensorBuffer, sensorModel)
14:            sensorBuffer.empty()
15:        end if
16:    end if
17:    if modelUpdateFromCamera then
18:        sensorModel ← getVisionBasedUpdatedModel()
19:    end if
20:    class ← sensorModel(sensorInstance)
21: until exit
```

Algorithm 2 Vision Subsystem Routine

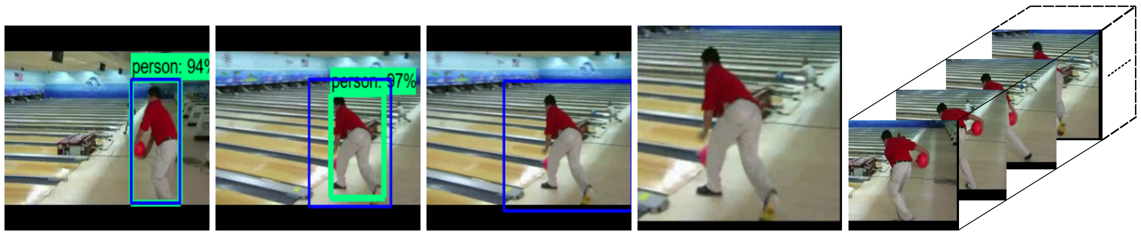
```
1: videoBuffer ← allocateLargeFIFOBuffer()
2: repeat
3:     sensorBuffer, sensorModel ← acceptReqFromMobile()
4:     videoInstances ← timeSynch(sensorBuffer, videoBuffer)
5:     for instance in videoInstances do
6:         label, classifConfidence ← videoClassifier(instance)
7:         if classifConfidence ≥ threshold then
8:             videoLabels.insert(label)
9:             sensorInstances.insert(sensorInstance)
10:        else
11:            videoLabels.insert("null_class")
12:            sensorInstances.insert(sensorInstance)
13:        end if
14:    end for
15:    sensorModel ← updateSensorModel(sensorModel,
16:                                     sensorInstances, videoLabels)
17:    sendModelToMobile(sensorModel)
18: until exit
19:
20:
21:
```

3.2 Vision Subsystem

This runs on cameras capable of performing local or in close proximity computations. Through a vision pipeline (explained further in this section), frames are normalized and stacked for HAR on visual input. The smart camera stores frames in a large FIFO (First In First Out) buffer to have the frames available for time synchronization on inertial sensor data upload from a mobile device to produce labels through vision classification. Time synchronization is performed with the phone sharing local device time on sensor data upload to align timestamps at the camera end, so all timestamps (video and sensors) can be synchronized backwards from these two relative times. The only drawback with this approach is network latency, but since this is done in a local network, we consider communication delays insignificant relative to human activity times. Algorithm 2 presents the routine for the Visual Subsystem. For each instance of inertial sensor data, a class is estimated by the vision classifier. This class is associated to the sensor data only if video instance is classified with a high confidence. If confidence is above a predetermined threshold, estimated class is associated to inertial sensor data, otherwise the class is considered a "null class". More details on handling the null class are provided in Section 3.4. After labeling instances received from a mobile device, the sensor-based HAR model is updated with these new instances and is sent back to mobile device after re-training. Cameras are commonly plugged in, hence using this to perform the training.

Another essential component of the Visual Subsystem is the Person Re-identification component, which matches the person in video with their sensor data through (person-id, device-id) pairs. There are many solutions proposed in literature for person re-identification [26], however, since the focus of this work is on transferring knowledge from vision to sensor data, and our chosen scenario reduces conditions of many people in camera view, in this work we neglect this component and assume just a single person in frames. The other advantage of performing computations locally, in the home network for the home automation scenario, is that user sensitive data is not transferred to the cloud, having the entire recognition and training chain in the local network.

As an extension to the current subsystem, to avoid irrelevant video buffering and processing of frames with no person in front of the camera, or when the person is static, a lightweight filter can be used to measure changes between frames over a time window of a few seconds. If the amount of change is below a threshold (chosen to be



(a) Early frame (b) Later frame (c) Activity region (d) Cropped frame (e) Stacked frames

Fig. 3. Activity region (blue box) is determined by combining all single frame person detections (green box) across frames.

insensitive to small light fluctuations and noise in image sensor data acquisition), then the frame is discarded. This filter avoids unnecessary computations by reducing the classification on static activities (the same class can be carried across similar set of frames) and saves valuable computing resources, by buffering fewer frames and thus performing fewer classifications.

3.3 Vision based Human Activity Recognition

The vision HAR pipeline is formed of 1) region detection where an activity takes place and 2) activity recognition over the detected region. For the detection part, the established MobileNet [15] architecture with the SSD detection technique [28] is used, which is fast due to depth-wise separable convolutions and the efficient single shot multiple detections (SSD) component. Network weights have been pretrained on the COCO dataset [27]. As output, this network gives bounding boxes around a detected object, together with confidence score.

3.3.1 Activity Region Detection. The detection framework is applied to extract person in video with a surrounding region for each frame with the detected person in the center. This region is determined by the overall region covered by this person throughout the entire sequence of frames. The coverage area is the complete overlap of all bounding boxes across all frames of activity.

Figure 3 shows frames with a detected person (green box) and the activity region (blue box) as determined by the intersection of all the single frame detections. This activity region (blue box) is cropped and used as input to a custom CNN that classifies human activity further in the pipeline.

The activity region box is observed during the first pass across all frames – Figure 3 shows the activity region (blue box) expanding with each frame by accumulating person detection regions (green box). In the second pass each frame is cropped to the size of the activity region. For cases when the activity region box exceeds the frame boundaries, the cropped image gets padded with additional black pixels to maintain a fixed size for the activity region. The use of an activity region ensures that all cropped images are precisely the same dimension to feed to the activity recognition CNN. Cropping and centering on the person assures a good local exposure of the person. Our experiments using the UCF-101 dataset [43] show that performing activity recognition on the whole frame is less effective, so this early normalization stage with the activity region detection is important for accuracy.

3.3.2 Activity Recognition. The set of stacked cropped frames (activity region) is used as input to a CNN, which is trained to classify specific set of activities, depending on use-case. Several consecutive frames containing a detected activity are stacked to form a third dimension and thus building a tube (clip) of frames as input to the HAR CNN. Because of different frame rates, these accumulate to 1.5-2.5 seconds, which is enough for achieving good recognition accuracy and for fast inference time. A frame rate can also be imposed and interpolating frames to maintain a fixed input size to our CNN. The formula for sampling ratio s is given by equation (1), where f is the frame rate in fps, d is the duration of activity in seconds and n is the number of frames for the activity tube.

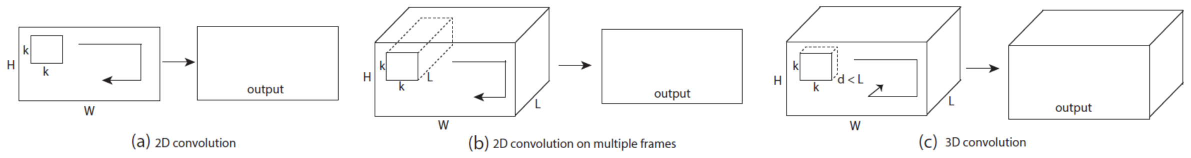


Fig. 4. Different operation mode between 2D- and 3D convolutional layers which impact the features they build on. 2D convolutions capture stronger correlations within frame, building on spatial locality, while 3D convolutions find correlations across time dimension more relevant.

$$s = \frac{df}{n} \tag{1}$$

Figure 3e illustrates how the cropped frames are stacked in a third dimension (time). This technique allows the use of 3D convolutions which are capable of capturing spatio-temporal dependencies and have been shown in previous work ([5, 46]) to achieve excellent results, which inspires our selection of network architectures.

3.3.3 Network Architectures. In order to get a good understanding of what neural network architecture solution to consider for our activity recognition task, we evaluate both 2D convolutional networks and 3D convolutional networks. The former is faster since it treats stacked frames independently, hence not exercising correlations across the time domain too, but it is still a good candidate for embedded computing systems due to its speed. The latter is slower in convolving the kernel across all three dimensions, but can identify features across time, which make it more accurate in activity recognition. These two candidates are designed to use small kernel size (3 × 3) for the spatial convolution, which is a popular choice in many other CNN architectures (VGG, MobileNet, ResNet, etc.), deployed for object [42] or activity recognition [46], due to their capability to extract fine grained features, also relevant for our recognition task. We consider two networks inspired from both image classification networks (2-D convolutions) and from the state-of-the-art in activity classification (3-D convolutions), presented in Table 1 and Table 2 respectively.

Table 1. CNN taking 10 images as input (30 RGB channels) for 2D convolution

input: stacked images (180 × 220 × 30)		
Layer name	**Kernel dimension** ($h \times w$)	**Output dimension** ($H \times W \times C$)
Conv1a	3 × 3	180 × 220 × 64
Conv1b	3 × 3	180 × 220 × 64
Pool1	2 × 2	90 × 110 × 64
Conv2a	3 × 3	90 × 110 × 64
Conv2b	3 × 3	90 × 110 × 64
Pool2	2 × 2	45 × 55 × 64
Conv3a	3 × 3	45 × 55 × 64
Conv3b	3 × 3	45 × 55 × 64
Pool3	2 × 2	23 × 28 × 64
FC4	-	64
FC5	-	num of activity classes

While the 2D convolutional network interprets stacked images the same as one image with multiple channels (10 times more channels in Table 1), the 3D convolutional network (Table 2) stacks the RGB frames up on a third

Table 2. CNN taking 16 stacked images input for 3D convolution

Input: stacked images ($16 \times 112 \times 112 \times 3$)		
Layer name	**Kernel dimension** ($d \times h \times w$)	**Output dimension** ($D \times H \times W \times C$)
Conv1a	$3 \times 3 \times 3$	$16 \times 112 \times 112 \times 64$
Pool1	$1 \times 2 \times 2$	$16 \times 56 \times 56 \times 64$
Conv2	$3 \times 3 \times 3$	$16 \times 56 \times 56 \times 128$
Pool2	$2 \times 2 \times 2$	$8 \times 28 \times 28 \times 128$
Conv3a	$3 \times 3 \times 3$	$8 \times 28 \times 28 \times 256$
Conv3b	$3 \times 3 \times 3$	$8 \times 28 \times 28 \times 256$
Pool3	$2 \times 2 \times 2$	$4 \times 14 \times 14 \times 256$
Conv4a	$3 \times 3 \times 3$	$4 \times 14 \times 14 \times 512$
Conv4b	$3 \times 3 \times 3$	$4 \times 14 \times 14 \times 512$
Pool4	$2 \times 2 \times 2$	$2 \times 7 \times 7 \times 512$
Conv5a	$3 \times 3 \times 3$	$2 \times 7 \times 7 \times 512$
Conv5b	$3 \times 3 \times 3$	$2 \times 7 \times 7 \times 512$
Zero_pad1	$1 \times 1 \times 1$	$2 \times 9 \times 9 \times 512$
Pool5	$2 \times 2 \times 2$	$1 \times 4 \times 4 \times 512$
FC6	-	4096
FC7	-	4096
FC8	-	num of activity classes

dimension indexing the order in time of each RGB frame to form a 3D space. The difference between 2D- and 3D convolution is illustrated in 4, where kernel dimensions d, h, w represent temporal dimension, height, width respectively and for the output dimensions D, H, W, C represent temporal dimension, height, width, number of output channels. The 3D architecture is identical (except for the last fully connected layer) to the 3D network presented in [46], which achieves very good performance for activity classification. In the evaluation section, we show how training this 3D CNN with transfer learning from the large UCF-101 activity recognition dataset achieves best performance in our video processing activity recognition pipeline (Section 4.1).

3.4 Null Class Handling

It is impossible to anticipate all possible activities to train the vision subsystem on. There are often situations in practice of corner-case activities, such as transitioning between activities and other irregular movements outside of the relevant set of activities, which need to be recognized accordingly and not interpreted as one of the relevant activities. For these situations we assign a *null class* to be representative of any instance that is not similar to our relevant activities. This is determined based on vision classification confidence to signal when an instance is not much similar with instances observed during training. Although there is no perfect solution to capture confidence of deep neural networks, previous research have shown that on calibration, last layer activations (after softmax) can be a good indicator of how similar one instance is to others in the training set [13]. This is intuitive since during training neural networks are optimized to produce an output vector with value 1 for the element representing the class and zero for all the other elements in the output vector. The more similar an input is to one of the instances seen in training, the closer to a full unit one class will get in the output vector. Other solutions for measuring neural network confidence include density modeling [45], ensembles based snapshot [17], Bayesian Neural Networks and others [32].

Using the final layer activation (output vector) as a proxy for classifier confidence, we choose a threshold above which sensor instances take the class estimated by the vision classifier, otherwise these are classified as null class (see Algorithm 2). It is worth noting that in our current system design, the null class is used with the sensor based classifier alone, the vision classifier estimating only from the relevant classes for the task. This class has the effect of avoiding deterioration of sensor classifier from potentially significantly bad estimations from the vision component (the Confidence based Filter in Figure 2), and can ignore activities the system is not trained to recognize (outside of the set of 5 actions considered for the home automation scenario). The value of this threshold is a hyperparameter that influences the sensor based classifier robustness and sensitivity (to approximate or precisely correct) activities.

3.5 Sensor based Activity Recognition

Previous work has shown that deep learning produces more accurate human activity recognition on sensor signals [34]. We adopt the same approach with a 2-hidden-layer deep neural network for activity recognition with concatenated sensor data input. Figure 5 shows the network architecture and dimension of the input vector – this varies depending on the number of sensors used as input.

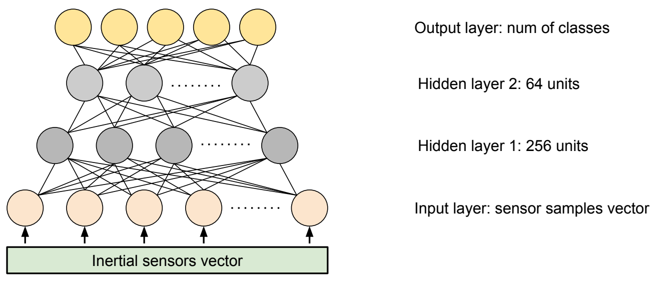

Output layer: num of classes
Hidden layer 2: 64 units
Hidden layer 1: 256 units
Input layer: sensor samples vector
Inertial sensors vector

Fig. 5. Deep Neural Network architecture (1-D Convolutional and Fully-connected layers) custom to recognize 6 classes (5 activities and the null class) for the home automation scenario.

We call the model that performs classification on sensor input DNN because it can contain a variety of structures that have proven efficient for processing sensor signals, such as convolutional neural networks [39], fully-connected networks [36] and recurrent neural networks [14]. We used convolutional layers and fully-connected layers, with an architecture determined by hyperparameter search on the activity recognition task.

4 EVALUATION

This section presents the experiments conducted to validate the robustness of Vision2Sensor components. We identify the best CNN classifier for activity recognition of 5 classes randomly selected from the UCF-101 dataset [43]. This classifier is then trained on our activity recognition dataset with randomly initialized network weights, and with knowledge transfer from a pre-trained model on Sports-1M dataset [46] – freezing trained features and fine-tuning final layers on our dataset. Estimations generated with this vision based classifier are assigned as labels to the corresponding sensor data for training the sensor based classifier. These are assessed on accuracy and performance on real hardware.

4.1 Vision Recognition Performance on UCF-101 Dataset

We test the performance of our vision based activity recognition classifiers on the popular UCF-101 dataset using the following randomly selected classes: "Bowling", "Hammering", "GolfSwing", "TennisSwing" and "YoYo". The UCF-101 dataset contains short videos (around 4-7 seconds) of 101 different activity classes. We choose this dataset because it captures a good number of instances for each activity, which is important to objectively assess the performance of a classifier.

To evaluate generalization, test videos are chosen from different groups in the same activity class, characterized by different scenes (people, angle, background, etc.). Implementation is done in Keras with Tensorflow backend,

Table 3. Comparing 4 CNN architectures trained to recognize 5 classes from the UFC-101 dataset, shows a 3D CNN with transfer learning is superior.

Method	**Accuracy**
Stacked 10 frames 2D	66.2%
Stacked 16 frames 2D	67.6%
Stacked 16 frames 3D (trained from scratch)	70.5%
Stacked 16 frames 3D (Sport-1M pre-training)	**99.2%**

and using adam [20] as optimizer, with a learning rate of 1e-5 and β_1=0.9, β_2=0.99 for the exponential decay rates of the moving averages. Rectified Linear Unit (ReLU) [30] is used as activation function. Table 3 shows the performance achieved with different CNN architectures on the selected 5 classes of the UCF-101 dataset.

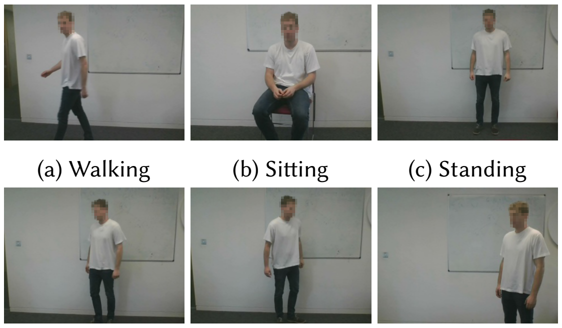
These results show that 3D convolutions outperform 2D convolutions, which confirms their strong capability to extract meaningful features for activity recognition. Although ideal for edge computing due to their speed of inference, 2D convolutions are not considered further in this evaluation since their accuracy is not robust to rely on for knowledge transfer to sensor data. The best result is achieved by performing transfer learning between vision recognition tasks. We take the classifier introduced by Karpathy et al. [46], with weights trained on Sports-1M dataset [19] (currently the largest video classification benchmark). This architecture is presented in Table 2. We retrain the last two layers of the model on the subset of 5 activities examined in this section, while the weights for the other layers are frozen as they have already shown excellent performance for feature extraction in previous work [46]. The pre-trained model is capable of achieving a general performance of 82% accuracy on the whole UFC-101 dataset as shown by Tran et al., [46] and Carreira et al., [5]. These early results, on 5 classes of UFC-101 indicates that our selection of 5-class vision classifier is accurate and reliable.

It is important to notice that our chosen architecture is capable of performing robust recognition on more than the 5 activity classes chosen here, having been validated for 101 classes on UFC-101 dataset [46], which indicates that we can extend this to capture other more complex activities for our envisioned scenario of home automation.

4.2 Data Collection

This section describes our data collection, which is used to fine-tune the vision classifier on the classes used in our evaluation scenario and to time synchronize with inertial sensor data collected on a smartphone with view of the same activities from a different modality perspective. This is the first dataset to be collected for the purpose of knowledge transfer from vision to inertial sensors, using a fixed camera as vision modality and inertial sensor data collected on a smartphone carried in pocket. Our dataset is collected in a controlled environment with only small changes in illumination, background and camera angle, although still realistic conditions for smart office buildings where artificial lighting is dominant.

We explore 5 simple but relevant human activities: "*Walking*", "*TurnLeft*", "*Standing*", "*TurnRight*" and "*Sitting*". These can be useful to control connected devices in smart homes for instance. Although these captures a relatively constrained number of activities, they are enough to enable some innovative control methods in home environments (as described in the section 2), while other previous research on activity recognition also explore a small set of classes to highlight interesting observations [44] and not to dilute the focus of the work with many irrelevant classes. Cumulatively, our dataset captures about 2 hours of human activity from two participants performing a full set of activities, well balanced between classes and well annotated both in video and sensor signals. Since we are interested in the events performed in front of the camera, we divide this data in videos and inertial data associate to them. These are between 5 seconds long for the fastest transitions in front of the camera when walking and up to a minute for the more static activities like sitting. Each video captures a different setup, with changes between videos of angle to the camera, position in scene, speed (of walking and turning), posture and even clothing. For each class we have the following number of videos: sitting (35 videos), sitting (32 videos),


(a) Walking (b) Sitting (c) Standing

(d) TurnRight (e) TurnLeft (f) Standing, other

Fig. 6. Instances of video (frames) for different activity classes.

turning left (31 videos), turning right (33 videos) and walking (80 videos), with their associated inertial sensor data. This is enough data to fine-tune the transferred vision model and to assess the impact of knowledge transfer between sensing modalities (vision and sensors).

Video data collection is controlled with the OpenCV computer vision library in python using a frame rate of 20fps and a resolution of 320x240 with 3 channels (RGB). Sensor data (accelerometer, gyroscope and magnetometer) is collected through a custom Android application running on a HTC-U11 device. Figure 6 shows some examples of frames from the collected data for several classes. The variations in setup mentioned before is also seen in Figures 6c and 6f for the "standing" class, showing difference between videos in (positions in the scene, angle to the camera, camera positioning and body exposure), which makes the dataset more complicated.

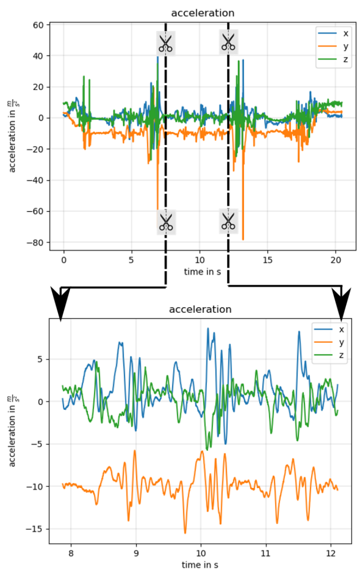
4.3 Data Annotation and Pre-processing

Synchronization in Vision2Sensor is done based on timestamps communicated between mobile device and camera as discussed in Section 3.2, however, because we wanted to work with this data offline to experiment different classifiers, we needed another method to time-synchronize data collected from all these sensing perspectives. We achieved this goal by introducing a distinctive signal noticeable on both the accelerometer and camera – an intensive shake of the phone. This was helpful for the human annotator to identify and guarantee that association between sensor data, video frames and ground-truth labels is done properly. Figure 7 shows how this alignment is achieved to provide the ground-truth information to instances in our dataset. Other sensing modalities (gyroscope, magnetometer) are cut to the same timestamps as the accelerometer as indicated in Figure 7b.

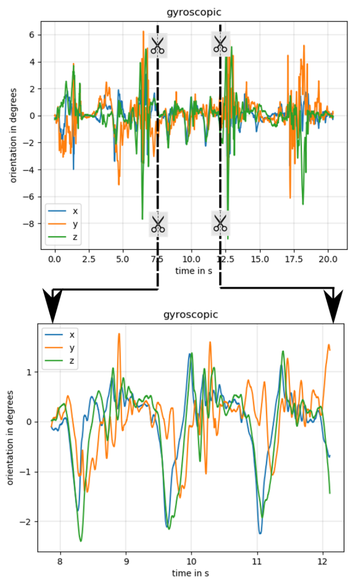
Because the Android API produces sensor samples on an event based at varying intervals of time between samples, we first normalize the sampling rate of sensors to 100 Hz (10 milliseconds between each point), achieved by taking the nearest neighboring points (in time domain) of each sensor data. This signal is then split into equally-sized time windows capturing 2 seconds of activity.

4.4 Vision Subsystem Performance on Our Activity Recognition Dataset

In the previous section we saw that a transfer learning 3D CNN trained on another dataset can achieve exceptional performance on our restrained task of 5 randomly selected classes of the UCF-101 dataset 4.1. This is restricted to just 5 classes to match the number of classes in our demonstration dataset (with all sensing modalities, as indicated above) for the multimodal activity recognition scenario. Our dataset captures 886 instances of video and sensor samples, each activity instance being 2 seconds long (for granular and interactive estimations). This is split randomly into training set (80%) and test set (20%).



(a) Acceleration sensor data



(b) Gyroscopic sensor data

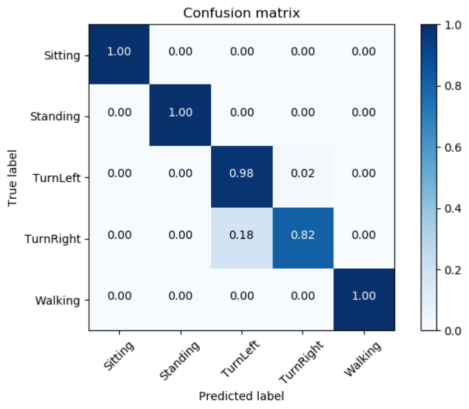
Fig. 7. The process of aligning sensor data with video frames by the human annotator to provide ground truth labels to our training set, observing an intensive shake of the phone on all modalities.

To expand the dataset, traditional image transformations are applied: a) vertical flipping, b) horizontal flipping. Furthermore, to make the network more robust, different kinds of noise (Gaussian and salt and pepper) are added to the images as data augmentation. The hyper-parameters are the same as the ones used in training the network on the UCF-101 dataset, with the only exception being a higher learning rate of 1e-4 in conjunction with the Adam optimizer. Table 4 shows the performance of our chosen 3D-CNN (selected as presented in Section 4.1), where extended dataset refers to the data augmentation in addition to the original samples. We can see that data augmentation plays an important role in our network generalization, accuracy increasing by 2% on the test set.

Table 4. Accuracy on our activity recognition dataset, both with original and with data augmentation extended training sets for our chosen 3D CNN for vision based activity recognition.

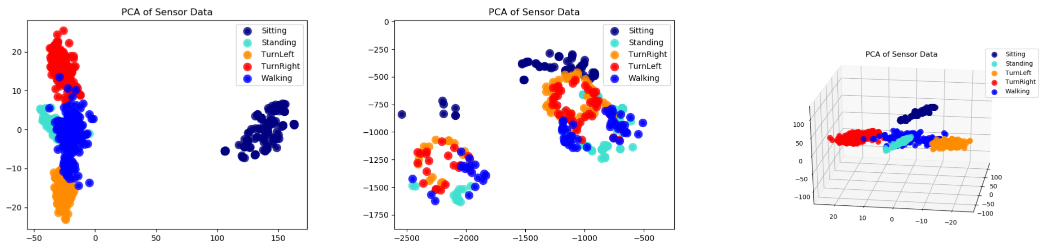
Training set	**Test accuracy**
Original set	90.35%
Expanded set	**92.38%**

These results show that by introducing noise and transformations as additional samples to a training set, the performance of our visual classifier is significantly improved. Figure 8 shows the confusion matrix for the 3D CNN trained with knowledge transfer from Sports-1M dataset, refined on our extended training dataset with data augmentation and tested on the vision component of our test set. The matrix gives insight into the types of error made by the predictor.



True label	Sitting	Standing	TurnLeft	TurnRight	Walking
Sitting	1.00	0.00	0.00	0.00	0.00
Standing	0.00	1.00	0.00	0.00	0.00
TurnLeft	0.00	0.00	0.98	0.02	0.00
TurnRight	0.00	0.00	0.18	0.82	0.00
Walking	0.00	0.00	0.00	0.00	1.00

Fig. 8. Confusion matrix of the 3D CNN for vision based activity recognition after fine-turning on our dataset the knowledge transferred model trained on Sports-1M dataset.



(a) PCA in 2D for sensor samples without the magnetic sensor

(b) PCA in 2D for sensor samples with the magnetic sensor

(c) PCA in 3D for sensor samples without the magnetic sensor

Fig. 9. PCA applied to sensor data showing mixed clusters of activities which indicates that a deep neural network is the ideal candidate for this hard classification task.

Some confusion exists only between the classes "TurnRight" and "TurnLeft". This observation is expected since the visual appearance of both actions is very similar. Also, the information about the direction of movement across frames in the activity tube is gradually lost through the 3D-pooling layers of the network. Nevertheless the accuracy is still very high for both classes, suggesting that the system is capable of dealing with the complexity of our task.

4.5 Sensors Subsystem Performance on Our Activity Recognition Dataset **(Ground-truth Labels)**

Inferences on sensor data is performed with a Deep Neural Network (DNN) using the architecture shown in Figure 5. A vector containing the concatenated sensor data from acceleration, gyroscope and magnetic sensors in x, y, and z coordinates is used as input. To visualize such high dimensional features, Figure 9 shows these data points by applying Principal Components Analysis (PCA) to reduce the high dimension of an input vector to 2D or 3D values, for with and without the magnetic features. These representations show the difficulty of performing the class estimations with a simple classifier. However, these are the kind of problems DNNs work well on.

Table 5. Baseline assessment of the sensors based DNN on our labeled dataset, with and without the magnetometer.

Instance features	Testing accuracy
Acceleration & Gyroscopic & Magnetic sensors	95.4%
Acceleration & Gyroscopic sensors	**97.7%**

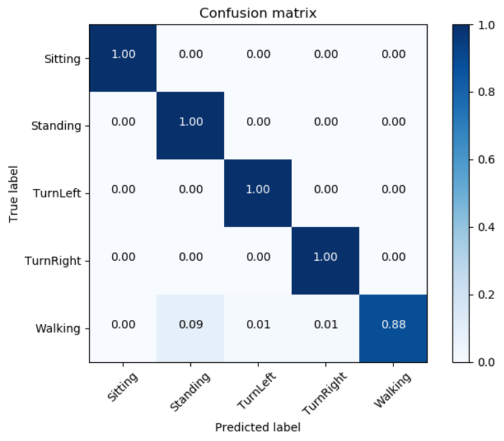


Fig. 10. Confusion matrix for sensor system using just acceleration and gyroscopic trained on the ground truth labels.

To assess the best performance of our chosen DNN for sensor data, we train the model on our sensors dataset well labeled with activity classes, following a similar split (80% : 20%). Table 5 shows the test accuracy for the classification on the 2 different combinations of modalities indicated before (with and without the magnetometer sensor). We can see that adding more sensors does not necessarily translate into better accuracy, as observed the best DNN input is one using only acceleration and gyroscope signals (Figure 10 presents the confusion matrix for this combination).Adding the magnetometer as a third modality to the DNN input decreases the accuracy. This is due to the hard anchoring in the geographical coordinates from magnetometer, whereas out dataset captures slight deviations in the orientation of performing each activity. This can affect by not covering the entire orientation space uniformly in training set, so for that reason we exclude it from our further exploration.

As observed, the classification using sensor based DNN can operate a high accuracy, confusions appearing only between the classes "Walking"/"Standing" and "Walking/(TurnLeft", "TurnRight"), which is expected since these are spatially close in the 2-dimensional PCA, as observed from Figure 9. This experiment asserts that our sensor DNN is robust to achieve high accuracy on sensor data inputs.

4.6 Sensors Subsystem Performance on Automated Labels **(Vision Generated Labels)**

After observing that our sensors based DNN has a good accuracy in training with well labeled data, we now train this with labels predicted by the vision classifier to investigate the impact of bad label estimations from the Vision Subsystem onto the Sensors Subsystem. We simulate a controlled bad behaviour of the vision based labeling subsystem by altering the quality of input images to the vision classifier. This is done by adding a controlled amount of Gaussian noise to image in each activity tube, effectively reducing the accuracy of the vision classifier. To control uncertainty in vision generated labels, with a clean interpretation of the accuracy points, we avoid the unpredictable impact of providing a null class label (when vision classifier confidence below threshold). We do this by lowering the threshold confidence to 0.29, which is sufficient for all our vision instances to be

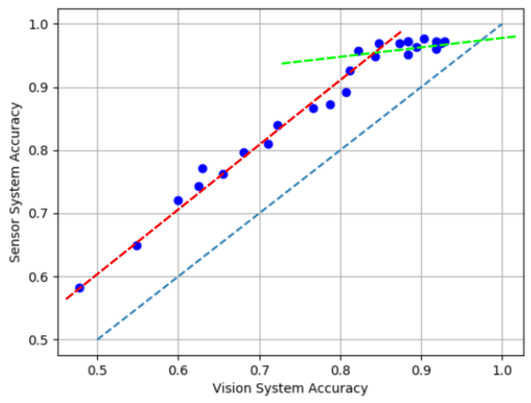


Fig. 11. Correlation between transferred labels quality (vision accuracy) and sensor model accuracy trained on vision labeled data

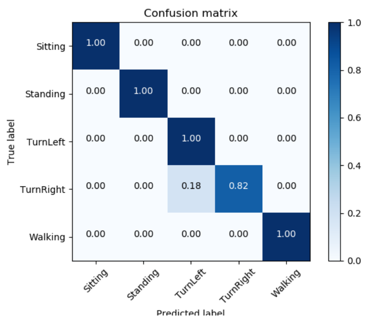
matched with one of the five determined classes, so a null class will not be produced in this experiment. This is also acceptable since our dataset captures only clean, single activity in each instance, without any corner-case activity (activity overlapping, ambiguous or unstructured activities, etc.). We use the vision classifier trained on clean instances and estimate on noisy instances to induce vision accuracy degradation in this controlled manner, associating estimations as labels to time-synchronised sensor data.

Testing the sensor system on the newly provided instances (with vision generated labels) is done by training on these instances and testing on instances with ground-truth information (human annotated). Dependency between sensors uncertainty and vision induced uncertainty is examined by comparing the known quality of vision labels (assessed on a vision test set) to that of the sensor system trained on the noisy data and tested against the ground-truth. Figure 11 shows the ground-truth test accuracy of the sensor model trained on labels generated by the vision system, plotted on the accuracy of vision labels. Each blue point represents a retraining of the sensors based DNN, as indicated above.

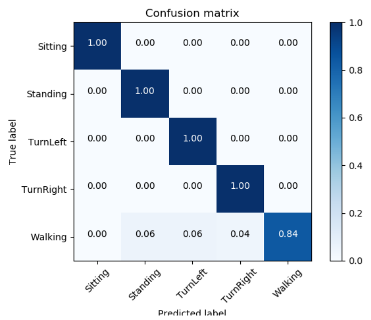
These results show that the sensor DNN accuracy evaluated on ground-truth data remains consistently higher than the quality of its training data (accuracy of the labels generated by the vision system), filtering the wrongly labeled data on its own during training, as their noise may not produce consistent patterns across all instances of the same class. The effect of this is a very slow linear decrease in accuracy for training with generated labels that have an accuracy (on the vision test set) higher than about 82% (green line) and a slightly steeper linear decrease below that point (red line trend). Although there is a bit of noise in the accuracy of sensor based DNN on different quality of labels, due to random initialization and potentially finding local optimal, there is a clear trend for the two correlations.

Figure 12 shows the confusion matrix of the vision system, which generates the labels for training the sensors classifier, and the confusion matrix of the produced sensor model across several points on the correlation between the two (visualised in Figure 11).

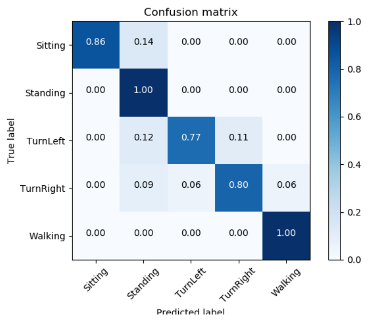
These cases of decreased labels quality cause rising confusion between classes which are already hard to classify for the sensor system (comparing Figures 12b and 10) for the class "Walking" for instance. Furthermore, sensors prediction remains robust for classes which are easy to classify like "Sitting" (Figure 9a), even if there is a considerable confusion for this class in the generated labels from the vision system (Figures 12c and 12d).



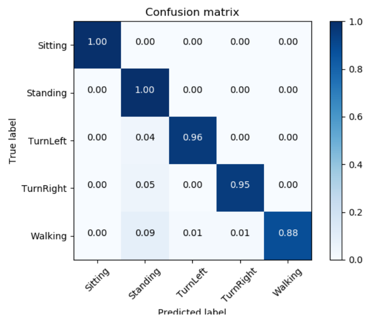
(a) Generated labels 1, 92.38% accuracy



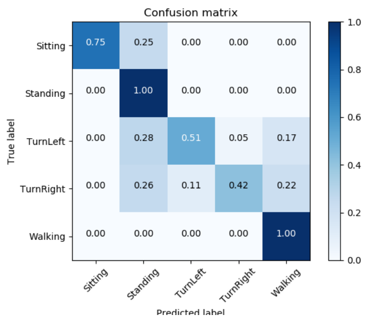
(b) Sensor system 1, 97.1% accuracy



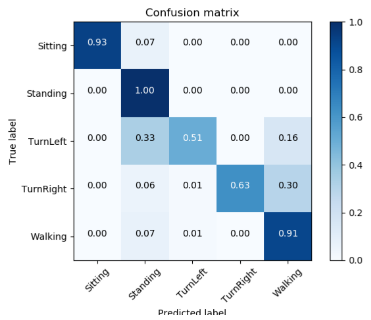
(c) Generated labels 2, 84.8% accuracy



(d) Sensor system 2, 95.2% accuracy



(e) Generated labels 3, 61.9% accuracy



(f) Sensor system 3, 74.2% accuracy

Fig. 12. Confusion matrices for generated labels (right) and sensor system trained on them (left)

Nevertheless, it can be observed by comparing the matrices in Figure 12e and Figure 12f, that for significant lower accuracy in labels, there is a strong correlation between labels and sensor confusion. This is the case because with a large amount of wrongly labeled data some unexpected patterns begin to emerge, which are picked up by the sensor network in training.

4.7 Performance on Hardware

For hardware evaluation we considered a scenario of running embedded GPUs close to the camera, such as the NVIDIA Jetson TX2 GPU on chip. For this, we evaluated the performance of running our proposed CNN on the Jetson TX2. For inference time, this is just below 400ms (Table 7), which is low enough to be an insignificant cost for running vision inferences. Also, training the user activity recognition DNN on instances collected while exposed to the camera is minimal on the Jetson TX2, Table 6 shows that forward pass of the DNN with different batch sizes. Training the DNN for 10 epochs is performed in about 3 seconds with 1024 samples, which is acceptable considering that this amount of samples is collected over a few minutes in front of the camera.

Table 6. Forward and backward pass of the sensor DNN on the Jetson TX2 with varying batch sizes (in seconds).

Sensor-DNN	batch=1	batch=10	batch=64	batch=128
Training Sensor-DNN	0.126	0.135	0.164	0.171

Table 7. Forward pass of the vision CNN on the Jetson TX2 with varying batch sizes (in seconds).

Vision-CNN	batch=1	batch=10
Inference Vision CNN	0.374	0.834

These results indicate that, Vision2Sensor can run in close to real time for DNN updates and user model enhancement with minimal delay. In these results we are not considering network communication delays and computation costs that may slow down our inference pipeline.

5 DISCUSSION

The results for training the sensor based DNN with vision generated labels show that sensor based predictions remain at a very high accuracy level even when vision labeling quality drops close to 82%. Below this labeling accuracy, the impact on the sensor based model begins to be noticeable, although still performing substantially better than the quality of provided labels, dropping linearly with the decreasing labeling quality, as seen from Figure 11. This is not surprising since neural networks are designed to be robust to some noise, and considering the small number of classes, the network identifies the right class features from the other well labeled instances. As long as the majority of instances are well labeled, the sensor based model performance is superior to the accuracy of transferred vision labels. For that reason, a decent vision based classifier suffices to produce just the right amount of quality labels, so a smaller models or even multi-task models can be considered.

As with the home automation scenario described before, Vision2Sensor is intended to work anywhere some cameras are already deployed to monitor the environment, though not evenly spread throughout, such that there is still need for additional sensing capability from wearables, not just with cameras alone. The opportunistic training, when in camera field of view, makes Vision2Sensor useful in scenarios where sensing modalities need to be adapted to new users, such as for museums, to calibrate the recognition system to each visitor exploring the museum, with additional recalibrations when near to a surveillance cameras; in hospitals to monitor patient activity or to signal to medical staff the evolution of a patient, medical staff can also use gestures captured by wearable sensors to control the medical equipment so they can make their work more efficient; in a smart gym environment, an extension of Vision2Sensor can enable automatic logging of exercise activity and the number of repetitions, and also offer advice on the quality of exercise execution. As seen in Section 3.3, our vision classifier can be trained to recognise many more classes, this giving flexibility to Vision2Sensor to adapt to larger recognition tasks and even dynamically add more classes to the recognised set.

One key feature of Vision2Sensor is data privacy, since all computations are performed in the secure local network of the deployment environment (home network, university network, hospital network), with sensor and video data not leaving the private network for the cloud. This makes it possible to exploit local computation resources, assuming a secure link between the mobile device and the GPU embedded camera. This computation paradigm will grow in importance with compute enabled smart cameras reaching the market.

6 RELATED WORK

Significant progress has been made in computer vision for human action recognition in videos. Successful methods have to extract spatio-temporal features beyond just object recognition. Recent research shows strong performance from 3D-CNNs [18], where convolutions are applied on 3D feature maps from both the spatial and temporal dimensions. These 3D-CNN models can capture motion information over multiple a sequence of frames. Another CNN based approach [19] for action recognition uses various frame-level fusion methods over the time dimension to take advantage of spatio-temporal information. Another 3D-CNN architecture, which using 3×3×3 kernels, shows good generalization and outstanding performance across a range of video analysis tasks [46]. They show experimentally that 3D-CNNs are powerful feature learning machines capable of modeling appearance and motion simultaneously. Inflating a trained 2D-CNN to a 3D-CNN, to capture the time dimension, has proven beneficial to accelerate the design and training of 3D-CNN models [5].

An earlier solution for synchronizing sensor data collected with a bracelet and video frames has been presented in [33], the main focus being there on synchronization between the two modalities, looking at a small set of hand gestures. Transfer learning [50] has been used before mostly in the computer vision, where a CNN trained on one dataset, such as on ImageNet [22], has been used successfully to perform estimations on another datasets by fine-tuning only the final layers of the model [31], thus accelerating training and generalization on the new task.

Previous works have shown inertial sensors from smartphones and smartwatches being used for activity recognition [24], with different supervised [6, 35] and unsupervised [2] methods, and other specialised machine learning solutions [24, 41]. Deep learning has just started being used for this task, exploring accelerometer and gyroscope for locomotion recognition [39], on activities related to daily living [49] and other activities across domains with multimodal architectures [34, 36]. Transfer learning has also been applied to HAR [8], but mostly within the same sensing modalities on similar datasets.

In a survey on transfer learning for sensor data, D. Cook et al. [8] observe that not much research effort has been invested in transferring knowledge from vision to sensor data or between radically different sensing modalities in general. Using transfer learning across completely different sensing modalities, a sensor based model has been shown to produce good enough labels to train another sensor based model running with sensor data of a different characteristic [23].

7 CONCLUSIONS

In this work we address one of the fundamental problems hindering the expansion of mobile sensing, not having enough labeled sensor data for training. Our system, Vision2Sensor, performs opportunistic mobile sensor data labeling, by relying on knowledge transferred from a more mainstream sensing modality, vision, to provide human activity recognition labels to inertial sensors (accelerometer, gyroscope), while in the field of view of a camera with computing resources. These labels are used to train a sensor based mobile activity recognition classifier, which runs efficiently on the mobile device outside of camera field of view. Our results show that potentially erroneous labels introduce by the vision modality are not a concern to transferring knowledge to other sensing modalities, our sensor based deep neural network classifier withstanding errors of about 15% in vision labels. We also show that Vision2Sensor runs efficiently and in real-time on resource constrained devices (such as on smart cameras).

REFERENCES

[1] Ionut Andone, Konrad BÅĆaszkiewicz, Mark Eibes, Boris Trendafilov, Christian Montag, and Alexander Markowetz. 2016. Menthal: a framework for mobile data collection and analysis. In *Proc. UbiComp*. ACM.

[2] Ferhat Attal, Samer Mohammed, Mariam Dedabrishvili, Faicel Chamroukhi, Latifa Oukhellou, and Yacine Amirat. 2015. Physical human activity recognition using wearable sensors. *Sensors* 15, 12 (2015).

[3] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury. 2009. Surround-Sense: Mobile Phone Localization via Ambience Fingerprinting. In *MobiCom*. ACM.

[4] Paramvir Bahl and Venkata N Padmanabhan. 2000. RADAR: An in-building RF-based user location and tracking system. In *Proc. INFOCOM*. IEEE.

[5] Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 4724–4733.

[6] Pierluigi Casale, Oriol Pujol, and Petia Radeva. 2011. Human Activity Recognition from Accelerometer Data Using a Wearable Device. In *IbPRIA*.

[7] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. 2010. Indoor localization without the pain. In *Proc. Mobicom 2010*. ACM.

[8] Diane Cook, Kyle D. Feuz, and Narayanan C. Krishnan. 2013. Transfer learning for activity recognition: a survey. *Knowledge and Information Systems* (2013).

[9] R. Silva Da, H. Rodgers, L. Shaw, F. van Wijck, S. A. Moore, D. Jackson, R. Francis, L. Sutcliffe, M. Balaam, T. Ploetz, et al. 2018. Wristband accelerometers to motivate arm exercise after stroke (WAVES): Activity data from a pilot randomised controlled trial. *Elsevier* (2018).

[10] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. 2014. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable. In *NDSS*.

[11] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2016. Deep Bayesian Active Learning with Image Data. In *Proc. NIPS*.

[12] Yu Guan and Thomas Ploetz. 2017. Ensembles of Deep LSTM Learners for Activity Recognition using Wearables. *PACM IMWUT* (2017).

[13] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *PMLR*.

[14] Nils Y Hammerla, Shane Halloran, and Thomas Ploetz. 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880* (2016).

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[16] Chen-Yu Hsu, Rumen Hristov, Guang-He Lee, Mingmin Zhao, and Dina Katabi. 2019. Enabling Identification and Behavioral Sensing in Homes using Radio Reflections. In *CHI*. ACM.

[17] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot: Train 1, Get M for Free. In *ICLR*.

[18] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 2013. 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence* 35, 1 (2013), 221–231.

[19] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 1725–1732.

[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[21] Kundan Krishna, Deepali Jain, Sanket V. Mehta, and Sunav Choudhary. 2018. An LSTM Based System for Prediction of Human Activities with Durations. In *ACM IMWUT*.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[23] Marc Kurz, Gerold Hölzl, Alois Ferscha, Alberto Calatroni, Daniel Roggen, and Gerhard Tröster. 2011. Real-time transfer and evaluation of activity recognition capabilities in an opportunistic system. *Machine Learning* 1, 7 (2011).

[24] Oscar D Lara and Miguel A Labrador. 2012. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials* 15, 3 (2012).

[25] Oscar D Lara, Alfredo J Pérez, Miguel A Labrador, and José D Posada. 2012. Centinela: A human activity recognition system based on acceleration and vital sign data. *Pervasive and mobile computing* 8, 5 (2012).

[26] Bahram Lavi, Mehdi Fatan Serj, and Ihsan Ullah. 2018. Survey on Deep Learning Techniques for Person Re-Identification Task. In *arXiv:1807.05284v3*.

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[28] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.

[29] Aleksandar Matic, Venet Osmani, Alban Maxhuni, and Oscar Mayora-Ibarra. 2012. Multi-modal mobile sensing of social interactions. *2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops* (2012), 105–114.

[30] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.

[31] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks. In *Proc. CVPR*. IEEE Computer Society.

[32] G. Papadopoulos, P.J. Edwards, and A.F. Murray. 2001. Confidence estimation methods for neural networks: a practical comparison. *Transactions on Neural Networks* (2001).

[33] Thomas Plötz, Chen Chen, Nils Y. Hammerla, and Gregory D. Abowd. 2012. Automatic Synchronization of Wearable Sensors and Video-Cameras for Ground Truth Annotation – A Practical Approach. *International Symposium on Wearable Computers* (2012).

[34] Valentin Radu, Nicholas D Lane, Sourav Bhattacharya, Cecilia Mascolo, Mahesh K Marina, and Fahim Kawsar. 2016. Towards multimodal deep learning for activity recognition on mobile devices. In *Proc. Ubicomp*. ACM.

[35] Valentin Radu and Mahesh K. Marina. 2013. Himloc: Indoor smartphone localization via activity aware pedestrian dead reckoning with selective crowdsourced wifi fingerprinting. In *Proc. IPIN*. IEEE.

[36] Valentin Radu, Catherine Tong, Sourav Bhattacharya, Nicholas D Lane, Cecilia Mascolo, Mahesh K Marina, and Fahim Kawsar. 2018. Multimodal Deep Learning for Activity and Context Recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 157.

[37] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 779–788.

[38] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

[39] Charissa Ann Ronao and Sung-Bae Cho. 2016. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications* 59 (2016).

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, and Michael Bernstein. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015).

[41] Muhammad Shoaib, Stephan Bosch, Ozlem Incel, Hans Scholten, and Paul Havinga. 2015. A survey of online activity recognition using mobile phones. *Sensors* 15, 1 (2015).

[42] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[43] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* (2012).

[44] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. 2015. Smart Devices are Different: Assessing and Mitigating Mobile Sensing Heterogeneities for Activity Recognition. In *The 13th ACM Conference on Embedded Networked Sensor Systems*.

[45] Akshayvarun Subramanya, Suraj Srinivas, and R. Venkatesh Babu. 2017. Confidence estimation in Deep Neural networks via density modelling. In *CVPR*.

[46] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 4489–4497.

[47] Zhuoling Xiao, Hongkai Wen, Andrew Markham, and Niki Trigoni. 2014. Lightweight map matching for indoor localisation using conditional random fields. In *Proc. IPSN*. ACM.

[48] Zhuoling Xiao, Hongkai Wen, Andrew Markham, and Niki Trigoni. 2014. Robust pedestrian dead reckoning (R-PDR) for arbitrary mobile device placement. In *IPIN*. IEEE.

[49] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. 2015. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition. In *IJCAI*.

[50] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How Transferable Are Features in Deep Neural Networks?. In *NIPS*.