An Empirical Comparison of DCNN libraries to implement the Vision Module of a Danger Management System

Sianna Puente C, Cindy E. Madrid, Miguel Realpe and Boris X. Vintimilla
Escuela Superior Politécnica del Litoral, ESPOL, CIDIS – FIEC
Campus Gustavo Galindo Km 30.5 Vía Perimetral, Guayaquil, Ecuador
[sppuente, cemadrid, mrealpe, boris.vintimilla] @espol.edu.ec

ABSTRACT

In this paper we discuss the feasibility of using a DCNN to implement Fall and Head detection for a Danger Management System. For this propose, AlexNet and Inception-v3 models in MatConvNet and TensorFlow libraries were used for training new DCNNs with Fine-Tuning method. Additionally, a new public dataset was created, which includes diverse fall poses, as well as, top views of people walking in a scene.

CCS Concepts

• Computing methodologies → Machine learning → Machine learning approaches→Neural networks.

Keywords

DCNN, Fall detection; Head detection; Tensorflow; Matconvnet; Danger Management Systems.

1. INTRODUCTION

Kishwar is an ongoing project which main goal is to develop a Danger Management Systems that will offer automated alarm management, supervision and control in order to secure the safety of people during emergency situations, especially those requiring building evacuation (e.g. fire disasters, hazardous material release, earthquakes, etc.). The Danger Management System needs to be able of reactively guiding people during emergency situations, in addition of detecting obstacles and fallen people near exits in order to change escape routes depending of the real-time conditions of the emergency and the building structure. Also, the system will provide information about the origin and state of the emergency with the intention of creating a better assessment of the current situation and possible rescue procedures.

The full Kishwar system (Figure 1) includes a sensor network that incorporates sensors such as video cameras, fire detectors and gas leak detectors for monitoring the current state of a building. However, the focus of the present work is to develop a computer vision system to detect people near exits, as well as, to identify the untoward event of a fallen person that may need some assistance or might represent some risk in case of evacuation. One important requirement of the system is real-time processing capability; likewise, the implemented solution should be not only accurate, but also fast. Thus, this paper compares the detection accuracy and the processing speed of two state-of-the-art Deep Convolutional Neural Networks (DCNN) libraries in order to choose the best option for the Kishwar's Vision Module.

A literature review is given in the next section of this paper. After that, the dataset generation and the training process are explained. Then, experimental results are shown. Finally, the last section depicts the conclusions and future works.

ICDLT '17, June 02-04, 2017, Chengdu, China
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5232-1/17/06…$15.00
http://dx.doi.org/10.1145/3094243.3094255

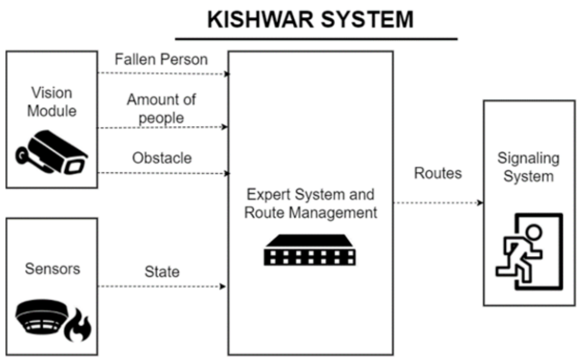

Figure 1. Kishwar's modules.

2. RELATED WORK

People detection is a key problem for many computer vision tasks and has been a widely studied problem in the past decades for many applications, such as recognition of human activity, security and pedestrian avoidance [1]. The basic human detection process consists of classifying patterns in the image into a specific class (usually, human or not human). A general pattern classification is posed as follows: Given a training sample S, consisting of n independent observations, the problem is to find a decision function that classifies new objects as accurately as possible. For that, many feature extraction and methods have been proposed. Some systems are based on simple features, for example Viola and Jones [2] propose a combination of several Haar-like filters such as lines and edges, which provide a decent performance and a low computation cost, but a high false alarm rate. In 2005, Dalal and Triggs [3] introduced the histograms of oriented gradients (HOG) feature for this pedestrians detection, which later was used by Felzenswalb et al. [4] in the deformable part model DPM, as well as, improved the detection performance and encouraged the use of complex features, such as skeleton and silhouette based ones [5, 6], these features have a good performance but a high computational cost that makes them unsuitable for real time applications.

In 2012 Krizhevsky [7] proposed a DCNN that improved the state-of-the-art accuracy of image traditional classification methods [8]. Afterwards, its specific network structure has become an increasingly popular tool in computer vision [9]. In addition to their high accuracy, the DCNNs have the ability of unsupervised feature learning. Because of that, these networks are used for the detection of objects [10], faces, people, heads [11] and fallen people [12], among other applications.

In the Kishwar project, the amount of people that enter and leave the building needs to be estimated. Many approaches have been proposed for detecting people in order to count them, the most popular are body detection [13], shoulder detection [14], head detection [11], and combinations of these. Nevertheless, head detection is the least affected by occlusions, especially in crowded situations and images acquired from a high point (above the average human height).

In [15] a CNN was implemented for head detection. The Neural Networks was used for features extraction, and then a SVM with Adaboost was applied for classification. Afterward, the results were compared with a HOG feature extraction algorithm. The CNNs provided better Correct Detection Rates (CDRs) but slower computing time.

In addition of counting people, the system has to include a detector for fallen people. The goal of fall detection is the subsequent implementation of monitoring systems with alert sending, most of the related works are focused on the analysis of fall detection for patients and elderly people in environments as homes and hospitals. In [16], recognition of lying poses in individual images is performed by analysing configurations of body parts such as limbs in regions detected by means of a structured tree model. Some works use information from sequences of video frames in order to perform tracking of an individual and human shape analysis. In [17], the silhouette of a person is extracted to analyse the human shape in order to differentiate a fall from some other activity, by comparing the changes in the orientation with defined thresholds. The complex nature of human motion makes thresholds less effective in detecting different kind of falls; thresholds are sensitive due to noisy trajectories or poor image segmentation, machine learning approaches are more appropriate to generalize detection of these fall poses. In [18] a feature vector is obtained by means of traditional techniques and a motions' classification is made by using a multi-class Support Vector Machine to determine a fall event, ten different poses are considered, three of them are forward, backward and sideway fall. In lateral views, there are usually more problems due to occlusion of objects. Deep learning approach attracts a lot of attention recently; DCNNs can be very efficient for pose recognition since there is no need to explicitly design feature representations and detectors for parts because they are learned directly from the data [19]. Previous works focused on differentiating between fall poses and ADLs (Activities of Daily Living), for example walking, standing, sitting, crouching, climbing or climbing stairs.

Since the cameras of the Kishwar project will be located in a high position near the exits, we have decided to apply head detection for counting people, in addition of a detector of fall poses performed on individual images of a simple scene acquired from a top view angle. Also, we chose DCNNs for classification, because on their accuracy. For the implementation we have compared two state-of-the-art DCNNs libraries in order to define the best option for the Kishwar's Vision Module based on their detection accuracy and processing speed.

3. PROPOSED APPROACH

As stated in the previous sections, DCNNs were selected for implementing the Kishwar's Vision Module (Fallen Person Detection and Head Detection) because they produce better CDRs than classic Machine Learning methods. We also decided to compare two different DCNNs libraries -i.e., TensorFlow and MatConvnet (section 3.2) - with a specific dataset in order to find the one with the best results for the project. Therefore, our next step was to define a dataset robust enough for our purpose.

Since top view images are needed for the training process, the use of popular pedestrians' dataset such as INRIA [3], Caltech-USA [20], and KITTI [21] were discarded because they do not offer the desire camera angle. Instead, we explored three top view people datasets: Edinburgh Informatics Forum Pedestrian Database [22], Top View Person Re-Identification Dataset (TVPR) [23] and UR Fall Detection Dataset [24]. The former is a dataset focused on people tracking, but it does not include enough raw data for training a neural network for people detection. The second one includes stereovision images of diverse people walking in the cameras' FOV, however no fall poses are provided and the latter provides fallen people frames but fall poses images are still very scarce. Due to this lack of a suitable training data, a new dataset that includes people walking and fall poses was created.

3.1 Dataset Generation[1]

The dataset was generated from videos recorded using a Basler acA1300 camera at 75 fps in top-view configuration (Figure 2). The videos were acquired in 4 sessions during varied days/times. Each session produced around of 10 videos, which include one or two people walking at the same time under the camera, obstacles in the environment and people simulating four different fall categories (face up, face down, on side, slip).

From these videos, 14,614 images were originally obtained (Table 1). Additionally, some transformations such as Gaussian noise, salt and pepper, and rotations were applied in order to increase the diversity and robustness of the dataset. As result, 146.124 images of 640x360 pixels were created. Some examples are shown in Figures 3-5.

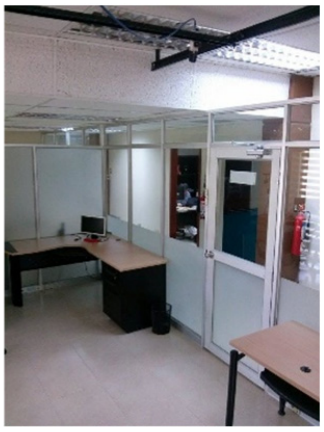

Figure 2: Top-view configuration for the dataset generation

[1] The whole set of images used for training and validation, as well as the obtained results, will be shared together with the proposed CNN's models.

3.2 DCNNs Training

The Fine-Tuning method has been applied in order to train the DCNNs. The idea of Fine-Tuning is that a pre-trained model will act as a feature extractor (until the next-to-last layer), then the last layer is trained for a new classification model. In this research, AlexNet and Inception-v3 models in MatConvNet and TensorFlow libraries were used for training new models with diverse images selected from the dataset (Table 2).

Three approaches were used for training the data: Head classification using TensorFlow, Fall classification using MatConvNet and classification comparison of libraries (for heads and falls). The next sections explain the libraries and training processes.

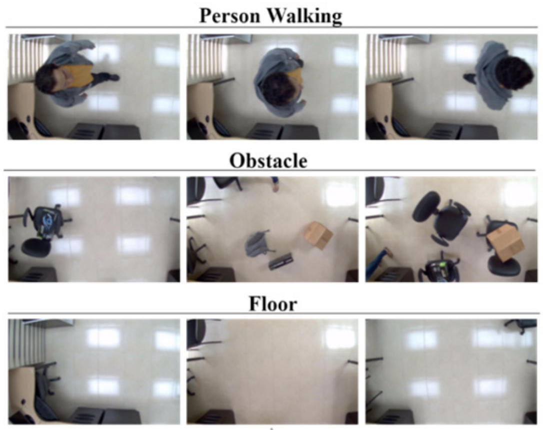

Figure 3: Images of scene with person walking, obstacles and floor.

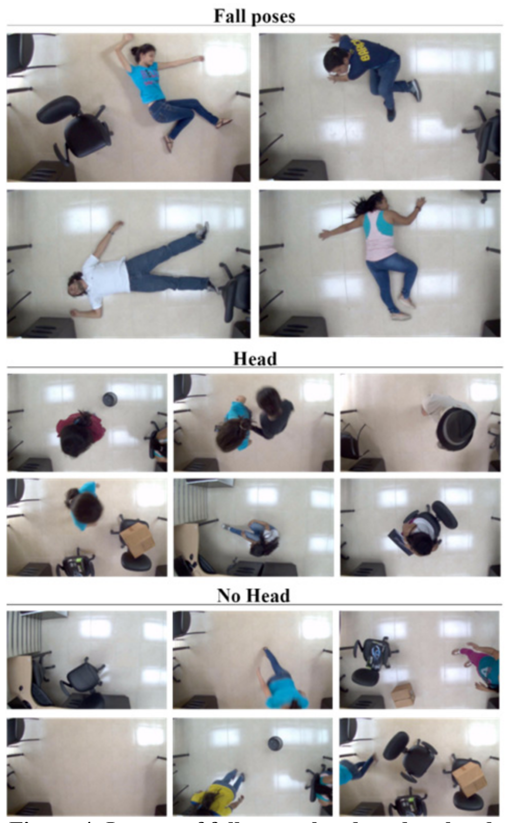

Figure 4: Images of fall poses, head, and no head.

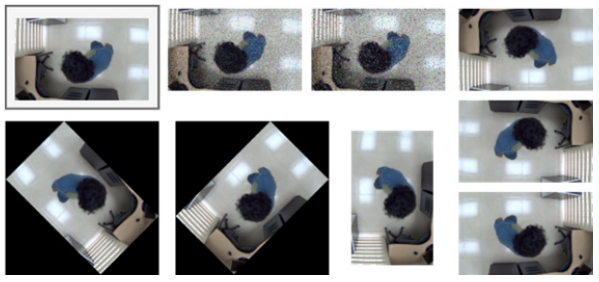

Figure 5: Original image with transformations

Table 1: Original number of images obtained for each category

Category	Subcategory	N° Images
Head	Head	3,662
	No Head	4,352
Fall	Fall1	1,114
	Fall2	800
	Fall3	1,077
	Fall4	1,135
Floor		2,019
Obstacle		455
TOTAL		14,614

Table 2: Number of images used for DCNNs training

Model	Category	N° Images
Head	Head	3,453
	No Head	4,250
Fall	Fall1	1,047
	Fall2	790
	Fall3	1,077
	Fall4	307
	Floor	455
	Obstacle	425
	Walk	1,007

3.2.1 TensorFlow training

TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows processing the data in one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

TensorFlow has good design considerations for neural network training, and at the same time avoids being totally a neural network framework. It includes graph collections, queues, image augmenters etc., which can be useful building blocks for a higher-level wrapper.

It was decided to use TensorFlow for head classification by applying the Fine-Tuning technique in the DCNN training. The available models in the TensorFlow library are: CIFAR10 and Inception-v3, however Inception-v3 is the only model that allows fine-tuning so far, which was chosen. Inception-v3 is really a great architecture and it is the result of multiple trial and error cycles, it often achieves the best performance for image recognition among other models.

Training with TensorFlow was performed on a Microsoft AZURE server with the following features: UBUNTU 16 Operating System, Intel Xeon R E5-2673 v3 (Haswell) processor, 4GB RAM and 500GB Hard Disk. Although TensorFlow has received a lot of attention for being a very promising framework, it currently only supports cuDNNv3 version and not the cuDNNv7.5 that was installed on the computer during the training processing.

All networks were trained with the same parameters, 10,000 training steps, 10% of data for testing and 10% of data for validation; values defined by recommendation in tutorials of the TensorFlow official page.

3.2.2 MatConvNet training

MatConvNet is a toolbox for the Convolutional Neural Networks implementation in the Matlab environment. MatConvNet allows easy manipulation of the DCNNs architecture by means of building blocks, it provides a variety of optimizations by supporting GPU calculations and a flexible use of existing models for Fine-Tuning application [25].

The implementation of this technique was performed with the pre-trained DCNN model AlexNet, which is one of the most popular models used in the field of DCNNs and winner of the ILSVRC-2012 challenge. In [7] its architecture is described, this network is composed of eight learned layers, five convolutional layers, max-pooling layers, dropout layers, and three fully connected layers, it was designed for classification of 1000 categories. In this existing model, the last fully-connected layer was replaced with random weights and new categories to be classified were specified, then dropout layers were added between the fully connected layers in order to avoid over-fitting. The "dropout" technique consists of randomly deactivating connections of neurons during the training process to avoid the dependence of a neuron with other particular neurons.

Before the procedure, the train and test sets were prepared with images of the proposed categories and respective labels were defined. A re-training of the network was then performed with the new dataset available. This approach indicates that the entire network were trained, but with better initialization of weights (first layers) and not random values, which would help make the learning of the network much faster and more effective.

The training was performed in Matlab 2016b with MatconvNet 1.0-beta20 version compiled for GPU on a computer with NVIDIA Quadro K2200 video card.

4. EXPERIMENTAL RESULTS

The accuracy was estimated based on the average of the Sensitivity or True Positive Rate (TPR) and the Specificity or True Negative Rate (TNR).

$$TPR = \frac{TP}{TP+FN} \quad (1)$$

$$TNR = \frac{TN}{TN+FP} \quad (2)$$

Table 3 and 4 show the result of the DCNN training processes using diverse amount of images randomly selected. The validation accuracy is estimated employing the same images used for training, while the test accuracy uses different images.

In order to assess the performance of the DCNNs trained for fall detection, 3400 images were tested (1730 falls and 1670 notfall). Similarly, 3700 test images were used for evaluating the head detection CNNs (1650 head and 2050 nohead). In both cases, image sets included original images and images after transformations were applied.

Table 3: Test results for Fall detection DCNN.

N° Images	Validation Accuracy	TPR	TNR	Test Accuracy
1,500	99.07%	97.93 %	82.45%	90.19%
2,500	99.52%	97.75 %	87.80%	92.77%
3,500	99.26%	98.91 %	88.28%	93.60%
4,500	99.29%	98.73 %	87.98%	93.35%
18,700	99.26%	97.98 %	94.94%	96.46%
49,500	99.51%	98.33 %	94.46%	96.40%

Table 4: Test results for Head detection DCNN.

N° Images	Validation Accuracy	TPR	TNR	Test Accuracy
2,906	97.80%	96.29 %	83.72%	90.00%
4,906	96.00%	96.05 %	85.04%	90.56%
6,906	97.60%	95.38 %	79.65%	88.76%
34,872	95.60%	96.05 %	94.33%	95.19%
58,872	96.40%	97.21 %	93.56%	95.38%
82,872	96.20%	96.68 %	93.41%	95.04%

In order to analyse the influence of transformations introduced into the dataset, the DCNNs were also trained using sets of images with and without transformations (1700 and 18700 images respectively) for 2 and 4 categories. These results are shown in Tables 5 and 6.

Table 5: Test results for Fall detection DCNN with and without transformation using 2 categories.

Transformations	Validation Accuracy	Fall (%)	Not Fall (%)	Test Accuracy
no	99.64%	97.12	82.90	90.01%
yes	99.26%	97.98	94.94	96.46%

Table 6: Test results for Fall detection DCNN with and without transformation using 4 categories.

Transformations	Validation Accuracy	Fall (%)	Floor (%)	Obst. (%)	Walk (%)	Test Accuracy
no	96.11%	86.82	95.05	85.85	86.50	88.55%
yes	96.37%	97.64	97.53	80.72	93.65	92.39%

4.1 Tensorflow vs Matconvnet

A new random set of 41000 images were used in order to compare the models of both libraries, based on the results of Table 3 and 4. These tests were realised in a Core i3 PC with 6GB RAM. The average processing time for each test image was 153.16 ms for Matconvnet and 0.022 ms for TensorFlow. Table 7 and 8 depict the results of both libraries trained for Head and Fall detection respectively.

Table 7: Libraries' results for Head detection.

Library	Val Accuracy	TPR	TNR	Test Accuracy
Tensorflow	96.00%	95.83%	93.65%	94.74%
Matconvnet	99.42%	99.58%	75.82%	87.70%

Table 8: Libraries' results for Fall detection.

Library	Val Accuracy	TPR	TNR	Test Accuracy
Tensorflow	98.00%	99.02%	98.21%	98.61%
Matconvnet	99.88%	99.45%	93.02%	96.24%

5. CONCLUSIONS AND FUTURE WORKS

The introduction of transformations into the original dataset increased the test accuracy of the DCNNs. In the case of 2 categories' DCNNs the accuracy increased 6%, while 4 categories' DCNNs increased around 3%. It could be appreciated that fall classification produce the lowest detection rates for obstacles, since many of the images in this categories were misclassified as background images with no people (floor) because they both share a lot of common characteristics. On the other hand, the category with the best accuracy was 'fall' because it has less overlapping with the other classes.

When comparing TensorFlow and Matconvnet results, it can be seen that TensorFlow produced the best result for Head detection and Fall detection with 94.74% and 98.61% accuracy, respectively. In addition, TensorFlow processing time (0.022 ms) allows real time application of the system, which is one requirement for Kishwar's vision module.

We are currently in the process of acquiring a larger dataset from 4 different top view perspectives of a scene (2 pair of parallel cameras), which will serve as the basis for new training data with a diverse combination of images. This dataset could also take advantage of the parallel arrange in order to create deep images that could provide more information for the classification process. Also, multiple people in the same scene will be introduced in the future dataset.

Since current dataset only includes a maximum of 2 people at the time, the system may suffer from occlusion for multiple moving objects. In addition a tracking process may need to be introduced in order to increase reliability for real scenarios.

6. ACKNOWLEDGMENTS

This work has been supported by the ESPOL under Project G4-DI-2014 "Integrated system for emergency management using sensor networks and reactive signaling" (Kishwar).

7. REFERENCES

[1] P. Turaga, R. Chellappa, V. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," IEEE Transactions on Circuits and Systems for Video Technology , vol. 18, pp. 1473 –1488, nov. 2008.

[2] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition, 1:511–518, 2001.

[3] Dalal, N., Triggs, B. : Histograms of oriented gradients f or human detection. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (2005) 886–893.

[4] Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: CVPR. (2008).

[5] Chen, Y.T.; Lin, Y.C.; Fang, W.H. A hybrid human fall detection scheme. In Proceedings of the International Conference on Image Processing, Hong Kong, China, 26–29 September 2010; pp. 3485–3488.

[6] Realpe, M., Vintimilla, B., Romero, D., Remagnino, P. (2009). Análisis de comportamiento humano: Metodologia para localización y seguimiento de personas en secuencias de video. Conferencia Iberoamericana en Sistemas, Cibernética e Informática.

[7] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097-1105.

[8] D. Tomè, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi, S. Tubaro, Deep Convolutional Neural Networks for pedestrian detection, Signal Processing: Image Communication, Volume 47, September 2016, Pages 482-489, ISSN 0923-5965, http://dx.doi.org/10.1016/j.image.2016.05.007.

[9] X.G. Chen, "Pedestrian Detection with Deep Convolutional Neural Network", Computer Vision-ACCV 2014 Workshops. Springer International Publishing, (2014)

[10] Yang, F., Choi, W., Lin, Y.: Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifier, CVPR (2016)

[11] Van Oosterhout, T., Bakkes, S., Kröse, B.J.: Head detection in stereo data for people counting and segmentation. In: VISAPP. (2011) 620–625.

[12] Niall McLaughlin, Jesus Martinez del Rincon, Paul Miller; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1325-1334

[13] Ouyang, W., Wang, X.: Joint deep learning for pedestrian detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 2056-2063

[14] Wang, S., Zhang, J., Miao, Z.: A new edge feature for head-shoulder detection. In: 2013 IEEE International Conference on Image Processing, IEEE (2013)

[15] Gao, C., Li, P., Zhang, Y., Liu, J., Wang, L.: People counting based on head detection combining adaboost and cnn in crowded surveillance environment. Neurocomputing (2016)

[16] Wang, S., Zabir, S., Leibe, B.: Lying pose recognition for elderly fall detection. Robotics: Science and Systems VII (2012) 345-353.

[17] Rougier, C., Meunier, J., St-Arnaud, A., Rousseau, J.: Robust video surveillance for fall detection based on human shape deformation. IEEE Transactions on Circuits and Systems for Video Technology 21 (2011) 611-622

[18] Foroughi, H., Rezvanian, A., Paziraee, A.: Robust fall detection using human shape and multi-class support vector machine. In: Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on, IEEE (2008) 413-420.

[19] Bearman, A., Dong, C.: Human pose estimation and activity classication using convolutional neural networks. CS231n Course Project Reports (2015)

[20] Dollar, P., Wojek, C., Schiele, B., Perona, P.: "Pedestrian detection: A benchmark". In: CVPR. (2009)

[21] Geiger, A., Lenz, P., Urtasun, R.: "Are we ready for autonomous driving? the kitti vision benchmark suite".

In: Conference on Computer Vision and PatternRecognition (CVPR). (2012).
[22] B. Majecka, "Statistical models of pedestrian behaviour in the Forum", MSc Dissertation, School of Informatics, University of Edinburgh, 2009.
[23] Liciotti, D., Paolanti, M., Frontoni, E., Mancini, A., Zingaretti, P. “Person Re-Identification Dataset with RGB-D Camera in a Top-View Configuration”, Video Analytics for Face, Face Expression Recognition, and Audience Measurement, Springer, 2017.
[24] Kwolek, B., & Kepski, M. (2014). Human fall detection on embedded platform using depth maps and wireless accelerometer. Computer methods and programs in biomedicine, 117(3), 489-501.
[25] Vedaldi, A., Lenc, K.: Matconvnet: Convolutional neural networks for matlab. In: Proceedings of the 23rd ACM international conference on Multimedia, ACM (2015) 689-692.